UNIVERSITY OF CALIFORNIA SAN DIEGO

Replicable Learning Algorithms

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Rex Lei

Committee in charge:

Professor Russell Impagliazzo, Chair
Professor Jelena Bradic
Professor Kamalika Chaudhuri
Professor Daniel M. Kane
Professor Shachar Lovett

2024

University of California San Diego

2024

DEDICATION

To my advisor, committee, collaborators, colleagues, teachers, friends, family; and, to everyone who believed in me.

EPIGRAPH

The best researcher is the one that is having the most fun.

Unknown

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

PREFACE

Like many other Ph.D. students, I did not begin my Ph.D. with the intention of writing this particular dissertation. The ideas that led to this dissertation first appeared essentially as a footnote — as a technical tool for a secondary result in our COLT 2021 conference paper on Massart boosting. We isolated those ideas, formally defining replicability in a short manuscript for the TPDP 2021 workshop. After we devised more complicated replicable algorithms, our expanded submission "Reproducibility in Learning" was accepted by STOC 2022, one of the major conferences in theoretical computer science. Our next paper, "Stability is Stable", showed a broad equivalence between replicability and differential privacy, another well-studied stability notion.

Replicability has become the dominant lens through which I approach research. For starters, one can ask about replicability for any algorithmic learning problem with data. First, design an algorithm that efficiently learns something about your data. Second, design an algorithm that efficiently learns the data replicably. Comparing the efficiency of the best algorithm and the best replicable algorithm tells us about the *cost* of replicability. In many cases, this cost is not insurmountable, but also not negligible.

Moreover, replicability research asks about the role of randomness in learning. Replicable algorithms sit on the boundary between random inputs and consistent outputs. By definition, the output of a replicable algorithm is *canonical* — it represents a property of your universe, not just the data you collected. When do these canonical objects exist? How can they be used? These questions, about discovering consistent signals in noisy data, are central to all sciences.

There are many different reasons for why you might be reading this dissertation. Regardless of your reasons, I hope you leave this document with a greater appreciation for replicability — as a lens to learn about new algorithms, as a model that permits clever mathematics, and as a medium for understanding what it means to be random.

ACKNOWLEDGEMENTS

To Russell, my advisor: *I have learned so much from you — about life, mathematics, collaboration, and advising. Thank you for all the support, kindness, and generosity over the years.*

To Daniel, Jelena, Kamalika, and Shachar, my committee: *Thank you for showing me your research style and helping me find my own.*

To my collaborators, both published and unpublished: *I am constantly amazed by your acumen, and I have had so much fun in our research meetings. Thank you for all the time and energy you have put into our projects.*

To previous UCSD Ph.D. students, who guided me in my early Ph.D. years: *Thank you for opening my eyes to how my life could be; thank you for encouraging me to have it all.*

To my Ph.D. cohort: *Thank you for sharing your joys, triumphs, anxieties, and failures with me, and thank you for letting me share mine with you.*

To the UCSD extended theory CS community, split into CSE, ECE, HDSI, math, and more: *Thank you for being welcoming and teaching me so much about mathematics. I wish we weren't so split.*

To the Columbia theory group: *Thank you for your kindness.*

To the UCSD community: *Thank you for making my life vibrant and exciting.*

To my friends in San Diego: *Thank you for giving me stability in times of instability, and instability in times of stability. Thank you for the wonderful adventures.*

To my friends not in San Diego: *Thank you for always giving me something to look forward to. I cherish the moments we get to spend together. Thank you for the wonderful adventures.*

To my family: *Thank you for your love.*

All the best,

Rex

CHAPTER ACKNOWLEDGEMENTS

Chapter 1, in part, is a reprint of the material as it appeared in three publications. The first, on Arxiv at https://arxiv.org/abs/2201.08430v2, which is the full version of the paper that appeared in the Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing 2022. Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sorrell, Jessica. "Reproducibility in Learning". The second, in the Proceedings of the 34th Annual Conference on Learning Theory 2022. Diakonikolas, Ilias; Impagliazzo, Russell; Kane, Daniel M.; Lei, Rex; Sorrell, Jessica; Tzamos, Christos. "Boosting in the Presence of Massart Noise". The third, on Arxiv at https://arxiv.org/abs/2303.12921v2, which is the full version of the paper that appeared in the Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing 2023. Bun, Mark; Gaboardi, Marco; Hopkins, Max; Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sivakumar, Satchit; Sorrell, Jessica. "Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization". The dissertation author was the primary investigator and author of these papers.

Chapter 2, in part, is a reprint of the material as it appeared in three publications. The first, on Arxiv at https://arxiv.org/abs/2201.08430v2, which is the full version of the paper that appeared in the Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing 2022. Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sorrell, Jessica. "Reproducibility in Learning". The second, in the Proceedings of the 34th Annual Conference on Learning Theory 2022. Diakonikolas, Ilias; Impagliazzo, Russell; Kane, Daniel M.; Lei, Rex; Sorrell, Jessica; Tzamos, Christos. "Boosting in the Presence of Massart Noise". The third, on Arxiv at https://arxiv.org/abs/2303.12921v2, which is the full version of the paper that appeared in the Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing 2023. Bun, Mark; Gaboardi, Marco; Hopkins, Max; Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sivakumar, Satchit; Sorrell, Jessica. "Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization". The dissertation author was the primary investigator and author of these papers.

Chapter 3, in full, is a reprint of the material as it appeared on Arxiv at https://arxiv.org/abs/2201.08430v2. This material is the full version of the paper that appeared in the Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing 2022. Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sorrell, Jessica. "Reproducibility in Learning". For this dissertation, the term "reproducibility" was modified to "replicability" where appropriate, and minor formatting edits were made to improve readability. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of the material as it appeared in Proceedings of the 34th Annual Conference on Learning Theory 2022. Diakonikolas, Ilias; Impagliazzo, Russell; Kane, Daniel M.; Lei, Rex; Sorrell, Jessica; Tzamos, Christos. "Boosting in the Presence of Massart Noise". For this dissertation, the term "reproducibility" was modified to "replicability" where appropriate, and minor formatting edits were made to improve readability. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of the material as it appeared on Arxiv at https://arxiv.org/abs/2303.12921v2. This material is the full version of the paper that appeared in the Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing 2023. Bun, Mark; Gaboardi, Marco; Hopkins, Max; Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sivakumar, Satchit; Sorrell, Jessica. "Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization". For this dissertation, minor formatting edits were made to improve readability. The dissertation author was the primary investigator and author of this paper.

VITA

2018	Bachelor of Science, Columbia University, New York City
2024	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Note: In the theoretical computer science community, author order is typically alphabetical and denotes equal contribution.

Ilias Diakonikolas, Russell Impagliazzo, Daniel M. Kane, Rex Lei, Jessica Sorrell, and Christos Tzamos. 2021. Boosting in the Presence of Massart Noise. Proceedings of Thirty Fourth Conference on Learning Theory, PMLR 134:1585-1644

Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell. 2022. Reproducibility in learning. In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022). Association for Computing Machinery, New York, NY, USA, 818831. https://doi.org/10.1145/3519935.3519973

Mark Bun, Marco Gaboardi, Max Hopkins, Russell Impagliazzo, Rex Lei, Toniann Pitassi, Satchit Sivakumar, and Jessica Sorrell. 2023. Stability Is Stable: Connections between Replicability, Privacy, and Adaptive Generalization. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC 2023). Association for Computing Machinery, New York, NY, USA, 520527. https://doi.org/10.1145/3564246.3585246

FIELDS OF STUDY

Computer Science

Studies in Theoretical Computer Science
Professor Russell Impagliazzo

ABSTRACT OF THE DISSERTATION

Replicable Learning Algorithms

by

Rex Lei

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Russell Impagliazzo, Chair

Reproducibility and replicability are central to the process of learning. Machine learning algorithms can solve a variety of tasks, in turn allowing researchers to ask new questions. However, reproducibility issues can limit the scope and utility of these algorithms. In this dissertation, we investigate replicability from a theoretical perspective. We formalize a new mathematical definition of replicability. Our definition applies to randomized algorithms that learn from independent and identically distributed (i.i.d.) samples. We propose replicable algorithms for fundamental learning tasks such as computing statistical queries, boosting weak learners, and learning halfspaces. We discuss techniques for designing replicable algorithms, resolving tensions among accuracy,

replicability, and efficiency. Furthermore, we construct black-box algorithmic reductions between replicability and other notions of algorithmic stability, such as differential privacy. We also note possible directions for future replicability research.

- The introduction is a general-audience explanation of the contents of this dissertation.
- In Chapter 1, we formally introduce replicability, the central concept of this dissertation. We mathematically define replicability and the associated model, noting key features and limitations. This chapter summarizes the theorems and algorithms in the remaining chapters. We mention promising future directions and argue why researchers should pursue them.
- In Chapter 2, we list central definitions and briefly note how they are used.

The remaining three chapters are reprints of three published conference papers related to replicable learning algorithms, with readability edits.

- In Chapter 3, we mathematically define replicability and motivate our definition. We design new replicable algorithms for fundamental learning tasks such as computing statistical queries and learning halfspaces. This chapter discusses basic properties of replicability, including alternate definitions.
- In Chapter 4, we exhibit a paper that predates and greatly contributed to this dissertation's work on replicability.
- In Chapter 5, we prove strong connections between replicability and other well-studied stability notions such as differential privacy. We show black-box ways to transform certain stable algorithms to replicable algorithms and vice versa. These transformations may increase the required sample sizes and run times. We prove these increases are necessary. Applications of our connections include new replicable learning algorithms and resolving open questions in algorithmic stability.

Introduction

This dissertation is concerned with the theoretical study of replicable algorithms. The following introduction is a general-audience explanation of what this entails.

Every day, we use algorithms to process our surroundings and inform our decision making. Algorithm users want algorithms to be efficient: to terminate quickly (run time), use little memory (space), require minimal data (sample sizes), and more. We also want algorithms to be accurate: to precisely compute statistics, deftly maneuver robots, faithfully simulate reality, and more.

In this dissertation, we investigate algorithms that are not only efficient and accurate, but also stable. Generally speaking, a stable algorithm can overcome irregularities, such as corrupted data, and still provide useful outputs. Designing stable algorithms expands the range of problems that can be solved. Stability can imply other desirable properties for algorithms, such as data privacy and better accuracy on unseen data.

Thanks to increasing computational resources and new algorithmic paradigms, many new questions can be investigated computationally. Across the sciences, researchers can use tools from machine learning and statistics to deduce correlations and infer causation. Often, machine learning algorithms are used iteratively — the results of one experiment inform the goals of the next.

Replicability is the phenomenon of reaching the same conclusion despite starting from different data. Replicability is often desired when creating new knowledge. Lack of replicability, if not caught immediately, may lead to pointless experiments with nonsensical conclusions. Or, lack of replicability could lead researchers to discover flaws in their understanding. Either way, investigating replicability itself is key to improving research.

This dissertation investigates replicability for machine learning algorithms, from a mathematical perspective. Broadly, we ask, “For which learning problems do there exist efficient replicable algorithms?” Our methodology involves defining a model, analyzing replicable algorithms, and comparing replicable algorithms with other (possibly non-replicable) algorithms.

To define a model for replicable learning algorithms, we need to define:

1. how input data is produced,
2. how algorithms can interact with the data,
3. what additional resources algorithms have (e.g., randomness),
4. which outputs are acceptable (correctness), and
5. what counts as replicability.

In order to permit meaningful mathematical guarantees, models make assumptions that can be unrealistic in practice. For example, a model may assume input data is chosen in a representative way. We should be careful to not wrongly extrapolate and make broad conclusions.

Instead, models are stepping stones, allowing us to use mathematics to explain phenomena in idealized settings. Models let us design new algorithmic techniques and explain why they work. Models let us quantify how efficient and effective algorithms are. Models let us define tough problems and explain why they are hard to solve. Later, we can carefully branch outwards, either by expanding the scope of the model or by switching to more experimental means.

Using the definitions of our model, we can analyze a replicable algorithm by:

1. writing a learning algorithm,
2. mathematically proving it is correct,
3. mathematically proving it is replicable, and

4. quantifying the computational resources needed to run the algorithm.

In this dissertation, we consider randomly generated inputs and allow algorithms to be randomized. Because of randomness, it might not be possible to design algorithms that are always correct and replicable. So, we focus on proving that our algorithms have these properties with high probability (low failure rate).

At this point, it might be helpful to discuss an example: computing statistical queries.

Problem. *Estimate the average height of a group of people.*

Say you are asked to estimate the average height of a group of one million people, but it is too costly to measure everyone individually. You could randomly sample a small subset, say 1000 people, and compute the average height. In fact, Law-of-Large-Numbers-style arguments tell us that small random subsets are sufficient to get very accurate statistics for the entire population.

However, this algorithm may not be replicable, even if it is accurate. For example, one sample may lead to a height estimate that is too large, and another sample may lead to a height estimate that is too small. Without knowing the true mean, which estimate should we believe?

In this case, there is a straightforward way to improve both accuracy and replicability: draw larger sample sets. Then, the sample means are not just more likely to be closer to the true mean, but also closer to each other.

We can do more to ensure replicability. The least significant digits of empirical means are probably a consequence of the sample, not of the population. By rounding these digits, we probably won't lose much accuracy. And, if estimates are close enough, we will round many estimates to the same number. Rounding might sacrifice closeness to the true mean (accuracy), but improves the likelihood of our algorithm producing a consistent output (replicability).

This approach can be summarized in the following algorithm sketch.

Algorithm 1. Replicable algorithm for computing average height

```
1: Draw a small random sample $S$
2: Compute an empirical average height $\hat{h}$ using $S$
3: Round $\hat{h}$ and output the result
```

By formalizing these arguments mathematically, we can prove this algorithm replicably computes statistics of populations. At the expense of larger sample sizes, the algorithm is both accurate and replicable (most of the time).

Statistical query algorithms (i.e., algorithms using only statistical queries) have a rich academic literature and many applications. This simple argument implies that all such applications can be computed replicably.

Designing replicable algorithms is a principal goal of this dissertation. Another central task is connecting replicability to other stability notions. There are many ways an algorithm could be stable, depending on the choice of the model. We show that, for many stability notions N, it is possible to transform algorithms that are replicable into algorithms that satisfy N, and vice versa. These connections connect replicability research to existing literature, automatically producing algorithms in new contexts. They also consolidate techniques for designing stable algorithms.

Our mathematical investigation of replicable algorithms takes many other forms in this dissertation. We discuss what replicable algorithms cannot do (lower bounds), properties of replicability, algorithmic techniques, proof techniques, applications of our results, and promising future directions. This dissertation contains direct connections from replicability to algorithmic stability, statistical learning theory, geometry, and cryptography.

Chapter Outline

In Chapter 1, we formally introduce replicability, the central concept of this dissertation. We mathematically define replicability and the associated model, noting key features and limitations of our choices. We discuss the inspirations of the model, as well as follow-up research. We summarize the theorems and algorithms in the remaining chapters, discussing some techniques used to design and analyze these algorithms. We mention some promising future directions, and we argue why researchers should pursue them.

In Chapter 2, we list central definitions and briefly note how they are used.

The remaining three chapters are reprints of three published conference papers related to replicable learning algorithms, with readability edits. For full bibliographic information, see the chapter acknowledgements pages in the preliminary pages.

In Chapter 3, we mathematically define replicability and motivate our definition. We design new replicable algorithms for fundamental learning tasks such as computing statistical queries, heavy-hitters, and approximate medians of distributions. We also give replicable algorithms for weak learning halfspaces and boosting weak learners into strong learners. This chapter discusses basic properties of replicability, including alternative definitions and a lower bound for replicable statistical queries.

In Chapter 4, we exhibit a paper that predates and greatly contributed to this dissertation's work on replicability. This paper is not about replicability — rather, it concerns using boosting algorithms to learn noisy data. However, replicability makes its appearance in the lower bound argument (a secondary result of the paper). Replicability is combined with cryptographic assumptions to prove the optimality of our primary result, a new Massart noise boosting algorithm. To better understand the motivations for the project and its early usage of replicability, we include the paper in its entirety. Chapter 1 contains a more detailed explanation of this paper's relationship with replicability research.

In Chapter 5, we prove strong connections between replicability and other well-studied stability notions such as differential privacy. We show that, for a wide array of statistical tasks, there are black-box ways to transform certain stable algorithms to replicable algorithms and vice versa. These equivalences imply that many stability notions are not distinct, but instead variations on the same theme. These transformations may increase the needed sample sizes, and we show that these sample increases are necessary. One transformation increases the computational complexity as well, and we use standard cryptographic assumptions to show that the computational increase is necessary. Furthermore, we show that the usage of the cryptographic assumption is necessary, by arguing that computational efficiency would be achievable if that cryptographic assumption is false. Investigating correlated sampling is key to our computational results. In addition, we give applications of our new connections between replicability and other stability notions. These include designing new learning algorithms and resolving open questions in algorithmic stability.

Chapter 1

What Is Replicability?

1.1 Why Replicability?

Reproducibility is vital to ensuring scientific conclusions are reliable, and researchers have an obligation to ensure that their results are replicable. In the last twenty years, lack of reproducibility has been a major concern in most scientific fields. For example, a 2012 Nature article by Begley and Ellis reported that the biotechnology company Amgen was only able to replicate 6 out of 53 landmark studies in haematology and oncology [BE12]. In a 2016 Nature article, Baker published a survey of 1500 researchers, reporting that 70% of scientists had tried and failed to replicate the findings of another researcher, and that 52% believed there is a significant crisis in reproducibility [Bak16].

A key issue underlying the "reproducibility crisis" and "credibility revolution", as articulated in many articles (e.g., [Ioa05]), is the massive increase of new data and publications. This has come with explosion of methods for data generation, screening, testing, and analysis, where only the combinations producing the most significant results are reported. P-hacking, data dredging, and researcher degrees of freedom can lead to erroneous findings that appear significant but cannot be replicated by other researchers.

Identifying and mitigating these problems is quite subtle. First, is not easy to come up with an agreed-upon set of practices that guarantees reproducibility. Second, testing whether a finding is statistically significant is a complex task.

Within machine learning and data science, there are similar concerns about the reliability of published findings. For example, the performance of models produced by machine learning algorithms may be highly dependent on the values of random seeds or hyperparameters chosen during training [HIB$^+$17, IHGP17, LKM$^+$18]. To begin addressing concerns about reproducibility, several prominent machine learning conferences started hosting reproducibility workshops and challenges. These are part of a broader effort to encourage researchers to share their methodology/code and create a new academic culture supporting replicability [PVLS$^+$20].

In this dissertation, we initiate the mathematical study of replicable learning algorithms. We define replicability solely as a property of algorithms, rather than the process by which their results are collected and reported. We define a new notion of replicability — informally, it says that a randomized algorithm is replicable if, with high probability, two runs of the algorithm on two distinct sets of samples drawn from the same distribution, with internal randomness fixed between both runs, produces the same output.

Replicability is an extremely strong stability constraint to place on an algorithm. Nevertheless, replicability is achievable for many fundamental data analysis tasks, including computing statistical queries, identifying heavy hitters, and learning large-margin halfspaces.

Replicability is not the first algorithmic stability definition aimed at improving the utility of machine learning algorithms. Other definitions play central roles in more mature research areas, such as differential privacy and adaptive data analysis. In this dissertation, we show that there are black-box reductions between replicable algorithms and algorithms that satisfy other well-studied stability notions.

1.2 Defining Replicability

Definition 1.2.1 (Replicability)**.** *Let D be a distribution over domain $\mathscr{X}$. Let $\mathscr{A}$ be a randomized algorithm that takes as input samples from D. We say that $\mathscr{A}$ is ρ-replicable if*

$$\mathbf{Pr}_{S_1,S_2,r}[\mathscr{A}(S_1;r) = \mathscr{A}(S_2;r)] \geq 1-\rho,$$

where S_1, S_2 are sets of samples drawn independently and indentically distributed (i.i.d.) from D and r represents the internal randomness of $\mathscr{A}$.

Let's start by discussing the model's scope.

Which algorithms? Algorithms with . . .

No implementation issues: In our model, algorithms do exactly as they are told. We overlook any possible implementation issues of the algorithms, including those that might lead to nonreplicability. Possible issues may arise from running algorithms using different hardware, programming languages, operating systems, random number generators, etc.

One might expect that running an algorithm twice on the same computational system should always automatically yield the same result. However, especially for complex machine learning models, it can be difficult to resolve every possible discrepancy between independent algorithm runs. Ensuring algorithmic reproducibility, the phenomenon of producing the same output when starting from the same input, is an active area of machine learning research. Instead, this dissertation assumes there are no implementation issues with the algorithms and instead focuses on replicability.

Access to samples: Definition 1.2.1 concerns algorithms that learn from samples. These samples are given to the algorithm, and then the algorithm produces some output. Other interaction between the algorithm and its data source, perhaps through directly querying the data source for specific information, is not allowed.

Access to i.i.d. samples: Furthermore, the samples are drawn independently and identically distributed (i.i.d.) from some unknown distribution. The assumption of a distribution is very powerful. To define replicability, we want to say two separate runs of the algorithm will yield similar outputs. There needs to be some underlying reason why we should expect *any* algorithm to yield similar results for different inputs. For us, that reason is that samples are drawn from the same underlying distribution.

The independence of samples allows us to apply tools such as concentration inequalities to prove high probability bounds. However, it means our model does not cover situations where the data may not be so strongly related. For example, elements in data sets could depend on each other, some small proportion of the data could be corrupted, or the distribution may evolve over time.

To some extent, the assumption of i.i.d. samples greatly simplifies achieving replicability. In many machine learning applications, acquiring usable sample data is incredibly hard and a source of nonreplicability. Here, we assume there is no issue in the sampling portion of the learning task. For this and other reasons, our definition of replicability is an algorithmic property, not a property of the entire process of learning.

Access to randomness: We consider algorithms that can make random decisions. To formalize this, we give algorithms access to random coin flips (bits $b \in \{0,1\}$), which can be used at any point for any algorithmic purpose. For example, these bits can be used to randomly sample a number in $[0,1]$ or randomly invoke subroutines.

Modeling randomness with individual bits is standard in the study of randomized algorithms, but it is not the only way. One limitation is that it requires an infinite number of bits to write any number in $[0,1]$. One could circumvent this limitation by proving that approximations are sufficient (e.g., truncate numbers after n-many bits) or by redefining the model to permit more powerful random operations. For more information on the study of randomized algorithms, there are many good resources (e.g., [MR95]).

In this dissertation, we do not focus on quantifying randomness needed to run replicable algorithms. Nevertheless, it is a very interesting question to determine how limited access to randomness can affect replicable algorithms.

Any inputs and outputs: No restrictions are made on the types of inputs and outputs of the algorithm. They can be real numbers, polynomials, neural networks — whatever. However, if we want our algorithms to be efficient, we usually restrict our attention to inputs and outputs that can be efficiently represented.

Which algorithms are replicable?

Our definition formalizes replicability by considering what happens when we run the algorithm twice, with different samples but the same randomness r.

Per Definition 1.2.1, $\mathscr{A}$ is ρ-replicable if $\mathbf{Pr}_{S_1,S_2,r}[\mathscr{A}(S_1;r) = \mathscr{A}(S_2;r)] \geq 1-\rho$, where $S_1, S_2 \sim_{i.i.d.} D$ and r is random.

Replicability parameter ρ: Replicability is measured as a probability. Replicability $\rho = 0$ implies an algorithm that is always consistent, and replicability $\rho = 1$ implies the algorithm is never consistent. In fact, replicability $\rho = 1$ implies that, fixing any random string r, the output $A(S;r)$ is different for every possible input sample S.

For all distributions D: In this setting, the distribution D is unknown and only accessible through random samples. Nevertheless, we require algorithms to be replicable on all distributions. This property has some interesting consequences. For example, the outputs of any randomized algorithm form a distribution, so replicable algorithms can be combined sequentially and analyzed.

Another consequence is using replicable algorithms for distribution testing. Say you have a sample S drawn i.i.d. from some unknown distribution D in a small family of distributions $\{D_1, \ldots, D_n\}$. If a replicable algorithm A outputs different answers for each distribution, you can identify D by comparing $A(S;r)$ with $A(S_i;r)$ for S_i drawn i.i.d. from D_i and multiple random strings r. An example of this approach appears in Section 5.4.1.

Exact same outputs: Under these conditions, an algorithm is replicable if it outputs the *exact same* output. It may seem difficult to ensure this, especially when solving problems with infinite output domains. Nevertheless, it is achievable for a variety of tasks, including when outputs are real numbers or functions. One may consider algorithms that produce “approximately equal” outputs, but this has its own slew of complications. Say your algorithm outputs descriptions of new algorithms. Depending on your definition of approximate equality, the same algorithm could be either replicable or not replicable. Furthermore, approximate equality may lose transitivity — $A \approx B$ and $B \approx C$ does not necessarily imply $A \approx C$.

In this dissertation, we interpret “exact same” as having the same representation. Nevertheless, there are many ways to relax the “exact same output” condition to create new, interesting models. For example, computer programs can be written in many different ways. However, testing if two programs (Turing machines) behave identically on all inputs is impossible (undecidable). Relaxing the equality constraint of our replicability definition could lead to new connections between replicability and indistinguishability.

Shared randomness r: Another key component of our definition of replicability is shared randomness. An algorithm is only required to behave replicably when the same random string r is used for both runs. One way to view this is by thinking of a randomized algorithm as a collection of deterministic algorithms, indexed by their randomness. Each deterministic algorithm may perform well (i.e., be replicable and accurate) on certain samples and perform poorly on others.

We want our algorithms to be replicable on samples from unknown distributions. (If we knew the distribution, there’s no need to learn something about the distribution.) Certain distributions may produce certain “bad” samples more often than others, so a replicable algorithm needs to be a mixture of deterministic algorithms that do not all fail on the same bad samples.

Using the same random string r naturally defines a canonical output Z_r for each string r. In other words, if the algorithm is very replicable, then $\mathbf{Pr}_{S\sim D}[A(S;r) = Z_r] > .5$ for many random strings r. This observation is the key to understanding the relationship between the single-parameter definition (ρ-replicability) and the two parameter definition (η, ν)-replicability. See Section 3.8 for more details.

Algorithmic stability: To reiterate, we define replicability as an algorithmic property. We care not just about what an algorithm outputs (its distribution of outputs), but how it outputs them (if that distribution of outputs correlates with the randomness *r*).

Practical Limitations

What are the practical limitations of mathematical replicability research? Although reproducibility and replicability are key to scientific discovery, naively replacing algorithms with replicable algorithms (as defined in Definition 1.2.1) will not alone solve existing replicability issues in science.

As discussed, this definition abstracts away the implementation of algorithms across differing systems and programming languages. Data is assumed to be nicely independent and identically distributed. Writing replicable pseudocode is not sufficient to create an entire algorithmic system that is replicable. Instead, replicable algorithms can be combined with reproducible computing environments to create overall systems that are replicable.

Mathematical replicability can be too costly in samples. For example, our replicable statistical query algorithm has an approximately $1/\rho^2$ factor increase in its sample complexity, compared to nonreplicable statistical queries. In Section 3.7, we show this is essentially tight. Depending on the context, a 10000-fold increase in samples may be too much to pay for 99% replicability. Changing the objective may help — one could transform the data, focus on average-case instead of worst-case analysis, or pivot to relaxed notions of replicability.

Using a replicable learning algorithm does not enforce honest research. Just like with p-hacking, a researcher could search for random strings *r* or data samples *S* that pair well. Then, they could publish their most significant result, falsely suggesting that their input data was randomly chosen. Dishonest or nonreplicable research can result from perverse incentives, which cannot be mitigated by simply restricting which algorithms can be used.

Optimistically, this replicability research is new — future researchers may find ways to circumvent these and other issues.

1.3 An Example: Replicable Statistical Queries

To better explain the definition of replicability, we explain and analyze the first algorithm in Chapter 3. This algorithm replicably computes statistical queries. For full details, see Section 3.2.

A query is a function ϕ from data domain $\mathscr{X}$ to real values in $[0,1]$. A statistical query for distribution D over $\mathscr{X}$ estimates the average value of $\phi(x)$ when $x \sim D$. A statistical query oracle $\mathscr{O}_D$ is an algorithm that takes queries ϕ and outputs estimates for $\mathbb{E}_{x \sim D}\phi(x)$. $\mathscr{O}_D$ is allowed a small failure probability δ and tolerance τ. Formally:

Definition 1.3.1 (Statistical query oracle)**.** *Let $\tau \in [0,1]$ and $\phi : \mathscr{X} \to [0,1]$ be a query. Let D be a distribution over domain $\mathscr{X}$. A* statistical query oracle *for D, denoted $\mathscr{O}_D(\tau,\phi)$, takes as input a tolerance parameter τ and a query ϕ, and outputs a value v such that $|v - \mathbb{E}_{x\sim D}[\phi(x)]| \leq \tau$.*

The statistical query model introduced by [Kea98] is a restriction of the PAC-learning model introduced by [Val84]. Many learning problems can be solved using only statistical queries (SQs). Therefore, replicably simulating an SQ oracle implies that these problems all have replicable algorithms.

Definition 1.3.2 (Simulating a statistical query oracle)**.** *Let $\delta \in [0,1]$ and τ, ϕ, D be as above. Let $\mathscr{O}_D$ be a statistical query oracle for D. Let $\vec{s}$ denote an i.i.d. sample drawn from D. We say that a routine* `STAT` *simulates* $\mathscr{O}_D$ *with failure probability δ if for all τ, δ, ϕ, there exists an $n_0 \in \mathbb{N}_+$ such that if $n > n_0$, $v \leftarrow \mathtt{STAT}(\tau, \phi, \vec{s})$ satisfies $|v - \mathbb{E}_{x\sim D}[\phi(x)]| \leq \tau$ except with probability δ.*

To simulate an SQ oracle with tolerance τ and replicability ρ, our algorithm first estimates the statistical query to tolerance $\tau' \approx \tau\rho$. Then, the interval $[0,1]$ is divided into intervals of size $\alpha \approx 2\tau/(1+\rho)$. These intervals do not start at 0; rather, they start at a value α_{off} chosen randomly in $[0,\alpha]$. The algorithm then finds the interval in which the statistical query falls, and outputs the midpoint of that region.

The randomness of the algorithm r is only used to choose α_{off}. By randomly rounding empirical estimates, we can guarantee that multiple runs of the algorithm have one canonical output to return — the midpoint of the closest interval. The better tolerance τ' is necessary to ensure that multiple estimates will land in the same interval.

Even with the better tolerance, the randomness r and random offset α_{off} are crucial for replicability. A replicable algorithm does not know the distribution producing its samples, and it must be prepared for all distributions. A deterministic rounding scheme is nonreplicable when the true mean $\mathbb{E}_{x\sim D}\phi(x)$ is near a rounding boundary. By randomizing the rounding boundaries, we can argue that empirical estimates avoid rounding boundaries with high probability.

Algorithm 2. $\texttt{rSTAT}_{\rho,\tau,\phi}(\vec{s})$
Parameters: τ - tolerance parameter
ρ - replicability parameter
ϕ: a query $X \to [0,1]$

```
1: $\alpha = \frac{2\tau}{\rho+1-2\delta}$
2: $\alpha_{\text{off}} \leftarrow_r [0,\alpha]$
3: Split $[0,1]$ in regions: $R = \{[0,\alpha_{\text{off}}), [\alpha_{\text{off}}, \alpha_{\text{off}}+\alpha), \ldots, [\alpha_{\text{off}}+i\alpha, \alpha_{\text{off}}+(i+1)\alpha), \ldots, [\alpha_{\text{off}}+k\alpha, 1)\}$
4: $v \leftarrow \frac{1}{|\vec{s}|} \sum_{x\in\vec{s}} \phi(x)$
5: Let $r_v$ denote the region in $R$ that contains $v$
6: **return** the midpoint of region $r_v$
```

The following theorem upper bounds the sample complexity of $\texttt{rSTAT}_{\tau,\rho,\phi}$. In Section 3.7, we show this upper bound is tight as a function of ρ.

Theorem 1.3.3 (`rSTAT` simulates a statistical query oracle)**.** *Let $\tau, \delta, \rho \in [0,1]$, $\rho > 2\delta$, and let $\vec{s}$ be a sample drawn i.i.d. from distribution D. Then if*

$$|\vec{s}| \in \tilde{O}\left(\frac{1}{\tau^2(\rho - 2\delta)^2}\right)$$

$\texttt{rSTAT}_{\rho,\tau,\phi}(\vec{s})$ *ρ-replicably simulates an SQ oracle $\mathscr{O}_{D,\tau,\phi}$ with failure rate δ.*

The following is a proof sketch. See Section 3.2 for full details.

Proof. Assume sample set $\vec{s}$ contains at least $\frac{4\log(2/\delta)}{2\tau^2(\rho-2\delta)^2}$ examples.

Tolerance (accuracy): $\texttt{rSTAT}_{\rho,\tau,\phi}$ simulates an SQ oracle $\mathscr{O}_{D,\tau,\phi}$ with failure rate δ.

Let $\tau' = \frac{\tau(\rho-2\delta)}{\rho+1-2\delta}$. By a Chernoff bound, empirical estimate $\frac{1}{|\vec{s}|}\sum\limits_{x\in\vec{s}} \phi(x)$ is within τ' of the true mean with failure probability δ. Moving the empirical estimate to the midpoint of region r_v can further offset the estimate by $\alpha/2$, giving overall tolerance τ.

Replicability: To show $\texttt{rSTAT}_{\rho,\tau,\phi}$ is ρ-replicable, consider running $\texttt{rSTAT}_{\rho,\tau,\phi}$ with common randomness r on samples $\vec{s}_1, \vec{s}_2 \sim D$ independently. By the same Chernoff bound, with probability at least $1-2\delta$, both empirical estimates of $\mathbb{E}_{x\sim D}[\phi(x)]$ are within tolerance τ' of the true mean. `rSTAT` only outputs different results between the two runs if the estimates do not land in the same random intervals. Since α_{off} is chosen uniformly in $[0,\alpha]$, this happens with probability at most $2\tau'/\alpha = \rho - 2\delta$. Combining $\texttt{rSTAT}_{\rho,\tau,\phi}(\vec{s})$ is ρ-replicable. $\square$

This example exhibits the tradeoffs among replicability, tolerance (accuracy), and sample complexity. In order to add replicability and achieve the same tolerance, we needed to increase the sample complexity and start from a smaller tolerance. Optimizing these tradeoffs is a key component of replicability research. For statistical query algorithms, in Section 3.7 we show this algorithm's tradeoffs are essentially tight.

To reiterate, shared randomness r is only used in `rSTAT` to pick random offset α_{off}. The random offset makes the algorithm replicable for all distributions D.

The main replicability technique of `rSTAT` is randomized rounding. If multiple runs of the algorithm are guaranteed to produce close outputs, rounding can convert these into a single, canonical output. The statistical query algorithm involves rounding in one dimension, on the interval $[0,1]$. A standard concentration bound guarantees closeness of outputs. In other words, concentration plus randomized rounding implies replicability.

In subsequent algorithms, this concept been generalized and applied more broadly. In Section 3.5, we replicably solve the problem of learning halfspaces in d dimensions. There, we combine a sum-of-vectors concentration bound with d-dimensional rounding schemes to create a replicable halfspace weak learning algorithm. Throughout Chapter 5, we investigate and use correlated sampling in our algorithms and reductions. Correlated sampling is a tool to sample from close distributions and, using shared randomness, output the same elements as often as possible. The randomized rounding procedure in `rSTAT` is a simple example of a correlated sampling procedure. Section 2.4 briefly describes correlated sampling and where it explicitly appears in this dissertation.

1.4 Properties of Replicability

We observe the following key properties of Definition 1.2.1.

Stability. Replicability is a strong stability property that implies independent parties can replicate previous results with high probability, so long as the randomness used to achieve these results is made public.

Generalization. Replicability implies generalization. A replicable learning algorithm, with high probability, outputs a hypothesis h such that the difference between the risk of h and the empirical risk of h on the training set is small. Intuitively, replicabilitiy implies that h is independent of the training set with high probability. Thus, a Hoeffding bound can be applied to bound the risk in terms of the empirical risk.

Privacy. Differential privacy (DP) is an important notion that requires small distance between the two distributions induced by an algorithm, when run on any two datasets that differ in a single element. Crucially, it asks for the guarantees in the *worst case over datasets*. Replicable algorithms guarantee a different form of privacy: If $\mathscr{A}$ is replicable, then what $\mathscr{A}$ learns (for example, a trained classifier) is almost always the same; thus, $\mathscr{A}$ is typically *independent* of the chosen training data. In this way, replicable algorithms are prevented from memorizing anything that is specific to the training data, similar to differentially private algorithms. Replicability is weaker than differential privacy in the sense that replicability only applies to in-distribution samples, whereas differential privacy applies to *any* training set. On the other hand, replicability is stronger in the sense that its guarantee for in-distribution samples is global rather than local.

Testability. While differential privacy has become the standard for privacy-preserving computation, an important issue that is the subject of extensive research is testing and verifying differential privacy. As discussed in [GNP20], DP-algorithms and their implementations are usually analyzed by hand, and proofs of differential privacy are often intricate and prone to errors. Implementing such an algorithm in practice often gives rise to DP leaks, due to coding errors or assumptions made in the proof that do not hold on finite computers (such as the ability to sample from continuous distributions). Moreover, the complexity of verifying differential privacy is hard. Verification in the black-box setting (where the auditor has oracle access to the learning algorithm) was recently shown to be infeasible, as low query complexity implies high values of the the privacy parameters ε and δ [GM18]. In the white-box setting where $\mathscr{A}$ is given to the tester, [GNP20] shows that testing for differential privacy is $coNP^{\#P}$-complete. This has led to an active research area aiming at developing automated as well as interactive testing and verification methods for differential privacy [NFPH15, GHH$^+$13, RP10, AH18, BGA$^+$15, BCK$^+$21, FJ14, ZK17]. In contrast, replicability is a form of privacy that can be *efficiently tested* in (randomized) polynomial time (in the dimension of the data universe and ρ) for individual distributions D.

1.4.1 Connections between Replicability and Algorithmic Stability Notions

Let us briefly recall the types of algorithmic stability that arise in these other areas:

Differential privacy.

A randomized algorithm is differentially private [DMNS16] if changing a single input record results in a small change in the distribution of the algorithm's output. When each input record corresponds to one individual's datum, differential privacy guarantees that nothing specific to any individual can be learned from the output of the algorithm. (See Section 5.2.4.) Differential privacy comes with a rich algorithmic toolkit and understanding of the feasibility of fundamental statistical tasks in query estimation, classification, regression, distribution estimation, hypothesis testing, and more.

Generalization in adaptive data analysis.

Generalization is the ability of a learning algorithm to reflect properties of a population, rather than just properties of a specific sample drawn from that population. Techniques for provably ensuring generalization form a hallmark of theoretical machine learning. However, generalization is particularly difficult to guarantee in settings where multiple analyses are performed adaptively on the same sample. Traditional notions of generalization do not hold up to downstream misinterpretation of results. For example, a classifier that encodes detailed information about its training sample in its lower order bits may generalize well, but can be used to construct a different classifier that behaves very differently on the sample than it does on the population. Interactive processes such as exploratory data analysis or feature selection followed by classification/regression can ruin the independence between the training sample and the method used to analyze it, invalidating standard generalization arguments.

Adaptivity in data analysis has been identified as one contributing factor to the replication crisis, and imposing stability conditions on learning algorithms offers solutions to this part of the problem. A variety of such stability conditions have been studied [DFH$^+$15a, DFH$^+$15b, BNS$^+$21, RZ16, CLN$^+$16, BF16, RRT$^+$16, BMN$^+$18, LS19, SZ20], each offering distinct advantages in terms of the breadth of their applicability and the quantitative parameters achievable. Two specific notions play a central role in this work. The first is *perfect generalization* [CLN$^+$16, BF16], which ensures that whatever can be inferred from the output of a learning algorithm when run on a sample S could have been learned just from the underlying population itself:

Definition 1.4.1. *An algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ is $(\beta, \varepsilon, \delta)$-perfectly generalizing if, for every distribution D over $\mathscr{X}$, there exists a distribution Sim_D such that, with probability at least $1-\beta$ over S consisting of n i.i.d. samples from D, and every set of outcomes $\mathscr{O} \subseteq \mathscr{Y}$,*

$$e^{-\varepsilon}(\mathbf{Pr}_{Sim_D}[\mathscr{O}] - \delta) \le \mathbf{Pr}[A(S) \in \mathscr{O}] \le e^{\varepsilon}\mathbf{Pr}_{Sim_D}[\mathscr{O}] + \delta. \tag{1.1}$$

The second is *max-information* [DFH$^+$15a] which constrains the amount of information revealed to an analyst about the training sample:

Definition 1.4.2. *An algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ has (ε, δ)-max-information with respect to product distributions if for every set of outcomes $\mathscr{O} \subseteq (\mathscr{Y} \times \mathscr{X}^n)$ we have*

$$\mathbf{Pr}[(A(S), S) \in \mathscr{O}] \le e^{\varepsilon}\mathbf{Pr}[(A(S), S') \in \mathscr{O}] + \delta,$$

where S and S' are independent samples of size n drawn i.i.d. from an arbitrary distribution D over $\mathscr{X}$.

As with differential privacy, both perfect generalization and max-information are robust to post-processing.

Each stability definition described above is tailored to model a distinct desideratum. At first glance, they may all appear technically incomparable. For instance, differential privacy is stricter than the other definitions in that it holds in the worst case over all input datasets without any assumptions on the data-generating procedure. On the other hand, it is weaker in that it only requires insensitivity to changing one input record, rather than to resampling the entire input dataset as in max-information, perfect generalization, or replicability. Meanwhile, differential privacy, max-information, and perfect generalization quantify the sensitivity of the algorithm's output in a weaker way than replicability; the former three notions only require that the distributions on outputs are similar, whereas replicability demands that precisely the same output realization is obtained with high probability.

Nevertheless, the (surprising!) technical connections between these definitions have enabled substantial progress on the fundamental questions in their respective areas. For example, it was exactly the adaptive generalization guarantees of differential privacy that kickstarted the framework of adaptive data analysis from [DFH$^+$15b]; the definition of max-information was subsequently introduced [DFH$^+$15a] to unify existing analyses based on differential privacy and description length bounds. As another illustration, variants of replicability were introduced in [BLM20, GGKM21, GKM21] for purely technical reasons, as it was observed that such algorithms could be immediately used to construct differentially private ones. This connection was essential in proving the characterization of private PAC learnability in terms of the Littlestone dimension from online learning [ALMM19, BLM20]. In fact, this characterization shows, that, in principle a private PAC learner using n samples can be converted to a replicable PAC learner using a number of samples that is an exponential tower of height n, but it is non-constructive and does not suggest what such a learner looks like in general.

1.4.2 Terminology: "Reproducibility" and "Replicability"

Chapter 3 is based on [ILPS22], which was titled "Reproducibility in Learning". In subsequent work, including this dissertation, we use the term "replicability" to refer to the same mathematical definition introduced in [ILPS22].

This terminology choice is more in line with the most recent Association for Computing

Machinery (ACM) guidance regarding artifact review and badging [Ass20], version 1.1, updated on August 24, 2020. This update changed the ACM's definitons of the terms "reproducible" and "replicable" to be more agreeable with the terminology currently used by the National Academies of Sciences, Engineering and Medicine (see Chapter 3: Understanding Reproducibility and Replicability, page 46, in [Nat19]).

According to both the ACM's and National Academies' current definitions, "reproducibility" refers to the ability of a second experimental group to obtain similar results using the *same* input data. Meanwhile, "replicability" refers to the ability of a second experimental group to obtain similar results using input data and methods that may be different than those used by the original experimental group.

The mathematical definition introduced in [ILPS22] is a guarantee that, with high probability, two executions of the same algorithm with the same randomness and different sample sets will produce the same answer. Since this guarantee is over different sample sets, the mathematical definition does not fit the "same input data" condition in the above definitions of reproducibility. Instead, the mathematical definition is a specific type of replicability — if the second experimental group runs the same algorithm with the same random string (but on a new sample), the two groups' results are guaranteed to be identical with high probability.

1.5 Results in this Dissertation

Next, we explain the main results of Chapters 3, 4, and 5.

1.5.1 Chapter 3: Replicability in Learning

This chapter establishes basic properties of our definition of replicability, designs efficient replicable algorithms for some fundamental statistical tasks, and investigates basic properties of replicability such as amplification and composition.

Replicability: Properties and Alternative Definitions

We discuss alternative definitions of replicability and show that they are all essentially equivalent. Then, we also prove some other nice properties of replicable algorithms.

1. **Alternative Definitions and Amplification.** We start by discussing two alternative definitions of replicability and relate them to our definition. First, we can generalize the definition to include algorithms $\mathscr{A}$ that not only have access to internal randomness and to random samples from an underlying distribution D, but that also have access to extra non-random inputs. This more general definition captures both the original definition of pseudodeterministic algorithms as well as our definition of replicable learning algorithms, and all of our results remain unchanged. Second, we discuss an alternative two-parameter definition, and show that the definitions are qualitatively the same. We show how to amplify the replicability parameter by a standard argument where the sample complexity is increased modestly.

2. **Public versus Private Randomness.** Recall that we define replicability as the probability that an algorithm returns the same answer when run twice using different random samples from D but the same internal randomness. In [GL19], the authors define a related concept in which the internal randomness is divided into two pieces, public and private randomness, but the algorithm should return the same answer when just the public randomness is held fixed. We show that, without loss of generality, it suffices to use only public randomness.

3. **Replicability Implies Generalization.** Learning algorithms attempt to use finite samples to generate hypotheses on unknown, possibly complex distributions. The error of a hypothesis h on the underlying distribution is called the generalization error. A replicable algorithm outputs the same hypothesis with high probability, and thus the algorithm seldom draws distinctions between specific samples and the entire distribution.

4. **Connections to Data Reuse.** We explore the connection between replicable algorithms and

the adaptive data analysis model discussed in [DFH$^+$15b] and [DFH$^+$15a]. We show that replicable algorithms are strongly resilient against adaptive queries. Informally, with respect to replicable algorithms, the sample complexity and accuracy of (replicably) answering m adaptively chosen queries behaves similarly to the sample complexity and accuracy of replicably answering m *nonadaptively* chosen queries.

Upper Bounds

Our main technical results are replicable algorithms for some well-studied statistical query and learning problems that are used as building blocks in many other algorithms.

1. **Simulating SQ Algorithms.** In Section 3.2, we give a generic algorithm that reduces the problem of ρ-replicably estimating a single statistical query with tolerance τ and error δ to that of *nonreplicably* estimating the same query within a smaller tolerance and error.

   Theorem 1.5.1 (Theorem 3.2.3, Restated). *Let $\psi : \mathscr{X} \to \{0,1\}$ be a statistical query. Then the sample complexity of ρ-replicably estimating ψ within tolerance τ and error δ is at most the sample complexity of (nonreplicably) estimating ψ within tolerance $\tau' = \tau\rho$ and error $\delta' = \tau\delta$.*

   The basic idea is to obtain an estimate of the statistical query with a smaller tolerance τ' and then use a randomized rounding scheme where the interval $[0,1]$ is divided into intervals of size roughly τ/ρ. Then, every value in the interval is rounded to the midpoint of the region it occurs in. The partition into intervals is chosen with a random offset so that with high probability nearby points will lie in the same region.

2. **Heavy-hitters.** Using our simulation of SQ queries, in Section 3.3, we demonstrate the usefulness of replicability by giving a replicable algorithm `rHeavyHitters` for identifying approximate v-heavy-hitters of a distribution, i.e. the elements in the support of the distribution with probability mass at least v.

Lemma 1.5.2 (Lemma 3.3.3, Restated)**.** *For all $\varepsilon \in (0,1/2)$, $v \in (\varepsilon, 1-\varepsilon)$, with probability at least $1-\rho$,* `rHeavyHitters`$_{\rho,v,\varepsilon}$ *is ρ-replicable, and returns a list of v'-heavy-hitters for some $v' \in [v-\varepsilon, v+\varepsilon]$. Furthermore, the sample complexity is bounded by $\widetilde{O}(\rho^{-2})$.*

The high level idea of our algorithm is to first draw sufficiently many samples, $\vec{s}_1$, $Q_1 = |\vec{s}_1|$, so that with high probability all heavy-hitters are in $\vec{s}_1$. In the second stage, we draw a fresh set $\vec{s}_2$ of Q_2 many samples and use them to empirically estimate the density of each element in $\vec{s}_1$, and remove those that aren't above the cutoff v', where v' is chosen randomly from $[v-\varepsilon, v+\varepsilon]$ to avoid boundary issues.

3. **Median Finding.** In Section 3.4, we design a replicable algorithm for finding an approximate median in an arbitrary distribution over a finite domain. Approximate median finding is a fundamental statistical problem, and is also extensively studied in the privacy literature.

   Theorem 1.5.3 (Theorem 3.4.2, Restated)**.** *Let $\tau, \rho \in [0,1]$ and let $\delta = 1/3$. Let D be a distribution over $\mathscr{X}$, where $|\mathscr{X}| = 2^d$. Then* `rMedian`$_{\rho,d,\tau,\delta}$ *is ρ-replicable, outputs a τ-approximate median of D with success probability $1-\delta$, and has sample complexity*

$$\tilde{\Omega}\left(\left(\frac{1}{\tau^2(\rho-\delta)^2}\right)\cdot\left(\frac{3}{\tau^2}\right)^{\log^*|\mathscr{X}|}\right)$$

   To describe the key ideas in the algorithm, we first show how approximate-median finding is useful for turning many algorithms into replicable ones. Consider any problem where the correct answers form an interval, and assume we start with a (not-necessarily) replicable algorithm that is mildly accurate. Then we can run a replicable approximate-median finding algorithm on the distribution of outputs of the original algorithm to construct a very accurate replicable algorithm.

   We will actually use this strategy recursively to replicably solve approximate median itself. Our algorithm recursively composes a mildly accurate replicable median algorithm with a generic very accurate non-replicable median algorithm. This recursive technique is inspired by, but simpler than, previous algorithms in the privacy literature [BNSV15, KLM$^+$20], and like these algorithms, the sample complexity of our algorithm has a non-constant but very slowly growing dependence on the domain size.

4. **Learning Halfspaces.** In Section 3.5, we obtain a replicable algorithm `rHalfspaceWkL` for weakly learning halfspaces. In Section 3.6, we transform it into a replicable strong learner by way of a replicable boosting algorithm `rBoost`. We stress that our algorithms for halfspaces are replicable in the stronger distribution-free setting.

   Theorem 1.5.4 (Corollary 3.6.5, Restated)**.** *Let D be a distribution over $\mathbb{R}^d$, and let $f : \mathbb{R}^d \to \{\pm 1\}$ be a halfspace with margin τ in D. For all $\rho, \varepsilon > 0$. Algorithm* `rBoost` *run with weak learner* `rHalfspaceWkL` *ρ-replicably returns a hypothesis* $\mathbf{h}$ *such that, with probability at least $1-\rho$, $\mathbf{Pr}_{\vec{x}\sim D}[\mathbf{h}(\vec{x}) = f(\vec{x})] \geq 1-\varepsilon$. Furthermore, the overall sample complexity is $\widetilde{O}\left(\frac{d^{10/9}}{\tau^{76/9}\rho^{20/9}\varepsilon^{28/9}}\right)$.*

   In order to replicably learn halfspaces, we start with a simple weak learning algorithm for halfspaces [Ser02] that takes examples $(\vec{x}_i, y_i) \in \mathscr{X} \times \{\pm 1\}$, normalizes them, and returns the halfspace defined by vector $\sum_i \vec{x}_i \cdot y_i$. We show a concentration bound on the sum of normalized vectors from a distribution, and then argue that all vectors within the concentration bound are reasonable hypothesis with non-negligible advantage.

   Our randomized rounding scheme is a novel application of the randomized rounding technique developed in the study of foams [KORW12]. The concentration bound together with the foams rounding scheme [KORW12] yields a replicable halfspace weak learner. We then obtain our replicable strong learner for halfspaces by combining it with a (new) replicable boosting algorithm. Our algorithm is sample-efficient but inefficient with respect to runtime, due to the inefficiency of the foams rounding scheme. We also give another randomized rounding procedure that gives a polynomial-time strong replicable halfspace learner, but with polynomially larger sample complexity.

The Price of Replicability.

In Section 3.7 we ask what is the cost of turning a nonreplicable algorithm into a replicable one. We first show that a τ-tolerant ρ-replicable SQ algorithm $\mathscr{A}$ for ϕ implies a ρ-replicable algorithm for the τ-coin problem: given samples from a p-biased coin with the promise that either $p \geq 1/2 + \tau$ or $p \leq 1/2 - \tau$, determine which is the case. Our main result in this section are nearly tight upper and lower bound bounds of $\Theta(\tau^{-2}\rho^{-2})$ on the sample complexity of ρ-replicably solving the τ-coin problem (for constant δ), and thus the same bounds for ρ-replicably answering SQ queries. On the other hand, it is well-known that the *nonreplicable* sample complexity of the τ-coin problem is $\Theta(\tau^{-2}\log(1/\delta))$ (see, e.g. [Mou]). So the cost of guaranteeing ρ-replicability for SQ queries is a factor of ρ^{-2}.

For upper bounds, our generic algorithm in Section 3.2 converts any SQ query into a replicable one: if our end goal is a ρ-replicable algorithm for estimating a statistical query with tolerance τ and error δ, then the sample complexity is at most the sample complexity of *nonreplicably* answering the query to within tolerance τ', and success probability $1-\delta'$ where $\tau' = O(\tau\rho)$ and $\delta' = O(\delta\tau)$, which has sample complexity $O(\tau^{-2}\rho^{-2}\log(1/\delta'))$. The main result in this section is the following lower bound for ρ-replicably answering statistical queries.

Theorem 1.5.5 (Theorem 3.7.1, Restated)**.** *Let* $\tau > 0$ *and let* $\delta < 1/16$. *Any* ρ*-replicable algorithm for solving the* τ*-coin coin problem with success probability at least* $1-\delta$ *requires sample complexity* $\Omega(\tau^{-2}\rho^{-2})$.

1.5.2 Chapter 4: Massart Boosting

This chapter is devoted to investigating boosting algorithms that are robust to Massart noise. Its main connection to this dissertation is the use of replicability to prove a lower bound theorem —

assuming one-way functions exist, no (computationally efficient) Massart noise boosting algorithm can achieve better accuracy than our boosting algorithm. Specifically, we construct a replicable, adversarial, "rude" weak learning algorithm `rWkL`. The material in this chapter was published prior to that of the other chapters, and portions of `rWkL` became the replicable statistical query and replicable heavy-hitters algorithms in Chapter 3.

Replicability ideas appear solely in the lower bound section of Chapter 4. Nevertheless, the chapter contains the full paper, so that an interested reader can fully appreciate the context of one of the first applications of replicability theory.

In the remainder of this section, we explain the statement of the lower bound and summarize how replicability is used in that argument. A reader familiar with boosting and Massart noise can skip to the end of this section, immediately before Section 1.5.3, for an explanation of the role replicability plays.

We study the problem of boosting the accuracy of a weak learner in the (distribution-independent) PAC model with Massart noise. In the Massart noise model, the label of each example x is independently misclassified with probability $\eta(x) \leq \eta$, where $\eta < 1/2$. The Massart model lies between the random classification noise model and the agnostic model. Our main positive result is the first computationally efficient boosting algorithm in the presence of Massart noise that achieves misclassification error arbitrarily close to η. Prior to our work, no non-trivial booster was known in this setting. Moreover, we show that this error upper bound is best possible for polynomial-time black-box boosters, under standard cryptographic assumptions. Our upper and lower bounds characterize the complexity of boosting in the distribution-independent PAC model with Massart noise. As a simple application of our positive result, we give the first efficient Massart learner for unions of high-dimensional rectangles.

Boosting is a general learning technique that combines the outputs of a weak base learner — a learning algorithm with low but non-trivial accuracy — to obtain a hypothesis of higher accuracy. Boosting has been extensively studied in machine learning and statistics since initial work by

Schapire [Sch90]. Here we study boosting in the context of learning classes of Boolean functions with a focus on Valiant's distribution-independent PAC model [Val84]. During the past three decades, several efficient boosting procedures have been developed in the *realizable* PAC model, i.e., when the data is consistent with a function in the target class. On the other hand, boosting in the presence of *noisy data* remains less understood.

In this work, we study the complexity of boosting in the presence of *Massart noise*. In the Massart (or bounded noise) model, the label of each example x is flipped independently with probability $\eta(x) \leq \eta$, for some parameter $\eta < 1/2$. The flipping probability $\eta(x)$ is bounded but is unknown to the learner and can depend on the example x in a potentially adversarial manner. Formally, we have the following definition.

Definition 1.5.6 (PAC Learning with Massart Noise)**.** *Let $\mathscr{C}$ be a concept class over $X = \mathbb{R}^n$, D_x be any fixed but unknown distribution over X, and $0 \leq \eta < 1/2$ be the noise parameter. Let $f \in \mathscr{C}$ be the unknown target concept. A* noisy example oracle*, $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, works as follows: Each time $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ is invoked, it returns a labeled example (x, y), where $x \sim D_x$, $y = f(x)$ with probability $1 - \eta(x)$ and $y = -f(x)$ with probability $\eta(x)$, for an* unknown *function $\eta(x) : X \to [0, \eta]$. Let D denote the joint distribution on (x, y) generated by the above oracle. A learning algorithm is given i.i.d. samples from D and its goal is to output a hypothesis h such that with high probability the misclassification error $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y]$ is as small as possible. We will use $\mathrm{OPT} \stackrel{\mathrm{def}}{=} \inf_{g \in \mathscr{C}} \mathbf{Pr}_{(x,y)\sim D}[g(x) \neq y]$ to denote the optimal misclassification error.*

Background on Massart Noise.

The Massart model is a natural semi-random input model that is more realistic and robust than random classification noise. Noise can reflect computational difficulty or ambiguity, as well as random factors. For example, a cursive "e" might be substantially more likely to be misclassified as "a" than an upper case Roman letter. Massart noise allows for these variations in misclassification rates, while not requiring precise knowledge of which instances are more likely to be misclassified.

That is, algorithms that learn in the presence of Massart noise are likely to be less brittle than those that depend on uniformity of misclassification noise. Agnostic learning is of course even more robust, but unfortunately, it can be computationally infeasible to design agnostic learners for many applications.

Boosting With Noisy Data.

An important research direction, which was asked in Schapire's original paper [Sch90], is to design boosting algorithms in the presence of noisy data. This broad question has been studied in the past two decades by several researchers. See Section 4.1.4 for a detailed summary of related work. Specifically, prior work has obtained efficient boosters for RCN [KS03] and agnostic noise [Ser03, Fel10]. It should be emphasized that these prior works do not immediately extend to give boosters for the Massart noise setting.

In this work, we ask the following question:

Can we develop efficient boosting algorithms for PAC learning with Massart noise?

Our focus is on the distribution-independent setting. Given a distribution-independent Massart weak learner for a concept class $\mathscr{C}$, we want to design a distribution-independent Massart learner for $\mathscr{C}$ with high(er) accuracy. Prior to this work, no progress had been made on this front. *We resolve the complexity of the aforementioned problem by providing (1) an efficient boosting algorithm and (2) a matching computational lower bound on the error rate of any black-box booster.*

This work is the first step of the broader agenda of developing a general algorithmic theory of boosting for other "benign" semi-random noise models, lying between random and fully adversarial corruptions.

Our main result is the first computationally efficient boosting algorithm for distribution-independent PAC learning in the presence of Massart noise that guarantees misclassification arbitrarily close to η, where η is the upper bound on the Massart noise rate. To state our main result, we will require the definition of a Massart weak learner (see Definition 4.2.5 for additional details).

Definition 1.5.7 (Massart Weak Learner)**.** *Let $\alpha, \gamma \in (0, 1/2)$. An (α, γ)-Massart weak learner* `WkL` *for concept class $\mathscr{C}$ is an algorithm that, for any distribution D_x over examples, any function $f \in \mathscr{C}$, and any noise function $\eta(x)$ with noise bound $\eta < 1/2 - \alpha$, outputs a hypothesis h that with high probability satisfies $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y] \leq 1/2 - \gamma$, where D is the joint Massart noise distribution.*

Theorem 1.5.8 (Main Result)**.** *There exists an algorithm* `Massart-Boost` *that for every concept class $\mathscr{C}$, given samples to a Massart noise oracle $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, where $f \in \mathscr{C}$, and black-box access to an (α, γ)-Massart weak learner* `WkL` *for $\mathscr{C}$,* `Massart-Boost` *efficiently computes a hypothesis h that with high probability satisfies $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y] \leq \eta(1 + O(\alpha))$. Specifically,* `Massart-Boost` *makes $O(\log^2(1/\eta)/\gamma^2)$ calls to* `WkL` *and draws*

$$\mathrm{polylog}(1/(\eta\gamma))/(\eta\gamma^2)\, m_{\mathtt{WkL}} + \mathrm{poly}(1/\alpha, 1/\gamma, 1/\eta)$$

samples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, where $m_{\mathtt{WkL}}$ is the number of samples required by `WkL`.

Prior to this work, no such boosting algorithm was known for PAC learning with Massart noise. Moreover, as we explain in Section 4.1.4, previous noise-tolerant boosters do not extend to the Massart noise setting. In Section 4.1.3, we provide a detailed overview of our new algorithmic ideas to achieve this.

Some additional comments are in order. First, we note that the $\eta + \varepsilon$ error guarantee achieved by our efficient booster can be far from the information-theoretic minimum of $\mathrm{OPT} + \varepsilon$. The error guarantee of our generic booster matches the error guarantee of the best known polynomial-time learning algorithm for Massart halfspaces [DGT19]. Interestingly, the learning algorithm of [DGT19] can be viewed as a specialized boosting algorithm for the class of halfspaces, in which the halfspace structure is used to downweight specific regions on which the current classifier achieves high accuracy. Theorem 4.1.3 is a broad generalization of this result that applies to *any* concept class. This connection was one of the initial motivations for this work.

A natural question is whether the error upper bound achieved by our booster can be improved.

Perhaps surprisingly, we show that our guarantee is best possible for black-box boosting algorithms (under cryptographic assumptions). Specifically, we have the following theorem:

Theorem 1.5.9 (Lower Bound on Error of Black-Box Massart Boosting). *Assuming one-way functions exist, no polynomial-time boosting algorithm, given black-box access to an (α, γ)-Massart weak learner, can output a hypothesis h with misclassification error $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y] \leq \eta(1 + o(\alpha))$, where η is the upper bound on the Massart noise rate. In particular, this statement remains true on Massart distributions with optimal misclassification error* $\mathrm{OPT} = \mathbb{E}_{x\sim D_x}[\eta(x)] \ll \eta$.

The reader is referred to Theorem 4.4.1 for a detailed formal statement. Our lower bound establishes that the error upper bound achieved by our boosting algorithm is best possible.

We show that no "black-box" generic boosting algorithm for Massart noise can have significantly better error than that for our algorithm, i.e., $\eta + \Theta(\eta\alpha)$. While this seemingly matches the lower bound for RCN boosting from [KS03], the RCN bound only implies a lower bound for RCN weak learners in the special case of Massart noise when $\eta = \mathrm{OPT}$. We show a similar lower bound in the Massart noise setting for a small but polynomial value of OPT. That is, Massart noise boosting algorithms cannot be improved even when only a very small fraction of instances are actually noisy.

To prove our lower bound, we consider a situation where the function r to be learned is highly biased, and there is a tiny fraction of inputs with the majority value that are noisy and indistinguishable from non-noisy inputs. If the distribution queried by a boosting algorithm does not reweight values in some way to favor the minority answer, an uncooperative weak learner can return the majority answer and have advantage γ. Doing so, the weak learner provides essentially no additional information to the boosting algorithm. Alone, the boosting algorithm cannot efficiently learn the function f, by the definition of pseudorandomness.

The boosting algorithm could try to reweight values, in an attempt to extract information from its weak learners. However, it risks adding too much noise to the small fraction of already noisy examples, violating the Massart condition. Again, we invoke pseudorandomness — if the boosting algorithm manages to avoid incorrectly reweighting the noisy examples x with nonnegligible probability, then it must have some advantage in inverting the function f, (which partially determines which examples are noisy).

We formalize these intutions by exhibiting an adversarial weak learner `rWkL`. `rWkL` returns a hypothesis h that outputs the maximum likelihood label for each heavy-hitter of given distribution D' and outputs a constant value for non-heavy-hitters. `rWkL` has a stability property called *replicability*. Using replicability, we argue that i) boosting with `rWkL` can be efficiently simulated without knowing the function f and ii) `rWkL` satisfies the definition of a Massart noise weak learner. We conclude that a black-box boosting algorithm must be able to efficiently learn pseudorandom functions in order to extract useful information from `rWkL` or achieve misclassification error better than $\eta + \Theta(\eta\alpha)$.

Replicability implies that, with high probability, `rWkL` does not reveal much about its input S. If `rWkL` were not replicable, it could leak information about which samples likely influenced its output. In turn, the boosting algorithm could attempt to use this information in subsequent rounds to get better than $\eta + \Theta(\eta\alpha)$ misclassifcation error. While it may be possible to argue that some nonreplicable weak learner cannot be used by any boosting algorithm to achieve error $o(\eta(1+\alpha))$, `rWkL`'s replicability immediately implies that the boosting algorithm is not learning from `rWkL`.

In order to invoke pseudorandomness to prove properties of black-box boosting algorithms, we first show how to use boosting algorithms' subroutines to construct efficient distinguishers for f. Using pseudorandomness, we prove `rWkL`'s input does not even have to be drawn from the original distributions (the "honest example generator", Algorithm 20). Rather, with high probability, giving `rWkL` random input with correct marginal distributions (Algorithm 22) yields the same output.

Substituting `hEG` with `rEG` saves the distinguisher from making additional function queries, simplifying the analysis. As boosting algorithms are iterative, subsequent rounds of computation may depend on the results of previous rounds. In absense of this substitution, one needs to be careful that simulating the sample generation process (for the weak learner in a boosting algorithm) does not require superpolynomial queries, and thus superpolynomial time. With this substitution, all nested subroutines can be simulated quickly, by using random samples.

1.5.3 Chapter 5: Stability is Stable

This chapter establishes new connections and separations between replicability and standard notions of algorithmic stability. In particular, we give sample-efficient algorithmic reductions between perfect generalization, approximate differential privacy, and replicability for a broad class of statistical problems. Conversely, we show any such equivalence must break down computationally: there exist statistical problems that are easy under differential privacy, but that cannot be solved replicably without breaking public-key cryptography. Furthermore, these results are tight: our reductions are statistically optimal, and we show that any computational separation between DP and replicability must imply the existence of one-way functions.

Our statistical reductions give a new algorithmic framework for translating between notions of stability, which we instantiate to answer several open questions in replicability and privacy. This includes giving sample-efficient replicable algorithms for various PAC learning, distribution estimation, and distribution testing problems, algorithmic amplification of δ in approximate DP, conversions from item-level to user-level privacy, and the existence of private agnostic-to-realizable learning reductions under structured distributions.

Equivalences

Our main result is a complete characterization of the relationships between these quantities. We prove that all four central stability notions — replicability, differential privacy, perfect generalization, and bounded max-information w.r.t. product distributions — are equivalent to one another via constructive conversions that incur at most a near-quadratic overhead in sample complexity.

Our equivalences apply to an abstract and broad class of *statistical tasks* that capture learning from i.i.d. samples from a population. An instance of such a task is obtained by considering a distribution D from a pre-specified family of distributions. Given i.i.d. samples from D, the goal of a learning algorithm is to produce an outcome that is "good" for D with high probability. This formulation of a statistical task captures problems such as PAC learning, where a sample from D is a pair $(x, f(x)) \in \mathscr{X} \times \{0,1\}$ where x is drawn from an arbitrary marginal distribution over $\mathscr{X}$, and f is an arbitrary function from a fixed concept class H. A "good" outcome for such a distribution D is a hypothesis $h : \mathscr{X} \to \{0,1\}$ that well-approximates f on D. Many other objectives such as regression, distribution parameter estimation, distribution learning, hypothesis testing, and confidence interval construction can be naturally framed as statistical tasks. (See Section 5.6.4 for other examples.)

The following figure illustrates the known relationships between the various stability notions that hold with respect to any statistical task.

From these equivalences we obtain the following consequences, resolving several open questions.

Sample-efficient replicable algorithms.

Any differentially private algorithm solving a statistical task (with a finite outcome space) can be converted into a replicable algorithm solving the same task with a near-quadratic blowup in its sample complexity. Thus, the wealth of research on private algorithm design can be brought to bear on designing replicable algorithms. We illustrate this algorithmic paradigm by describing new replicable algorithms for some PAC learning, distribution parameter estimation, and distribution testing problems in Section 5.6.4.

Equivalence between perfect generalization and differential privacy.

For simplicity, the relationships summarized in Figure 5.1 are stated in terms of a one-way variant of perfect generalization, where only the inequality on the right of (5.1) is required to hold. But the original two-way definition turns out to be statistically equivalent for tasks with a finite outcome space. This is because a one-way perfectly generalizing algorithm can be converted to

$$\begin{tikzcd}
\text{(0.1)-replicability} \arrow[rr, dashed, "n \mapsto n \cdot \frac{\log(1/\delta)}{\varepsilon}\text{, Thm. 5.3.1 [GKM21]}"] \arrow[d, dashed, bend left, "n \mapsto n \cdot \frac{\mathrm{poly}\log(1/\varepsilon, 1/\delta)}{\varepsilon^2}\text{ Thm. 5.3.19}"] & & \text{(ε,δ)-differential privacy} \arrow[d, dashed, "n \mapsto n^2\text{ Cor. 5.3.13 [RRST16]}"'] \\
\text{$(\delta,\varepsilon,\delta)$-one-way perfect generalization} \arrow[u, dotted, bend left, "n \mapsto n\text{ Thm. 5.3.17}"] & & \text{(ε,δ)-max-information w.r.t. product distributions} \arrow[ll, "\text{Lem. 5.3.14}"]
\end{tikzcd}$$

Figure 1.1. Algorithmic relationships among replicability, differential privacy, max information, and perfect generalization.
The solid arrow from A to B means that every algorithm satisfying A also satisfies B. A dashed arrow means that for every statistical task, a solution satisfying A can be computationally efficiently transformed into a solution satisfying B with the stated blowup in sample complexity. The thin dotted arrow means an explicit transformation exists, but is not always computationally efficient, and assumes the outcome space is finite.
This figure suppresses constant factors everywhere and polynomial factors in δ, assumes ε is below a sufficiently small constant, and assumes that δ is a sufficiently small inverse polynomial in n.

a replicable algorithm using Theorem 5.3.17, and Theorem 5.3.19 actually yields the stronger conversion back to a two-way perfectly generalizing algorithm (See Theorem 5.6.3). Thus, an (ε,δ)-differentially private algorithm (with a finite outcome space) can be converted to a perfectly generalizing one solving the same statistical task with a near quadratic blow-up in sample complexity. This resolves an open question of [CLN$^+$16]. Their work also gave a conversion from perfectly generalizing algorithms to differentially private ones with no sample complexity overhead, and while their transformation preserves accuracy for (agnostic) PAC learning, it is not clear how to analyze it for general statistical tasks. Our conversion from perfect generalization to replicability and then to differential privacy holds for all statistical tasks with a finite outcome space.

Converting item-level to user-level privacy.

Consider a "user-level" learning scenario in which n individuals each hold m training examples drawn i.i.d. from the same distribution. When is (ε, δ)-differentially private learning possible if we wish to guarantee privacy with respect to changing *all* of any individual's samples at once? Ghazi, Kumar, and Manurangsi [GKM21] showed that this is possible when $n \geq O(\log(1/\delta)/\varepsilon)$ and the task admits a replicable learner. For the special case of PAC learning a concept class H, they argued that this implies a user-level private learning algorithm whenever H is privately PAC learnable with respect to changing a single *sample*. They posed the open problem of extending this result beyond PAC learning, e.g., to private regression [JKT20, Gol21]. Our conversion from any differentially private algorithm to a replicable one implies that such a transformation is possible for *any* statistical task with a finite outcome space (Section 5.6.1). Moreover, one can always take each indvidual's number of samples m to be nearly quadratic in the sample complexity of the original item-level private learner.

Amplifying differential privacy parameters.

While almost all (ε, δ)-differentially private algorithms enjoy a mild $\propto \log(1/\delta)$ dependence in their sample complexity on the parameter δ, it was not known how to achieve this universally, say by amplifying large values of δ to asymptotically smaller ones. [BLM20] showed that for private PAC learning, such amplification is possible in principle, but posed the open question of giving an explicit amplification algorithm. By converting an (ε, δ)-differentially private algorithm with weak parameters to a replicable one, and then back to a differentially private one with strong parameters, we resolve this question for the general class of statistical tasks with a finite outcome space, and with a much milder sample complexity blowup (Section 5.6.2).

Agnostic-to-realizable reductions for distribution-family learning.

[HKLM22] introduced a simple and flexible framework for converting realizable PAC learners to agnostic learners without relying on uniform convergence arguments. The framework applies to diverse settings such as robust learning, fair learning, partial learning, and (as observed in this work) replicable learning, with differential privacy providing a notable exception.[1] While an agnostic-to-realizable reduction for private PAC learning is known [BNS16b, ABMS20], it relies on uniform convergence and is only known to hold in the distribution-free PAC model. By converting a realizable private learner to a realizable replicable learner, then to an agnostic replicable learner, and back to an agnostic private learner, we obtain a reduction that works in the absence of uniform convergence (Section 5.6.3). In particular, this reduction applies to the *distribution-family* learning model, where one is promised that the marginal distribution on unlabeled examples comes from a pre-specified family of distributions.

Separating Stability: Computational Barriers and the Complexity of Correlated Sampling

All of the transformations appearing in Figure 5.1 preserve computational efficiency, with the lone exception of the transformation from perfectly generalizing algorithms to replicable ones. This transformation makes use of the technique of *correlated sampling* from the distribution of outputs of a perfectly generalizing algorithm A when run on a fixed sample S (elaborated on more in Sections 5.1.2 and 5.2.5). This step can be explicitly implemented via rejection sampling from the output space of A, with the rejection threshold determined by the probability mass function of $A(S)$, but in general it is not computationally efficient.

We show that under cryptographic assumptions, this is inherent (Section 5.4). Specifically, we show that under standard assumptions in public-key cryptography, there exists a statistical task that admits an efficient differentially private algorithm, but does not have any efficient replicable algorithm. The task is defined in terms of a public-key encryption scheme with the following rerandomizability property: Given a ciphertext $\mathsf{Enc}(\mathbf{pk}, b)$, there is an efficient algorithm producing a uniformly random encryption of b. Fixing such a rerandomizable PKE, the statistical task is as

[1]We note the technique we introduce to adapt [HKLM22] to the replicable setting has no clear translation to the private setting.

follows. Given a dataset consisting of random encryptions of the form $\mathsf{Enc}(\mathbf{pk}, b)$ where $\mathbf{pk}$ is a fixed public key and $b \in \{0,1\}$ is a fixed bit, output any encryption of b.

One can solve this problem differentially privately, essentially by choosing a random ciphertext from the input dataset and rerandomizing it. On the other hand, there is no efficient replicable algorithm for this task. If there were, then one could use the public key to produce many encryptions of 0 and 1 and run the replicable algorithm on the results to produce canonical ciphertexts c_0 and c_1, respectively. Then, given an unknown ciphertext, one could repeatedly rerandomize it, run the replicable algorithm on the results, and compare the answer to c_0 and to c_1 to identify the underlying plaintext.

We also show that cryptographic assumptions are necessary even to separate replicability from perfect generalization. Recalling again that the bottleneck in computationally equating the two notions is in implementing correlated sampling, we show in Section 5.4.2 that if one-way functions do not exist, then correlated sampling is always tractable. In addition to addressing a natural question about the complexity of correlated sampling, this shows that function inversion enables an efficient transformation from perfectly generalizing algorithms into replicable ones. (See Section 5.2.5 for more discussion.)

Separating Stability: Statistical Barriers

Our equivalences show that the sample complexities of perfectly generalizing and replicable learning are essentially equivalent. Moreover: (1) An approximate-DP algorithm can be converted to a perfectly generalizing/replicable algorithm with near-quadratic blowup; and (2) A perfectly generalizing/replicable algorithm can be converted to an approximate-DP one using roughly the same number of samples. We prove that both of these conversions are optimal by showing:

1. **Quadratic separations between differential privacy and (perfect generalization, replicability).** We first consider the problem of estimating the parameters of a product of d Bernoulli distributions. By simply taking the empirical mean of an input dataset, this problem

can be solved using $O(\log d)$ without any stability constraints. However, with differential privacy, it is known that $\tilde{\Theta}(\sqrt{d})$ samples are necessary and sufficient. By adapting the "fingerprinting" method underlying these privacy lower bounds [BUV18, DSS^{+}15, BSU19] to perfect generalization, we prove that any perfectly generalizing or replicable algorithm for this problem requires $\tilde{\Omega}(d)$ samples (Section 5.5.1).

By reducing from a variant of this one-way marginals problem, we also show a general lower bound for replicable agnostic learning. Namely, we show that every concept class H requires $\tilde{\Omega}(VC(H)^2)$ samples. For concept classes of maximal VC dimension $VC(H) = \log|H|$, this too gives a quadratic separation between replicable learning and both private and unconstrained learning (Section 5.5.2).

2. **No separation between differential privacy and (perfect generalization, replicability).** Complementing our lower bounds, we also show that every finite class H can be replicably PAC learned (in the realizable setting) to error α with sample complexity $\tilde{O}_H(1/\alpha)$ (Section 5.5.3). Up to logarithmic factors, this matches the learning rate achievable for both unconstrained and differentially private learning. Our learner works by selecting a random threshold v, and selecting a random concept from H whose error with respect to the sample is at most v. A more involved random thresholding strategy also yields an agnostic learner with sample complexity $\tilde{O}_H(1/\alpha^2)$.

1.6 Related Work

1.6.1 Prior Work

Our definition of replicability is inspired by the literature on pseudodeterministic algorithms [GG11, GGR13, GG17, GGH18, GGMW19, GL19, Gol19]. In particular, [GL19] and [Gol19] define reproducibility in the context of pseudodeterminism. In the pseudodeterministic setting, the primary concern is replicating the output of an algorithm given the same input, over different choices of the algorithm's internal randomness. There, the input of a reproducible algorithm is a fixed string. In our setting, the input of a replicable learning algorithm is a distribution, only accessible by randomly drawing samples.

Our work is related to other notions of stability in machine learning which, like our definition, are properties of learning algorithms. In the supervised learning setting, stability is a measure of how much the output of a learning algorithm changes when small changes are made to the input training set. An important body of work establishes strong connections between the stability of a learning algorithm and generalization [DW79b, DW79a, KR99, BE02, SSSSS10]. Distributional notions of stability which remain stable under composition and postprocessing, were defined and shown to be closely connected to differential privacy and adaptive data analysis (e.g., [BNS$^+$16a, DFH$^+$15a]). In fact, the definition of differential privacy itself is a form of stability known as max-KL stability. Stability-based principles have also been explored in the context of unsupervised learning where model selection is a difficult problem since there is no ground truth. For example, a stable algorithm for clustering has the property that when the algorithm is applied to different data sets from the same distribution, it will yield similar outputs (e.g., [vL10]).

In all of these settings, stability depends on how close the outputs are when the inputs are close; what varies is the particular measure of closeness in input and output space. For example, closeness in the output can be with respect to function or parameter space; for distributional stability close means that the output distributions are close with respect to some metric over distributions. Our definition of replicability can be viewed as an extreme form of stability where the output is required to be *identical* almost all of the time, and not just similar. Thus replicability enjoys many of the nice properties of stable algorithms (e.g., postprocessing, composition) but has the advantage of being far easier to verify.

Stability has a long history as a tool for ensuring generalization. Early work [RW78, DW79a, BE02, SSSSS10] showed that the stability of a learning algorithm with respect to a specific loss function could ensure strong generalization guarantees with respect to that loss. A more recent literature has focused on stability notions that are not tied to a specific loss, and which ideally are robust under post-processing and adaptive composition. This includes understanding the generalization guarantees of differential privacy [DFH$^+$15a, DFH$^+$15b, BNS$^+$21, RRST16, RRS$^+$20, ZH19, JLN$^+$20] and other constraints on the information-theoretic relationship between the input and output of a learning algorithm [RZ16, BMN$^+$18, XR17, RRT$^+$16, LS19, SZ20]. A related line of work [CLN$^+$16, BF16, NSS$^+$18] considers more "semantic" notions of stability, defining it in terms of the difficulty of inferring properties specific to the sample rather than of the underlying distribution. Perfect generalization, one of the main definitions we study in Chapter 5, was introduced by [CLN$^+$16] and is a special case of *typical stability* that was introduced in independent work of Bassily and Freund [BF16].

1.6.2 Concurrent Work

Independently of the work in Chapter 3, [GKM21] defines a property equivalent to replicability, called "pseudo-global stability". Their (α, β)-accurate (η', ν')-pseudo-global stability definition is equivalent to the (η, ν)-replicability definition discussed in Appendix 3.8, except that pseudo-global stability includes explicit parameters for correctness and sample complexity. In Appendix 3.8, we show that these two definitions are equivalent to Definition 3.1.1 up to polynomial factors. [GKM21] gives pseudo-globally stable SQ algorithms, an amplification of the stability parameter, and an algorithm to find a heavy-hitter of a distribution. The authors use pseudo-global stability to show that classes with finite Littlestone dimension can be learned user-levelly privately, and they connect pseudo-global stability to approximate differential privacy. Pseudo-global stability is a generalization of global stability, introduced in [BLM20]. Those authors use global stability as an intermediate step to show that classes with finite Littlestone dimension can be learned privately, and they show how global stability implies generalization.

Several elements of our approach in Chapter 5 were inspired by [GKM21]. Correlated sampling played a crucial role in their work by allowing individuals to use shared randomness to reach consensus on a learned hypothesis. In fact, it provided a key step in their conversion from "list globally stable" algorithms [GGKM21] (learning algorithms that output a short list of hypotheses, one of which is almost guaranteed to be canonical for the given distribution) to pseudo-globally stable ones.

Independent of the work in Chapter 5, [KKMV23] study similar relationships between notions of stability. They focus on the PAC-learning setting, where they show a statistical equivalence between differential privacy, replicability, and a notion called "TV-indistinguishability" which can be thought of as a special case of perfect generalization with $\varepsilon = 0$. Our approach gives us a constructive procedure for converting a private algorithm for a general statistical task into a replicable algorithm, so long as the private algorithm has finite range. Our transformations induce a modest sample complexity increase, resulting in a replicable algorithm with sample complexity n^2, given a private learner with sample complexity n. By contrast, the results of [KKMV23], while non-constructive, apply to countably infinite domains (and therefore to some uncountably infinite ranges). However, their results go through Littlestone dimension, which may be an exponential tower in n, and so they obtain sample complexity bounds which are an exponential tower in n as well. For more details, see Section 5.1.3.

1.6.3 Subsequent Work on Mathematical Replicability

Our research has inspired other researchers to design provably replicable algorithms and to extend mathematical replicability to new contexts.

The following is a brief, incomplete list of follow-up work that has been published or accepted for publication. These papers are grouped together for ease of exposition. However, these categories and short descriptions do not entirely capture the scope of each works' results.

Researchers have defined similar notions of replicability in other learning models. These include replicable reinforcement learning [KVYZ23, EHKS23] and replicable bandits [EKK$^+$23].

Researchers have developed the new replicable algorithms and theory for the i.i.d. setting. [EKM$^+$23] gives replicable clustering algorithms, [KKL$^+$24] improves the efficiency of halfspace learners in Section 3.5 and Section 3.6, and [HIK$^+$24] connects replicable algorithms to isoperimetric tilings.

Researchers have defined relaxed and stricter notions of replicability, further connecting stability to the learning theory landscape. [DPVWV23, CMY23, CCMY24] explore list replicability, a variant of replicability where an algorithms output are required to belong to a short list of canonical outputs, rather than a single output per random string. In fact, [CMY23] shows an equivalence between list replicability and global stability, introduced in [BLM20] (and a precursor to pseudo-global stability, introduced in [GKM21] and equivalent to replicability).

[DPVWV23] defines certificate replicability, where the canonical hypothesis an algorithm must return depends on the first l bits of the random string r. [KKVZ24] explores computational relationships between replicability, privacy, and online learning. [MSS23] summarizes relationships among "Bayesian definitions of stability" — these include replicability, differential privacy, finite clique dimension, mutual information stability, and distribution-dependent KL-stability.

1.7 Future Directions for Replicability Research

Why should we study replicability mathematically?

Replicability can be used as a tool to prove theorems outside of replicability theory. In fact, the first applications of replicability (according to Definition 2.1.1) were for proving a Massart boosting lower bound (Section 4.8) and an item-level to user-level reduction for differential privacy [GKM21]. Section 5.6 contains additional applications of replicability.

Replicability itself may be a desired property. Our definition of replicability may not be suitable for a specific application, but it may be helpful in finding a definition that is. For example, this work on replicability has led to theory for replicability variants (Section 1.6.3). The new context necessitates new algorithms and new techniques for algorithm designers to examine.

Broadly, theory can be used to contextualize individual applications and problems. Theory can guide algorithm designers towards successful paradigms (upper bounds) and away from impossible situations (lower bounds). Theory can help suggest new heuristics and verify that heuristics are not captured under known lower bounds.

Even if replicability is not explicitly desired, replicability theory may be useful because replicability implies other desirable properties. Section 3.1.1 summarizes how replicable algorithms can be amplified, adaptively compose, and generalize. Section 5.3 describes how replicable algorithms can be transformed to have other algorithmic stability properties, such as differential privacy. So, an algorithm designer looking for such properties can examine replicable algorithms as a starting point for designing their own algorithm.

Similarly, replicability theory has connections across mathematics. In this dissertation, we see connections between replicability and

- cryptography: one-way functions (Sections 4.8, 5.4),
- geometry: randomized rounding and tilings (Sections 3.5.2, 3.5.3, 5.7),
- learning theory: statistical queries, halfspaces, boosting, agnostic learning, (Sections 3.2, 3.5, 3.6, 5.5.2), and
- algorithmic stability: differential privacy, perfect generalization, and max information (Chapter 5).

As discussed in Section 1.6.3, subsequent work has expanded these connections. For example, [CMY23] and [CCMY24] connect replicability to topology by applications of the Poincaré-Miranda theorem and the Borsuk-Ulam theorem!

Mathematical academic research in universities is fractured. Replicability research is a possible avenue for reconvening researchers from segregated areas of mathematics.

As an example, numerical stability may have connections with replicability research. Numerical stability is a subarea of mathematics concerning the propagation of errors in numerical algorithms. For example, rounding errors in an algorithm's inputs may be magnified, depending on the design of the algorithm. How can the replicability theory established so far influence numerically stable algorithms, and vice versa?

Coordination games could be another new connection. Replicable algorithms given canonical outputs, even when there may be many acceptable outputs. In coordination games, multiple players want to make optimal individual decisions with minimal communication (e.g., crossing a traffic light). Replicable algorithms could be possible subroutines for players to reach equilibria.

We conclude this chapter by discussing future directions. These directions are presented in sections, divided by context. However, all contexts share the following themes: deciding on an appropriate notion of replicability, designing replicable algorithms, and pursuing their applications. For additional open questions, see the open question sections of papers listed in Section 1.6.3.

1.7.1 Replicability in This Dissertation's Model

First, we discuss theoretical directions for the model and definition of replicability introduced in this dissertation (Definition 2.1.1).

New Replicable Algorithms

As replicability is defined as a property of algorithms, any algorithmic learning problem using i.i.d. data can be made to also ask about replicability. By comparing the best possible non-replicable algorithm for a problem with the best replicable algorithm, one can quantify the parameter tradeoffs (accuracy, efficiency) in making algorithms ρ-replicable.

For algorithms and algorithmic paradigms that operate on samples, without any distributional assumptions, one can ask if these algorithms are replicable when their data is sampled from an arbitrary distribution.

For example: replicable parity learning, replicable principle component analysis (PCA), replicable stochastic gradient descent, replicable dimensionality reduction, replicable regression, replicable naive Bayes algorithms, replicable random forest algorithms, etc.

Adaptive Composition — Combining Replicable Algorithms

What is the best way to run a series of replicable algorithms? Is there any advantage to running algorithms together, rather than just separately?

In Chapter 3, we show that replicable algorithms compose adaptively. That is, a sequence of k adaptively chosen ρ-replicable algorithms yields a transcript that is $O(k\rho)$-replicable. One way to interpret this result is as follows: Given a sequence of k analyses that are each (0.01)-replicable using a sample of size n, one can amplify their individual replicability parameters to $O(1/k)$ at the expense of increasing their sample complexity to $O(k^2 n)$. This yields a (0.01)-replicable algorithm for performing all k analyses at a sample cost of $O(k^2 n)$.

Our conversions between replicability and differential privacy yield a different tradeoff, at least for simulating non-adaptive composition. Given k analyses that are each (0.01)-replicable using a sample of size n, one can convert them to $\tilde{O}(1/\sqrt{k})$-differentially private algorithms each using a sample of size $\tilde{O}(\sqrt{k}n)$. "Advanced" composition of differential privacy [DRV10] yields an $(0.01, \delta)$-differentially private algorithm using $\tilde{O}(\sqrt{k}n)$ samples, which can then be turned back into a (0.01)-replicable algorithm using $\tilde{O}(kn^2)$ samples.

What is the optimal sample cost for conducting, or at least statistically simulating, the adaptive composition of k replicable algorithms? Is it possible to do so at a cost of $O(kn)$ samples?

Transformations between Replicability and Differential Privacy

The main result of Chapter 5 is a general equivalence between replicable algorithms and differentially private algorithms. While an individual algorithm may not satisfy both notions simultaneously, we show how to transform replicable algorithms to DP-algorithms and vice versa.

Furthermore, we show that these black-box reductions are tight (i.e. tight when we do not make additional assumptions about the problem).

Are there natural problems for which there are larger separations than those suggested by our reductions? For example, in Section 5.5.3, we give a direct replicable algorithm for the task of realizable PAC learning of finite classes with sample cost inverse linear in the accuracy parameter α. If we directly apply our reduction from replicability to approximate DP, we get inverse quadratic. See Theorem 5.6.13 and the following discussion for more information.

For which problems does our replicability to approximate DP reudction give tight bounds? Can we prove tight bounds for other problems?

Turning to computational efficiency, what are the minimal cryptographic assumptions under which a computational separation between replicability and differential privacy exists? Our results in Section 5.4 show that one-way functions are necessary, while public-key assumptions are sufficient.

Correlated Sampling over Infinite Domains

While correlated sampling introduces no sample complexity overhead in terms of the output space, it is only known to be possible when the output space is finite or the class of distributions to be sampled from is structured. Examples of such structures include when the distributions in the class all have uniformly bounded Radon-Nikodym derivative with respect to some fixed base measure. Formally, this case falls into a restricted notion of correlated sampling over a subset of distributions, similar to the multiple coupling of [AS19].

One application is in reinforcing the connection between perfect generalization and replicability. Is a transformation from one-way perfectly generalizing algorithms to replicable algorithms possible for infinite output spaces in general?

In independent work, [KKMV23] make progress towards this goal by giving a transformation from TV-indistinguishability to replicability when there are only countably many options for the TV-indistinguishable algorithm $\{A(S)\}_{S \in X^n}$. It follows from Lemma 5.3.8 that $(\beta, \varepsilon, \delta)$-one-way perfect generalization implies $(4\varepsilon+2\delta+2\beta)$-TV indistinguishability, and so the result of [KKMV23] gives the following corollary.

Corollary 1.7.1. *Fix $n \in \mathbb{N}$, $\beta, \varepsilon, \delta \in (0,1]$. Let $\mathscr{X}$ be a countable domain and $A : \mathscr{X}^n \to \mathscr{Y}$ be a $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing algorithm for a statistical task. Then there exists an algorithm $A' : \mathscr{X}^n \to \mathscr{Y}$ that is $\left(\frac{2\rho}{1+\rho}\right)$-replicable for $\rho = 4\varepsilon + 2\delta + 2\beta$, and for all $S \in \mathscr{X}^n$, $A(S) = A'(S)$.*

The existence of an analogous transformation for general measure spaces remains an open problem. In the PAC-setting, one can resolve this issue via factoring through Littlestone Dimension and replicable heavy-hitters (Section 3.3), but this results in tower sample complexity. In Section 5.6.3 we discuss the *list heavy-hitters* problem, that may be a candidate for separating perfect generalization from replicability over infinite output spaces.

Replicable Smooth Boosting and Connections

In Section 3.6, we present the first replicable smooth boosting algorithm. We do this by modifying the smooth boosting algorithm in [Ser03]. Smooth boosting has been shown to have deep connections with complexity theory, combinatorics, and number theory via hard-core lemmas, dense-model theorems, and regularity lemmas [KS99, GT08, RTTV08, Gow10, TTV09]. What implications do replicable smooth boosting algorithms have for these intimately related mathematical objects?

1.7.2 New Models for Replicability

As discussed in Section 1.2, our definition of replicability only applies to algorithms that use samples drawn i.i.d. from some distribution. This may not be a sensible assumption. For example, datasets may evolve over time, or the algorithm may have interactive access to portions of the data.

To be clear, the scientific term "replicability" is not identical to our mathematical term "replicability" (Definition 2.1.1).

In 2019, the National Academies defined replicability as follows: “Replicability is obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data” [Nat19].

In 2020, the Association for Computing Machinery (ACM) defined replicability as follows: “The measurement can be obtained with stated precision by a different team, a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently” [Ass20].

These definitions can be summarized as “different experimental setup, same results”.

Our mathematical definition (Definition 2.1.1) makes explicit choices. The different data/artifacts are different samples S_1, S_2, while the same results are exactly identical outputs to the underlying computational problem. But these choices represent only one type of replicability.

For example, “studies aimed at answering the same scientific question” could refer to different methods of gathering samples or different algorithmic approaches entirely. Similarly, independent “artifacts” could refer to the algorithms run themselves, not just independent samples. Whereas, in our definition, the sampling method is the same (i.i.d. samples for both experiments) and the algorithm run is the same (fixing the randomness r makes both experiments run the same deterministic algorithm).

If we want a broad mathematical theory of replicability, we must investigate replicability in different models. This involves defining a sensible notion of replicability, generalizing previous techniques for designing replicable algorithms, and analyzing limitations of replicability.

Models without I.I.D. Samples

As defined in Definition 2.1.1, replicable algorithms interact with the underlying data source through a fixed sample. However, there are many other models of learning from data. An algorithm could interact by querying a data source, receiving data insertions and deletions dynamically, get rewards from its environment, handle corrupted data, build a data structure, etc. Our definition of replicability may not naturally generalize to these new settings; yet, some form of replicability may be desired.

Shared randomness is key to the definition of replicability introduced in this dissertation. It acts as advice that two groups can use to correlate the results of their algorithms. In this sense, shared randomness may be useful for achieving replicability in new algorithmic models. For example, see the replicability models in the context of bandits [EKK$^+$23] and reinforcement learning [KVYZ23, EHKS23].

Models with I.I.D. Samples

Even without changing the i.i.d. sample assumption, one can create new, mathematically interesting models for replicability. For example, [DPVWV23, CMY23, CCMY24, HIK$^+$24] already have defined certificate replicability, list replicability, and other related notions.

One immediate motivation for such models is to overcome limitations of replicable algorithms in the existing i.i.d. model. In Section 3.7, we proved a lower bound showing a $(1/\rho^2)$-factor increase in the sample complexity of computing statistical queries replicably. Roughly, this implies that guaranteeing 99% replicability requires a 10,000-fold increase in the sample size, which may be prohibitive for applications. While replicability can be tested on known distributions, there are adversarial situations in which it is computationally infeasible to determine an algorithm's replicability on other (arbitrary) distributions.

Refocusing on approximate replicability notions is one way to circumvent these and other limitations. Investigating these notions will lead to new techniques for analyzing and designing replicable algorithms.

As defined in Definition 2.1.1, algorithms are replicable if their outputs are exactly the same. We could relax this equality condition to outputs that *behave* the same. For example, consider algorithms that output functions. An algorithm is *conduct replicable* on domain $\mathscr{X}$ if the functions output across two runs $f_1 \leftarrow \mathscr{A}(S_1;r)$ and $f_2 \leftarrow \mathscr{A}(S_2;r)$ are consistent on all points $x \in \mathscr{X}$. In other words, $f_1(x) = f_2(x) \forall x \in \mathscr{X}$. Functions f_1 and f_2 need not have the same representation.

One motivation of this definition is that many machine learning models are complex and difficult to precisely write down. Checking exact representation equality may be time-consuming and not as important as checking behavioral equality.

Say we devise a conduct replicable algorithm for some problem. Can one efficiently test the outputs of the algorithm for consistency on all $x \in \mathscr{X}$? Even if the replicable algorithm is efficient, it may produce functions that cannot be efficiently checked by brute force. The complexity of testing replicability in this model could offer connections to computational indistinguishability and meta-complexity, the study of determining complexities of problems about complexities.

Another way to relax the exact equality condition of Definition 2.1.1 is the following. Again, consider algorithms that output functions. Rather than guarantee that output functions are the same with high probability, we guarantee that output functions are the same on any individual point $x \in \mathscr{X}$ with high probability. Formally, $\mathscr{A}$ is ρ*-per-point* replicable if: $\forall D, \forall x \in \mathscr{X}$: $\mathbf{Pr}_{S,S' \sim D^n, r \sim R}[A(S;r)(x) = A(S';r)(x)] \geq 1 - \rho$. This notion is weaker than Definition 2.1.1, and it may be easier to achieve.

One possible application is constructing replicable PAC-learners. We do not know if replicability can always be achieved for PAC-learning problems. Rather than trying to build replicable PAC-learning algorithms directly from nonreplicable PAC learners, we could use approximate replicability notions as stepping stones in between. The techniques we used for amplification and composition of replicability do not directly work for per-point replicability. So, this investigation will yield new ways to design and analyze replicable algorithms.

There are many other sensible ways to define approximate replicability notions. For example, one can allow perturbations to the input distribution, restrict the algorithm's randomness usage (e.g., [DPVWV23]), or allow algorithms to produce lists of outputs (e.g., [CMY23, DPVWV23]). One could also consider restricting replicability to a set of known distributions, leveraging properties of distributions specific to that set (e.g., the Distribution-Family Model discussed in Section 5.6.3).

Characterizing the relationships between these notions will provide a more fine-grained understanding of the scope of replicable algorithms. A larger repertoire of replicable algorithms will help researchers incorporate replicability into new contexts.

Distributional Shift

To what extent is replicability preserved under distributional shift? One way the i.i.d. sample data assumption can be violated is if the underlying distribution D changes between sampling sample S_1 and sample S_2. For example, a city may see residents come and go over time, changing the city's population. Does replicability give accuracy guarantees, if the distributions are close?

In Appendix 5.9, we give a trivial argument showing that a ρ-replicable algorithm is $\rho(1-\delta)^{2m}$-replicable across two close distributions. Are there tighter replicability and non-replicability bounds for specific families of distributions, problems, and algorithms under distributional shifts? In other words, is there an algorithmic model that cleanly connects replicability to metrics between distributions?

Properties Implied by Replicability

Once a mathematical model of replicability is formalized, one can ask if replicability automatically implies other desirable properties for algorithms.

- Testability: Given an algorithm, can we efficiently test that it is replicable? There are many subtleties to consider. For example, how is the algorithm given to us? Do we get black-box access to the algorithm itself? Or maybe a description of the algorithm? Perhaps a definition of replicable algorithm could require the algorithm come with a more efficient prover algorithm that provides an interactive proof. It could also be interesting if there are settings in which replicability testing is hard in the worst case but easy "on average".

- Amplification: When can the replicability of an algorithm be improved? For our i.i.d.-setting notion of replicability, the replicability parameter ρ can be amplified in a black-box way. Therefore, users of replicable algorithms can easily determine what sample sizes are needed for their desired replicability. Is this true for other notions of replicability, or are there specific characteristics of different notions of replicability that lend themselves better to amplification? For example, do approximate replicability notions always admit efficient black-box amplification? If not, why not?

- Composition and adaptivity: Are there benefits to running multiple replicable algorithms at once? In the i.i.d. setting, replicability composes almost by definition with a linear (union-bound-style) increase in the replicability loss. However, our work connecting replicability to differential privacy directly implies a not-directly-comparable replicability loss. What is the true replicability loss when running multiple replicable algorithms, perhaps one after another? Are there situations in which we can reuse computational resources to save overall? Which replicability definitions lend themselves better to these savings?

1.7.3 Applying Replicability Theory Beyond Mathematics

At the beginning of this section, we discussed connections between replicability theory and other areas of mathematics. Next, we discuss the potential for applications beyond mathematics.

First, a clarification: theoretical work is often inspired by practical concerns, but the methods of evaluating theoretical work are different than those of evaluating non-mathematical work. Doing theory for theory's sake is valuable. In principle, theoretical research can be influenced by practical concerns, but there is no mathematical reason why theory should be restricted to practical concerns.

However, there is room for translation error when extrapolating a theoretical model outside of its original setting. Even if a mathematical model is inspired by a specific application, that model may still be a bad starting point for applications. For example, the mathematical notion of replicability could completely ignore the true sources of non-replicability. How should the responsibility of determining how theory should influence practice be divided? How can we all (theorists, algorithm designers, algorithm implementers, regulatory bodies, and those affected by algorithmic decisions) make this responsibility easier to manage?

Replicable Algorithms for Statistical Tests

Statistical tests, such as linear regression, Student's t-test, analysis of variance (ANOVA), principal component analysis, and chi-squared test, have seen wide application in experimental sciences. Creating versions of these tests with additional replicability guarantees is an interesting theoretical question and deserves investigation.

Practical application of replicable statistical tests raises additional concerns. Which groups have a stake in the implementation and outcome of these tests? By what mechanism are these groups' opinions combined to produce an algorithm? If these mechanisms are lacking, how can we improve them?

Even if all groups agree on using a replicable algorithm, there are still important decisions that need to be made. What does replicability mean? Are the tradeoffs of replicability clearly communicated and understood? Can a quantity ρ for replicability be agreed upon by all parties in a reasonable and fair way?

Focusing on theorists: what role should a theorist play after designing an algorithm that inherently includes value judgements (e.g., "What parameter of ρ is acceptable?")? Is warning about possible pitfalls in the original publication enough, or should the theorist interactively engage with future users of the algorithm?

Replicability and Differential Privacy

Differential privacy is a notion of algorithmic stability that has seen both theoretical and practical applications. Perhaps most notably, differential privacy was incorporated into the release of data collected for the 2020 United States Census. Differential privacy has been incorporated not only in various theoretical data analysis algorithms, but also implemented in software libraries (see, e.g., the Theory and Practice of Differential Privacy Workshop Series).

In Chapter 5, we prove black-box algorithmic reductions between replicability and differential privacy. These reductions may not directly yield sample-efficient or computationally efficient replicable algorithms. Nevertheless, the reductions suggest that many previously implemented differentially private algorithms have replicable counterparts.

Using a differentially private algorithm requires a value judgement for the implemented (ε, δ) parameters. Similarly, scientists make a collective value judgement when agreeing to fix a threshold of .05 for signifiance of p-values. In the absence of infinite samples and computational power, concessions to the privacy parameters must be made to preserve efficiency and accuracy. Moreover, using differentially private algorithms requires value judgements regarding the incorporation of privacy itself. Namely, is differential privacy (and all its associated, implied properties) appropriate for the algorithmic context? Of course, replicability shares these issues. Is replicability, as defined mathematically, appropriate for the algorithmic context? How can this determination lead to more appropriate solutions?

In the previous discussion of replicable statistical tests, we posed a question of agency — how do the groups involved in implementing stable algorithms and those affected by their outputs collectively decide on an implementation? There is also a question of progress — does implementing an algorithm in a specific way preclude better algorithms and better implementations? Or, can implementing a suboptimal algorithm be constructively used towards improving future implementations? These questions, in the implementation process, are necessarily answered.

To be clear: new research necessitates mistakes. These arguments are not meant to suggest that specific research is invalid or impossible to do. Rather, these arguments are meant as an encouragement to examine our own behavior and reevaluate possible improvements.

How “Should” Theory and Practice Collaborate?

How can the theory of replicable algorithms inform reliable data science?

The theoretical study of replicability provides algorithmic tricks such as randomized rounding, correlated sampling, and algorithmic reductions. Knowledge of these tools can expedite the design of new replicable algorithms. Furthermore, one does not necessarily need to use theoretically proven replicable algorithms to benefit. Replicability theory may be used as a starting point for developing heuristics for algorithmic stability. Lower bounds for replicable algorithms can suggest new, efficient approaches.

However, these facts cannot be helpful if they are not clearly conveyed. How should decisions about algorithmic choices be recorded and disseminated, so that algorithm implementers can more easily identify the validity of their approach? Maybe these facts are clearly conveyed, but they are not applicable because of discrepancies between the model and the application. Early collaboration may help avoid these pitfalls. What incentives are there for different groups to engage in such collaboration? Are these incentives appropriate, and should they be adjusted by organizations (e.g., universities, conferences, funding agencies, and governments)?

Replicability is a concern shared among all researchers, including those in mathematics. Is there value in future interdisciplinary collaboration regarding replicability? Are the current mechanisms for creating these collaborations sufficient? What replicability research do we, as a society, want?

Acknowledgements

Chapter 1, in part, is a reprint of the material as it appeared in three publications. The first, on Arxiv at https://arxiv.org/abs/2201.08430v2, which is the full version of the paper that appeared in the Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing 2022. Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sorrell, Jessica. “Reproducibility in

Learning". The second, in the Proceedings of the 34th Annual Conference on Learning Theory 2022. Diakonikolas, Ilias; Impagliazzo, Russell; Kane, Daniel M.; Lei, Rex; Sorrell, Jessica; Tzamos, Christos. "Boosting in the Presence of Massart Noise". The third, on Arxiv at https://arxiv.org/abs/2303.12921v2, which is the full version of the paper that appeared in the Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing 2023. Bun, Mark; Gaboardi, Marco; Hopkins, Max; Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sivakumar, Satchit; Sorrell, Jessica. "Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization". The dissertation author was the primary investigator and author of these papers.

Chapter 2

Preliminaries

In this section, we briefly list relevant definitions for this dissertation.

2.1 Replicability

The following is the main definition in this dissertation. For a detailed discussion of this definition, see Section 1.2.

Definition 2.1.1 (Replicability)**.** *Let D be a distribution over domain $\mathscr{X}$. Let $\mathscr{A}$ be a randomized algorithm that takes as input samples from D. We say that $\mathscr{A}$ is ρ-replicable if*

$$\mathbf{Pr}_{S_1,S_2,r}[\mathscr{A}(S_1;r) = \mathscr{A}(S_2;r)] \geq 1-\rho,$$

where S_1, S_2 are sets of samples drawn i.i.d. from D and r represents the internal randomness of $\mathscr{A}$.

One could also parametrize replicability with two parameters (η, ν), instead of just ρ.

Definition 2.1.2 ((η,ν)-Replicability)**.** *Let $A(S;r)$ be an algorithm operating on a sample set $S \in \mathscr{X}^n$ and internal coins r. We say that coin tosses r are η-good for A on distribution D if there exists a "canonical output" z_r such that $\mathbf{Pr}_{S\sim D^n}[A(S;r) = z_r] \geq 1-\eta$. We say that A is (η,ν)-replicable if, for every distribution D, with probability at least $1-\nu$, the coin tosses r are η-good on distribution D.*

Two-parameter replicability is qualitatively the same as ρ-replicability, but might differ by polynomial factors. For every $0 \leq \rho \leq v \leq 1$,

1. Every ρ-replicable algorithm is also $(\rho/v, v)$-replicable.

2. Every (ρ, ν)-replicable algorithm is also $\rho + 2\nu$-replicable.

See Section 3.8 for a short proof.

Given an replicable algorithm $\mathscr{A}$, one can create a new replicable algorithm $\mathscr{A}'$ with better replicability but worse sample complexity.

Theorem 2.1.3 (Amplification of Replicability)**.** *Let $0 < \eta, \nu, \beta < 1/2$ and $m > 0$. Let $\mathscr{A}$ be an (η, ν)-replicable algorithm for distribution D with sample complexity m and failure rate β. If $\rho > 0$ and $\nu + \rho < 3/4$, then there is a ρ-replicable algorithm $\mathscr{A}'$ for D with sample complexity $m' = \widetilde{O}(m(\log 1/\beta)^3/(\rho^2(1/2 - \eta)^2)$ and failure rate at most $O(\beta + \rho)$. The construction of $\mathscr{A}'$ does not depend on D.*

2.2 Algorithmic Stability Notions

In this section, we list other definitions of algorithmic stability. This section is **not** a comprehensive list of algorithmic stability notions, nor is it a comprehensive list of algorithmic stability notions that are mathematically related to replicability.

In Section 3.8, we show an equivalence between replicability and pseduo-global stability. In Section 5.3, we show equivalences between replicability, approximate differential privacy, β-approximate max information, and one-way perfect generalization.

2.2.1 Pseudo-Global Stability

Independently of the work in this dissertation, [GKM21] introduced a definition called pseudo-global stability.

Definition 2.2.1 (Pseudo-global stability, Definition 15 in [GKM21]). *A learning algorithm $\mathscr{A}$ with sample complexity m is said to be (α,β)-accurate, (η',ν')-pseudo-globally stable if there exists a hypothesis h_r for every $r \in supp(R)$ (depending on D) such that $\mathbf{Pr}_{r\sim R}[\mathrm{err}_D(h_r) \leq \alpha] \geq 1-\beta$ and*

$$\mathbf{Pr}_{r\sim R}\left[\mathbf{Pr}_{\vec{s}\sim D^m}[\mathscr{A}(\vec{s};r) = h_r] \geq \eta'\right] \geq \nu'$$

where $\vec{s}$ is a sample of m (labeled) examples (x_i, y_i) drawn from distribution D.

Pseudo-global stability is very similar to two-parameter replicability. The final condition of the pseudo-global-stability definition is equivalent to saying that i) a randomly chosen r is η'-good with probability at least ν', and ii) for every r, h_r is the output that witnesses the η-goodness.

2.2.2 Differential Privacy

Total variation distance is commonly used to quantify how different two distributions are.

Definition 2.2.2 (Total Variation Distance). *Let P and Q be probability distributions over some domain S. Then*

$$d_{\mathrm{TV}}(P,Q) := \sup_{E\subseteq S} |\mathbf{Pr}_P[E] - \mathbf{Pr}_Q[E]|.$$

For differential privacy, (ε,δ)-indistinguishability is used to quantify how different two distributions are.

Definition 2.2.3 ((ε,δ)-indistinguishability). *Let P and Q be probability distributions over some domain $\mathscr{Y}$. Then, we say that P is (ε,δ)-indistinguishable from Q (denoted as $P \approx_{\varepsilon,\delta} Q$) if for all $O \subseteq \mathscr{Y}$,*

$$e^{-\varepsilon}[\mathbf{Pr}_P[O] - \delta] \leq \mathbf{Pr}_Q[O] \leq e^{\varepsilon}\mathbf{Pr}_P[O] + \delta.$$

For notational convenience, we say random variables are (ε,δ)-indistinguishable to mean that their distributions are (ε,δ)-indistinguishable.

Two datasets $S, S' \in \mathscr{X}^n$ are *neighboring* if the difference in datasets is exactly one entry. An algorithm is differentially private if its distributions of outputs are indistinguishable, when run on neighboring datasets.

Definition 2.2.4 (Differential Privacy [DMNS16]). *A randomized algorithm $\mathscr{A} : \mathscr{X}^n \to \mathscr{Y}$ is said to be* (ε, δ)-differentially private *if for every pair of neighboring datasets $S, S' \in \mathscr{X}^n$, we have that for all subsets $O \subseteq \mathscr{Y}$,*

$$\mathbf{Pr}[\mathscr{A}(S) \in O] \leq e^{\varepsilon} \cdot \mathbf{Pr}[\mathscr{A}(S') \in O] + \delta.$$

That is, we have $\mathscr{A}(S) \approx_{\varepsilon,\delta} \mathscr{A}(S')$ for all neighboring S, S'.

Differential privacy is closed under post-processing by arbitrary functions.

Lemma 2.2.5 (Post-Processing [DMNS16]). *If $\mathscr{A} : \mathscr{X}^n \to \mathscr{Y}$ is (ε, δ)-differentially private, and $\mathscr{B} : \mathscr{Y} \to \mathscr{Z}$ is any randomized function, then the algorithm $\mathscr{B} \circ \mathscr{A}$ is (ε, δ)-differentially private.*

The sensativity of a function measures how different a function's outputs can be on neighboring datasets.

Definition 2.2.6 (ℓ_1-Sensitivity). *Let $f : \mathscr{X}^n \to \mathbb{R}^d$ be a function. Its ℓ_1-sensitivity is*

$$\Delta_f = \max_{\substack{S,S' \in \mathscr{X}^n \\ S,S' neighbors}} \|f(S) - f(S')\|_1.$$

The exponential mechanism is one technique used to make differentially private algorithms.

Lemma 2.2.7 (Exponential Mechanism [MT07]). *Let L be a set of outputs and $g : L \times \mathscr{X}^n \to \mathbb{R}$ be a function that measures the quality of each output on a dataset. Assume that for every $m \in L$, the function $g(m, .)$ has ℓ_1-sensitivity at most Δ. Then, for all $\varepsilon > 0$, there exists an $(\varepsilon, 0)$-DP mechanism that, on input $S \in \mathscr{X}^n$, outputs an element $m \in L$ such that, for all $a > 0$, we have*

$$\mathbf{Pr}\left[\max_{i \in [L]} g(i, S) - g(m, S) \geq 2\Delta \frac{\ln|L| + a}{\varepsilon}\right] \leq e^{-a}.$$

Standard (ε, δ)-differential privacy implies privacy for groups of individuals.

Lemma 2.2.8 (Group Privacy [DMNS16]). *Let $k \in \mathbb{N}^+$ and let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an (ε, δ)-DP algorithm. Then for all datasets $S, S' \in \mathscr{X}^m$ such that $\|S - S'\|_0 \leq k$,*

$$\mathscr{A}(S) \approx_{k\varepsilon, \delta \frac{e^{k\varepsilon}-1}{e^{\varepsilon}-1}} \mathscr{A}(S').$$

Privacy parameters are amplified when a differentially private algorithm is run on a subsample of a dataset.

Lemma 2.2.9 (Secrecy of the sample [KLN+11, BBG18]). *Let algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ be (ε, δ)-differentially private. Consider the algorithm $A' : \mathscr{X}^m \to \mathscr{Y}$ that, given a dataset of size m, randomly samples n items without replacement and runs A on the resulting subsample. Then A' is (ε', δ')-differentially private for*

$$\varepsilon' = \frac{n}{m}(e^{\varepsilon} - 1), \qquad \delta' = \frac{n}{m} \cdot \delta.$$

2.2.3 Perfect Generalization

An algorithm satisfies perfect generalization if its outputs on i.i.d. sample data S can be simulated using just the underlying distribution.

Definition 2.2.10 (Perfect Generalization [CLN+16, BF16]). *Algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ is $(\beta, \varepsilon, \delta)$-perfectly generalizing if, for every distribution D over $\mathscr{X}$, there exists a distribution Sim_D such that, with probability at least $1 - \beta$ over S consisting of n i.i.d. samples from D, and every set of outcomes $\mathscr{O} \subseteq \mathscr{Y}$,*

$$e^{-\varepsilon}(\mathbf{Pr}_{Sim_D}[\mathscr{O}] - \delta) \leq \mathbf{Pr}[A(S) \in \mathscr{O}] \leq e^{\varepsilon}\mathbf{Pr}_{Sim_D}[\mathscr{O}] + \delta. \tag{2.1}$$

Perfect generalization has many versions, including sample perfect generalization and one-way perfect generalization. See Section 5.3.2 for more details about the variations of perfect generalization and their relationships.

2.2.4 Max Information

Max information quantifies the information revealed about an algorithm's training sample.

Definition 2.2.11 (Max Information [DFH+15a])**.** *An algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ has (ε,δ)-max-information with respect to product distributions if for every set of outcomes $\mathscr{O} \subseteq (\mathscr{Y} \times \mathscr{X}^n)$ we have*

$$\mathbf{Pr}[(A(S),S) \in \mathscr{O}] \le e^{\varepsilon}\mathbf{Pr}[(A(S),S') \in \mathscr{O}] + \delta,$$

where S and S' are independent samples of size n drawn i.i.d. from an arbitrary distribution D over $\mathscr{X}$.

The dissertation uses the following two-sided version of max information.

Definition 2.2.12 (β-Approximate Max Information, based on [DFH+15a])**.** *The β-approximate max information between two correlated random variables X and Z, denoted $I_\infty^\beta(X,Z)$, is defined as the minimum (infimum) value k such that for all output sets O, we have that*

$$\mathbf{Pr}_{(a,b)\sim(X,Z)}[(a,b) \in O] \le 2^k \mathbf{Pr}_{(a,b)\sim X\otimes Z}[(a,b) \in O] + \beta \tag{2.2}$$

where $X \otimes Z$ represents the product measure of the 2 random variables.

See Section 5.3.2 for more details about the relationships among differential privacy, bounded max information, and perfect generalization.

2.3 Statistical Tasks, PAC Learning, and Noise

Solving problems replicably is a central issue of this dissertation. Next, we list some settings in which we consider designing replicable algorithms.

Definition 2.3.1. *A* statistical task *with data domain $\mathscr{X}$ and output space $\mathscr{Y}$ is a set of pairs $\mathscr{T} = \{(D, G_D)\}$, where D is a distribution over $\mathscr{X}$ and $G_D \subseteq \mathscr{Y}$ is a "good" set of outputs for distribution D. A randomized algorithm $\mathscr{A}$ solves statistical task $\mathscr{T}$ using m samples and with failure probability β if for every $(D, G_D) \in \mathscr{T}$,*

$$\mathbf{Pr}_{S \sim D^m, \mathscr{A}}[\mathscr{A}(S) \in G_D] \geq 1 - \beta.$$

In Chapter 5, we give broad equivalences for statistical tasks. The definition of a statistical task generalizes many well-studied problems and models, including PAC learning.

2.3.1 PAC Learning

Definition 2.3.2 (Realizable PAC learning, [Val84, VC74])**.** *A learning problem is defined by a hypothesis class H. For any distribution D over the input space $\mathscr{X}$, consider m independent draws $x_1, x_2, \ldots, x_m$ from distribution P. For $f \in H$, a labeled sample of size m is the set $\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_m, f(x_m))\}$. We say an algorithm A is an (α, β)-accurate PAC learner for the hypothesis class H if for all functions $f \in H$ and for all distributions D over the input space, A on being given a labeled sample of size m drawn from D and labeled by f, outputs a hypothesis h such that with probability greater than or equal to $1 - \beta$ over the randomness of the sample and the algorithm,*

$$\mathbf{Pr}_{x \in D}[h(x) \neq f(x)] \leq \alpha.$$

Uniform convergence is a powerful tool in PAC learning.

Theorem 2.3.3 (Uniform Convergence, e.g., [BEHW89])**.** *Let H be a binary class of functions with domain $\mathscr{X}$. Let its VC dimension be d. Then, for any distribution D over $\mathscr{X}$, for all $m > 0$,*

$$\mathbf{Pr}_{x_1, \ldots, x_m \sim D}\left[\sup_{h_z \in H}\left|\frac{1}{m}\sum_{i=1}^{m} \mathbb{1}[h_z(x_i) = 1] - \mathbf{Pr}_{x \sim D}[h_z(x) = 1]\right| \geq \gamma\right] \leq 4(2m)^d e^{-\gamma^2 m/8}.$$

2.3.2 PAC Learning with Agnostic Noise

Agnostic PAC learning is one PAC learning variant that considers corrupted input data.

Definition 2.3.4 (Agnostic PAC learning, [Hau92, VC74]). *A learning problem is defined by a hypothesis class H. We say an algorithm $\mathscr{A}$ is an (α,β)-accurate PAC learner for the hypothesis class H if for all distributions D over input, output pairs, $\mathscr{A}$ on being given a sample of size m drawn i.i.d. from D outputs a hypothesis h such that with probability greater than or equal to $1-\beta$ over the randomness of the sample and the algorithm,*

$$\mathrm{err}_D(h) \leq \inf_{f\in H} \mathrm{err}_D(f) + \alpha.$$

where $\mathrm{err}_D(h) = \mathbf{Pr}_{(x,y)\in D}[h(x) \neq y]$. *In this context, we will sometimes refer to PAC learning as the* ***realizable setting****.*

2.3.3 PAC Learning with Massart Noise

Chapter 4 is concerned with PAC learning in the presence of Massart noise.

Definition 2.3.5 (PAC Learning with Massart Noise [MN06]). *Let $\mathscr{C}$ be a concept class over $X = \mathbb{R}^n$, D_x be any fixed but unknown distribution over X, and $0 \leq \eta < 1/2$ be the noise parameter. Let $f \in \mathscr{C}$ be the unknown target concept. A* noisy example oracle, $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, *works as follows: Each time $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ is invoked, it returns a labeled example (x,y), where $x \sim D_x$, $y = f(x)$ with probability $1-\eta(x)$ and $y = -f(x)$ with probability $\eta(x)$, for an* unknown *function $\eta(x) : X \to [0,\eta]$. Let D denote the joint distribution on (x,y) generated by the above oracle. A learning algorithm is given i.i.d. samples from D and its goal is to output a hypothesis h such that with high probability the misclassification error $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y]$ is as small as possible.*

The Massart noise model is a semi-random input model, in which different inputs are allowed to have different probabilities of being corrupted. For more information, see Section 4.1.1.

2.4 Correlated Sampling

Correlated sampling is an important primitive for designing replicable algorithms.

In the correlated sampling problem, multiple players are given probability distributions $\mathscr{P}, \mathscr{Q}$ over the same (typically finite) set and access to shared randomness. Without communicating, the players want to sample from their respective distributions, while maximizing the probability that their outputs agree (i.e., are correlated).

Formally, let $Y = \{0,1\}$, and for a set Ω, 2^Ω denotes the set of all functions from Ω to Y, and Δ_Ω denotes the set of all sampleable distributions on Ω. For two distributions $\mathscr{P}$, $\mathscr{Q}$ over Ω, let $d_{TV}(\mathscr{P}, \mathscr{Q})$ denote the total variational distance between $\mathscr{P}$ and $\mathscr{Q}$.

Definition 2.4.1. *(Correlated Sampling) A correlated sampling strategy for a finite set Ω with error $\varepsilon : [0,1] \to [0,1]$ is an algorithm $CS : \Delta_\Omega \times \mathscr{R}'$ and a distribution $\mathscr{R}'$ on random strings such that:*

- *(Marginal Correctness) For all $\mathscr{P} \in \Delta_\Omega$ and $w \in \Omega$, $\mathbf{Pr}_{r' \sim \mathscr{R}'}[CS(\mathscr{P}, r') = w] = \mathscr{P}(w)$.*
- *(Error Guarantee) For all $\mathscr{P}, \mathscr{Q} \in \Delta_\Omega$, $\mathbf{Pr}_{r' \sim \mathscr{R}'}[CS(\mathscr{P}, r') \neq CS(\mathscr{Q}, r')] \leq \varepsilon(d_{TV}(\mathscr{P}, \mathscr{Q}))$*

Correlated sampling can be used to construct replicable algorithms as follows. First, design an algorithm $A(S;r)$ with stable outputs. Then, apply a correlated sampling strategy on the outputs.

This overall procedure is replicable if algorithm the outputs distributions of A are close in total variational distance. Let $\mathscr{P}_S$ be the distribution over outputs induced by running algorithm $A(S;r)$ on a fixed input S. Say we designed algorithm A such that, for most i.i.d. samples S_1, S_2, the total variation distance $d_{\mathrm{TV}}(\mathscr{P}_{S_1}, \mathscr{P}_{S_2})$ is small. Then, a correlated sampling strategy can use shared randomness r to correlate the outputs of $A(S_1;r)$ and $A(S_2;r)$, without prior knowledge of the specific S_1 and S_2 chosen. Correctness and replicability of this procedure are implied by the definition of correlated sampling strategies.

However, there are computational limitations with respect to using correlated sampling generally. For more information, see Section 5.2.5.

In Section 5.4.2, we show how to design correlated sampling algorithms, assuming the existence of uniform one-way function inverters. We model samplable distributions by considering the distribution induced by evaluating circuits on random inputs. Furthermore, we incorporate a distributional error parameter ν, giving some slack in the marginal correctness guarantee of a correlated sampler.

Definition 2.4.2 (Implicit Correlated Sampling Algorithm)**.** *Let $m,n \in \mathbb{Z}^+$, and let $C : \{0,1\}^m \to \{0,1\}^n$ denote a circuit. Let distributional error parameter $\nu > 0$. $\mathscr{B}(C,\nu;r)$ is an (m,n,ν)-*implicit correlated sampling algorithm *if the following conditions hold:*

1. ***Inputs/Outputs:*** *$\mathscr{B}$ takes as input a circuit $C : \{0,1\}^m \to \{0,1\}^n$, a distributional error parameter ν, and a random string r. $\mathscr{B}$ outputs a string in $\{0,1\}^n$.*

2. ***ν-distributional accuracy:*** *For all circuits $C : \{0,1\}^m \to \{0,1\}^n$, the distributions D_C and $D_{\mathscr{B}(C,\nu)}$ satisfy $d_{\mathrm{TV}}(D_C, D_{\mathscr{B}(C,\nu)}) \leq O(\nu)$.*

   Here, D_C denotes the distribution over $\{0,1\}^n$ induced by querying $C(r)$ on uniformly random inputs r, i.e., probability density function $p_{D_C}(x) = \mathbf{Pr}_{r \sim U^m}[C(r) = x]$. Similarly, $D_{\mathscr{B}(C,\nu)}$ denotes the distribution over $\{0,1\}^n$ induced by querying $\mathscr{B}(C,\nu;r)$ with uniformly random strings r.

3. ***Correlated sampling:*** *For all pairs of circuits $C_1, C_2 : \{0,1\}^m \to \{0,1\}^n$, $\mathbf{Pr}_r[\mathscr{B}(C_1,\nu;r) \neq \mathscr{B}(C_2,\nu;r)] \in O(d_{\mathrm{TV}}(D_{C_1}, D_{C_2}) + \nu)$.*

We construct correlated sampling algorithms from the assumption that one can invert any one-way function on almost all inputs.

Definition 2.4.3 (Uniform One-Way Function Inverters)**.** *Let $\nu' > 0$. $\mathscr{I}_{\nu'}(C,y)$ is a* uniform one-way function inverter with error ν' *if $\mathscr{I}$ runs in randomized polynomial time in m, n, and $1/\nu'$ and if, for any circuit $C : \{0,1\}^m \to \{0,1\}^n$, $\mathbf{Pr}_{r' \sim \{0,1\}^m}[C(\mathscr{I}_{\nu'}(C, C(r'))) = C(r')] \geq 1 - \nu'$.*

Pairwise-independent hash families are also used in our correlated sampler's subroutines.

Definition 2.4.4 (Pairwise-Independent Hash Family)**.** *A family of Boolean functions* $\mathscr{H} = \{H | H : \{0,1\}^m \to \{0,1\}^n\}$ *is* pairwise-independent *if, for all* $r_1 \neq r_2 \in \{0,1\}^m$ *and* $x_1, x_2 \in \{0,1\}^n$, $\mathbf{Pr}_{H \in \mathscr{H}}[H(r_1) = x_1 \wedge H(r_2) = x_2] = 2^{-2n}$.

See Section 5.4.2 for more details.

Acknowledgements

Chapter 2, in part, is a reprint of the material as it appeared in three publications. The first, on Arxiv at https://arxiv.org/abs/2201.08430v2, which is the full version of the paper that appeared in the Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing 2022. Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sorrell, Jessica. "Reproducibility in Learning". The second, in the Proceedings of the 34th Annual Conference on Learning Theory 2022. Diakonikolas, Ilias; Impagliazzo, Russell; Kane, Daniel M.; Lei, Rex; Sorrell, Jessica; Tzamos, Christos. "Boosting in the Presence of Massart Noise". The third, on Arxiv at https://arxiv.org/abs/2303.12921v2, which is the full version of the paper that appeared in the Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing 2023. Bun, Mark; Gaboardi, Marco; Hopkins, Max; Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sivakumar, Satchit; Sorrell, Jessica. "Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization". The dissertation author was the primary investigator and author of these papers.

Chapter 3

Replicability* in Learning

We introduce the notion of a *replicable*[1] algorithm in the context of learning. A replicable learning algorithm is resilient to variations in its samples — with high probability, it returns the exact same output when run on two samples from the same underlying distribution. We begin by unpacking the definition, clarifying how randomness is instrumental in balancing accuracy and replicability. We initiate a theory of replicable algorithms, showing how replicability implies desirable properties such as data reuse and efficient testability. Despite the exceedingly strong demand of replicability, there are efficient replicable algorithms for several fundamental problems in statistics and learning. First, we show that any statistical query algorithm can be made replicable with a modest increase in sample complexity, and we use this to construct replicable algorithms for finding approximate heavy-hitters and medians. Using these ideas, we give the first replicable algorithm for learning halfspaces via a replicable weak learner and a replicable boosting algorithm. Interestingly, we utilize a connection to foams [KORW12] as a higher-dimension randomized rounding scheme. Finally, we initiate the study of lower bounds and inherent tradeoffs for replicable algorithms, giving nearly tight sample complexity upper and lower bounds for replicable versus nonreplicable SQ algorithms.

[1]Terminology Note: This chapter is largely based on a publication originally titled "Reproducibility in Learning". In that work, we defined a mathematical notion of algorithmic stability and termed it "reproducibility". Since then, we have renamed this mathematical notion "replicability". We changed the terminology to be more consistent with current guidance from the Association for Computing Machinery [Ass20] and the broader scientific community [Nat19], regarding usage of the terms "replicability" and "replicability". Section 5.2.6 discusses this change in more detail.

3.1 Introduction

Reproducibility is vital to ensuring scientific conclusions are reliable, and researchers have an obligation to ensure that their results are replicable. In the last twenty years, lack of reproducibility has been a major issue in nearly all scientific areas of study. For example, a 2012 Nature article by Begley and Ellis reported that the biotechnology company Amgen was only able to replicate 6 out of 53 landmark studies in haematology and oncology [BE12]. In a 2016 Nature article, Baker published a survey of 1500 researchers, reporting that 70% of scientists had tried and failed to replicate the findings of another researcher, and that 52% believed there is a significant crisis in reproducibility [Bak16].

A key issue underlying the reproducibility crisis (as articulated in many articles, e.g., [Ioa05]) is the fact that new data/publications are growing at an exponential rate, giving rise to an explosion of methods for data generation, screening, testing, and analysis, where, crucially, only the combinations producing the most significant results are reported. Such practices (also known as P-hacking, data dredging, and researcher degrees of freedom) can lead to erroneous findings that appear to be significant, but that don't hold up when other researchers attempt to replicate them. Identifying and mitigating these problems is quite subtle. First, is not easy to come up with an agreed-upon set of practices that guarantees reproducibility, and secondly, testing to determine whether or not a finding is statistically significant is a complex task.

Within the subfields of machine learning and data science, there are similar concerns about the reliability of published findings. The performance of models produced by machine learning algorithms may be affected by the values of random seeds or hyperparameters chosen during training, and performance may be brittle to deviations from the values disseminated in published results [HIB$^+$17, IHGP17, LKM$^+$18]. To begin addressing concerns about reproducibility, several prominent machine learning conferences have begun hosting reproducibility workshops and holding reproducibility challenges, to promote best practices and encourage researchers to share the code used to generate their results [PVLS+20].

In this work, we aim to initiate the study of replicability as a property of algorithms themselves, rather than the process by which their results are collected and reported. We define the following notion of replicability, which informally says that a randomized algorithm is replicable if two distinct runs of the algorithm on two sets of samples drawn from the same distribution, with internal randomness fixed between both runs, produces the *same* output with high probability.

Definition 3.1.1 (Replicability)**.** *Let D be a distribution over a universe $\mathscr{X}$, and let $\mathscr{A}$ be a randomized algorithm with sample access to D. $\mathscr{A}(\vec{s})$ is ρ-replicable if*

$$\mathbf{Pr}_{\vec{s}_1,\vec{s}_2,r}\left[\mathscr{A}(\vec{s}_1;r)=\mathscr{A}(\vec{s}_2;r)\right]\geq 1-\rho,$$

where $\vec{s}_1$ and $\vec{s}_2$ denote sequences of samples drawn i.i.d. from D, and r denotes a random binary string representing the internal randomness used by $\mathscr{A}$.

Our definition of replicability is inspired by the literature on pseudodeterministic algorithms, particularly the work of Grossman and Liu [GL19] and Goldreich [Gol19]. In the pseudodeterministic setting, the primary concern is replicating the output of an algorithm given the same input, over different choices of the algorithm's internal randomness. Our notion (Definition 3.1.1) is more suitable for the setting of machine learning, where it is desirable to reproduce the exact same output of an algorithm (with high probability) over different sample sets drawn from a distribution D.

We observe the following key properties of Definition 3.1.1.

Stability. Replicability is a strong stability property that implies independent parties can replicate previous results with high probability, so long as the randomness used to achieve these results is made public. For researchers solving machine learning and data analysis tasks, replicability allows researchers to verify published results with high probability, as long as the datasets are drawn from the same distribution.

Generalization. Replicability implies generalization. A replicable learning algorithm, with high probability, outputs a hypothesis h such that the difference between the risk of h and the empirical risk of h on the training set is small. Intuitively, replicabilitiy implies that h is independent of the training set with high probability. Thus, a Hoeffding bound can be applied to bound the risk in terms of the empirical risk.

Privacy. Differential privacy (DP) is an important notion that requires small distance between the two distributions induced by an algorithm, when run on any two datasets that differ in a single element. Crucially, it asks for the guarantees in the *worst case over datasets*. Replicable algorithms guarantee a different form of privacy: If $\mathscr{A}$ is replicable, then what $\mathscr{A}$ learns (for example, a trained classifier) is almost always the same; thus, $\mathscr{A}$ is usually *independent* of the chosen training data. In this way, replicable algorithms are prevented from memorizing anything that is specific to the training data, similar to differentially private algorithms. Replicability is weaker than differential privacy in the sense that replicability only applies to in-distribution samples, whereas differential privacy applies to *any* training set. On the other hand, replicability is stronger in the sense that its guarantee for in-distribution samples is global rather than local (for neighboring samples).

Testability. While differential privacy has become the standard for privacy-preserving computation, an important issue that is the subject of extensive research is testing and verifying differential privacy. As discussed in [GNP20], DP-algorithms and their implementations are usually analyzed by hand, and proofs of differential privacy are often intricate and prone to errors. Implementing such an algorithm in practice often gives rise to DP leaks, due to coding errors or assumptions made in the proof that do not hold on finite computers (such as the ability to sample from continuous distributions). Moreover, the complexity of verifying differential privacy is hard. Verification in the black-box setting (where the auditor has oracle access to the learning algorithm) was recently shown to be infeasible, as low query complexity implies high values of the the privacy parameters ε and δ [GM18]. In the white-box setting where $\mathscr{A}$ is given to the tester, [GNP20] shows that testing for differential privacy is $coNP^{\#P}$-complete. This has led to an active research area aiming at developing automated as well as interactive testing and verification methods for differential privacy [NFPH15, GHH$^+$13, RP10, AH18, BGA$^+$15, BCK$^+$21, FJ14, ZK17]. In contrast, replicability is a form of privacy that can be *efficiently tested* in (randomized) polynomial time (in the dimension of the data universe and ρ).

3.1.1 Our Main Results

Replicability: Properties and Alternative Definitions

We discuss alternative definitions of replicability and show that they are all essentially equivalent. Then, we also prove some other nice properties of replicable algorithms. (All formal statements and proofs are in Appendix A.)

1. **Alternative Definitions and Amplification.** We start by discussing two alternative definitions of replicability and relate them to our definition. First, we can generalize the definition to include algorithms $\mathscr{A}$ that not only have access to internal randomness and to random samples from an underlying distribution D, but that also have access to extra non-random inputs. This more general definition captures both the original definition of pseudodeterministic algorithms as well as our definition of replicable learning algorithms, and all of our results remain unchanged. Second, we discuss an alternative two-parameter definition, and show that the definitions are qualitatively the same. We show how to amplify the replicability parameter by a standard argument where the sample complexity is increased modestly.

2. **Public versus Private Randomness.** Recall that we define replicability as the probability that an algorithm returns the same answer when run twice using different random samples from D but the same internal randomness. In [GL19], the authors define a related concept in which the internal randomness is divided into two pieces, public and private randomness, but the algorithm should return the same answer when just the public randomness is held fixed.

We show that, without loss of generality, it suffices to use only public randomness.

3. **Replicability Implies Generalization.** Learning algorithms attempt to use finite samples to generate hypotheses on unknown, possibly complex distributions. The error of a hypothesis h on the underlying distribution is called the generalization error. A replicable algorithm outputs the same hypothesis with high probability, and thus the algorithm seldom draws distinctions between specific samples and the entire distribution.

4. **Connections to Data Reuse.** We explore the connection between replicable algorithms and the adaptive data analysis model discussed in [DFH$^+$15b] and [DFH$^+$15a]. We show that replicable algorithms are strongly resilient against adaptive queries. Informally, with respect to replicable algorithms, the sample complexity and accuracy of (replicably) answering m adaptively chosen queries behaves similarly to the sample complexity and accuracy of replicably answering m *nonadaptively* chosen queries.

Upper Bounds

Our main technical results are replicable algorithms for some well-studied statistical query and learning problems that are used as building blocks in many other algorithms.

1. **Simulating SQ Algorithms.** In Section 3.2, we give a generic algorithm that reduces the problem of ρ-replicably estimating a single statistical query with tolerance τ and error δ to that of *nonreplicably* estimating the same query within a smaller tolerance and error.

   Theorem 3.1.2 (Theorem 3.2.3, Restated). *Let $\psi : \mathscr{X} \to \{0,1\}$ be a statistical query. Then the sample complexity of ρ-replicably estimating ψ within tolerance τ and error δ is at most the sample complexity of (nonreplicably) estimating ψ within tolerance $\tau' = \tau\rho$ and error $\delta' = \tau\delta$.*

   The basic idea is to obtain an estimate of the statistical query with a smaller tolerance τ' and then use a randomized rounding scheme where the interval $[0,1]$ is divided into intervals of size roughly τ/ρ. Then, every value in the interval is rounded to the midpoint of the region it occurs in. The partition into intervals is chosen with a random offset so that with high probability nearby points will lie in the same region.

2. **Heavy-hitters.** Using our simulation of SQ queries, in Section 3.3, we demonstrate the usefulness of replicability by giving a replicable algorithm `rHeavyHitters` for identifying approximate v-heavy-hitters of a distribution, i.e. the elements in the support of the distribution with probability mass at least v.

   Lemma 3.1.3 (Lemma 3.3.3, Restated). *For all $\varepsilon \in (0,1/2)$, $v \in (\varepsilon, 1-\varepsilon)$, with probability at least $1-\rho$, $\texttt{rHeavyHitters}_{\rho,v,\varepsilon}$ is ρ-replicable, and returns a list of v'-heavy-hitters for some $v' \in [v-\varepsilon, v+\varepsilon]$. Furthermore, the sample complexity is bounded by $\widetilde{O}(\rho^{-2})$.*

   The high level idea of our algorithm is to first draw sufficiently many samples, $\vec{s}_1$, $Q_1 = |\vec{s}_1|$, so that with high probability all heavy-hitters are in $\vec{s}_1$. In the second stage, we draw a fresh set $\vec{s}_2$ of Q_2 many samples and use them to empirically estimate the density of each element in $\vec{s}_1$, and remove those that aren't above the cutoff v', where v' is chosen randomly from $[v-\varepsilon, v+\varepsilon]$ to avoid boundary issues.

3. **Median Finding.** In Section 3.4, we design a replicable algorithm for finding an approximate median in an arbitrary distribution over a finite domain. Approximate median finding is a fundamental statistical problem, and is also extensively studied in the privacy literature.

   Theorem 3.1.4 (Theorem 3.4.2, Restated). *Let $\tau, \rho \in [0,1]$ and let $\delta = 1/3$. Let D be a distribution over $\mathscr{X}$, where $|\mathscr{X}| = 2^d$. Then $\texttt{rMedian}_{\rho,d,\tau,\delta}$ is ρ-replicable, outputs a τ-approximate median of D with success probability $1-\delta$, and has sample complexity*

$$\tilde{\Omega}\left(\left(\frac{1}{\tau^2(\rho-\delta)^2}\right)\cdot\left(\frac{3}{\tau^2}\right)^{\log^*|\mathscr{X}|}\right)$$

To describe the key ideas in the algorithm, we first show how approximate-median finding is useful for turning many algorithms into replicable ones. Consider any problem where the correct answers form an interval, and assume we start with a (not-necessarily) replicable algorithm that is mildly accurate. Then we can run a replicable approximate-median finding algorithm on the distribution of outputs of the original algorithm to construct a very accurate replicable algorithm.

We will actually use this strategy recursively to replicably solve approximate median itself. Our algorithm recursively composes a mildly accurate replicable median algorithm with a generic very accurate non-replicable median algorithm. This recursive technique is inspired by, but simpler than, previous algorithms in the privacy literature [BNSV15, KLM$^+$20], and like these algorithms, the sample complexity of our algorithm has a non-constant but very slowly growing dependence on the domain size.

4. **Learning Halfspaces.** In Section 3.5, we obtain a replicable algorithm `rHalfspaceWkL` for weakly learning halfspaces. In Section 3.6, we transform it into a replicable strong learner by way of a replicable boosting algorithm `rBoost`. We stress that our algorithms for halfspaces are replicable in the stronger distribution-free setting.

   Theorem 3.1.5 (Corollary 3.6.5, Restated)**.** *Let D be a distribution over $\mathbb{R}^d$, and let $f : \mathbb{R}^d \to \{\pm 1\}$ be a halfspace with margin τ in D. For all $\rho, \varepsilon > 0$. Algorithm* `rBoost` *run with weak learner* `rHalfspaceWkL` *ρ-replicably returns a hypothesis $\mathbf{h}$ such that, with probability at least $1-\rho$, $\mathbf{Pr}_{\vec{x}\sim D}[\mathbf{h}(\vec{x}) = f(\vec{x})] \geq 1-\varepsilon$. Furthermore, the overall sample complexity is $\widetilde{O}\left(\frac{d^{10/9}}{\tau^{76/9}\rho^{20/9}\varepsilon^{28/9}}\right)$.*

   In order to replicably learn halfspaces, we start with a simple weak learning algorithm for halfspaces [Ser02] that takes examples $(\vec{x}_i, y_i) \in \mathscr{X} \times \{\pm 1\}$, normalizes them, and returns the halfspace defined by vector $\sum_i \vec{x}_i \cdot y_i$. We show a concentration bound on the sum of normalized vectors from a distribution, and then argue that all vectors within the concentration bound are reasonable hypothesis with non-negligible advantage.

Our randomized rounding scheme is a novel application of the randomized rounding technique developed in the study of foams [KORW12]. The concentration bound together with the foams rounding scheme [KORW12] yields a replicable halfspace weak learner. We then obtain our replicable strong learner for halfspaces by combining it with a (new) replicable boosting algorithm. Our algorithm is sample-efficient but inefficient with respect to runtime, due to the inefficiency of the foams rounding scheme. We also give another randomized rounding procedure that gives a polynomial-time strong replicable halfspace learner, but with polynomially larger sample complexity.

The Price of Replicability.

In Section 3.7 we ask what is the cost of turning a nonreplicable algorithm into a replicable one. We first show that a τ-tolerant ρ-replicable SQ algorithm $\mathscr{A}$ for ϕ implies a ρ-replicable algorithm for the τ-coin problem: given samples from a p-biased coin with the promise that either $p \geq 1/2+\tau$ or $p \leq 1/2-\tau$, determine which is the case. Our main result in this section are nearly tight upper and lower bound bounds of $\Theta(\tau^{-2}\rho^{-2})$ on the sample complexity of ρ-replicably solving the τ-coin problem (for constant δ), and thus the same bounds for ρ-replicably answering SQ queries. On the other hand, it is well-known that the *nonreplicable* sample complexity of the τ-coin problem is $\Theta(\tau^{-2}\log(1/\delta))$ (see, e.g. [Mou]). So the cost of guaranteeing ρ-replicability for SQ queries is a factor of ρ^{-2}.

For upper bounds, our generic algorithm in Section 3.2 converts any SQ query into a replicable one: if our end goal is a ρ-replicable algorithm for estimating a statistical query with tolerance τ and error δ, then the sample complexity is at most the sample complexity of *nonreplicably* answering the query to within tolerance τ', and success probability $1-\delta'$ where $\tau' = O(\tau\rho)$ and $\delta' = O(\delta\tau)$, which has sample complexity $O(\tau^{-2}\rho^{-2}\log(1/\delta'))$. The main result in this section is the following lower bound for ρ-replicably answering statistical queries.

Theorem 3.1.6 (Theorem 3.7.1, Restated)**.** *Let $\tau > 0$ and let $\delta < 1/16$. Any ρ-replicable algorithm for solving the τ-coin coin problem with success probability at least $1-\delta$ requires sample complexity $\Omega(\tau^{-2}\rho^{-2})$.*

Related Work.

A subset of these results [ILS21] was presented at the TPDP 2021 workshop.

Our Definition 3.1.1 is inspired by the literature on pseudodeterministic algorithms [GG11, GGR13, GG17, GGH18, GGMW19, GL19, Gol19]. In particular, [GL19] and [Gol19] define reproducibility in the context of pseudodeterminism. There, the input of a reproducible algorithm is a fixed string. In our setting, the input of a replicable learning algorithm is a distribution, only accessible by randomly drawing samples.

Independently of our work, [GKM21] define a property equivalent to replicability, called "pseudo-global stability". Their (α,β)-accurate (η',ν')-pseudo-global stability definition is equivalent to the (η,ν)-replicability definition discussed in Appendix 3.8, except that pseudo-global stability includes explicit parameters for correctness and sample complexity. In Appendix 3.8, we show that these two definitions are equivalent to Definition 3.1.1 up to polynomial factors. [GKM21] gives pseudo-globally stable SQ algorithms, an amplification of the stability parameter, and an algorithm to find a heavy-hitter of a distribution. The authors use pseudo-global stability to show that classes with finite Littlestone dimension can be learned user-levelly privately, and they connect pseudo-global stability to approximate differential privacy. Pseudo-global stability is a generalization of global stability, introduced in [BLM20]. Those authors use global stability as an intermediate step to show that classes with finite Littlestone dimension can be learned privately, and they show how global stability implies generalization.

Our work is related to other notions of stability in machine learning which, like our definition, are properties of learning algorithms. In the supervised learning setting, stability is a measure of how much the output of a learning algorithm changes when small changes are made to the input training set. An important body of work establishes strong connections between the stability of a learning algorithm and generalization [DW79b, DW79a, KR99, BE02, SSSSS10]. Distributional notions of stability which remain stable under composition and postprocessing, were defined and shown to be closely connected to differential privacy and adaptive data analysis (e.g., [BNS$^+$16a, DFH$^+$15a]). In fact, the definition of differential privacy itself is a form of stability known as max-KL stability. Stability-based principles have also been explored in the context of unsupervised learning where model selection is a difficult problem since there is no ground truth. For example, a stable algorithm for clustering has the property that when the algorithm is applied to different data sets from the same distribution, it will yield similar outputs (e.g., [vL10]).

In all of these settings, stability depends on how close the outputs are when the inputs are close; what varies is the particular measure of closeness in input and output space. For example, closeness in the output can be with respect to function or parameter space; for distributional stability close means that the output distributions are close with respect to some metric over distributions. Our definition of replicability can be viewed as an extreme form of stability where the output is required to be *identical* almost all of the time, and not just similar. Thus replicability enjoys many of the nice properties of stable algorithms (e.g., postprocessing, composition) but has the advantage of being far easier to verify.

Open Questions and Future Work

One motivation for examining replicability in algorithms is the "replicability crisis" in experimental science. Can we use replicability to create statistical methodologies that would improve replicability in published scientific work? A concrete step towards this would be to design replicable hypothesis testing algorithms. We can view a null hypothesis as postulating that data will come from a specific distribution D, and want algorithms that accept with high probability if the data comes from D (or a "close" distribution) and reject with good probability if the data distribution is "far" from D. For example, the coin problem is a degenerate case in which the data are Boolean and the distance is the difference in the expected values. For different types of data and distance metrics, what is the optimal sample complexity of hypothesis testing, and how much more is that for replicable hypothesis testing?

A related problem is that of learning under distributional shifts, or individual-based fair learning (where we want the learning algorithm to treat similar people similarly with respect to a similarity metric defining closeness). A key step in making algorithms replicable is a randomized procedure to round the output of a standard empirical learner to a *single* hypothesis in a way that is independent of the underlying distribution. Can similar ideas be used to design learning algorithms robust to distributional shifts, or to give more informed performance metrics?

This work establishes that there exist replicable algorithms for a variety of learning problems. However, we do not characterize exactly which learning algorithms can be made replicable, or how replicability affects the required sample complexity. Is it possible to identify an invariant of concept classes which characterizes the complexity of replicable learning, analogous to VC-dimension for PAC learning [VC71], representation dimension and one-way communication complexity for exact differential privacy [FX14, BNS13], and Littlestone Dimension for approximate differential privacy [BLM20]? A specific problem of interest is that of learning linear functions over finite fields. If the data has full dimension, the function can be solved for uniquely; so, designing replicable algorithms when the data does not form a basis seems interesting.

Also, we described the first replicable boosting algorithm. Are there natural conditions under which a boosting algorithm can always be made replicable? Are the sample complexity upper bounds we obtain for our applications tight or close to tight? In particular, is there a replicable algorithm for approximate median that has only $\log^* |\mathscr{X}|$ dependence on the domain size?

Replicability provides a distinctive type of privacy. Except with the small probability ρ, a replicable algorithm's outputs are a function entirely of the underlying distribution and the randomness of the algorithm, not the samples. Thus, a replicable algorithm seldom leaks information about the specific input data. We borrowed techniques from the study of private data analysis and differential privacy, and we hope that future work will formalize connections between replicability and private data analysis. We also hope that some applications of differential privacy will also be achievable through replicability.

3.2 Statistical Queries

We show how to use randomized rounding to replicably simulate any SQ oracle and therefore any SQ algorithm. The statistical query model introduced by [Kea98] is a restriction of the PAC-learning model introduced by [Val84]. We consider the statistical query oracle primarily in the context of unsupervised learning (e.g., see [Fel16]).

Definition 3.2.1 (Statistical query oracle)**.** *Let $\tau \in [0,1]$ and $\phi : \mathscr{X} \to [0,1]$ be a query. Let D be a distribution over domain $\mathscr{X}$. A* statistical query oracle *for D, denoted $\mathscr{O}_D(\tau,\phi)$, takes as input a tolerance parameter τ and a query ϕ, and outputs a value v such that $|v - \mathbb{E}_{x\sim D}[\phi(x)]| \leq \tau$.*

Definition 3.2.2 (Simulating a statistical query oracle)**.** *Let $\delta \in [0,1]$ and τ, ϕ, D be as above. Let $\mathscr{O}_D$ be a statistical query oracle for D. Let $\vec{s}$ denote an i.i.d. sample drawn from D. We say that a routine* `STAT` simulates *$\mathscr{O}_D$ with failure probability δ if for all τ, δ, ϕ, there exists an $n_0 \in \mathbb{N}_+$ such that if $n > n_0$, $v \leftarrow \texttt{STAT}(\tau, \phi, \vec{s})$ satisfies $|v - \mathbb{E}_{x\sim D}[\phi(x)]| \leq \tau$ except with probability δ.*

To denote a routine simulating a statistical query oracle for fixed parameters τ, ϕ, and (optionally) ρ, we write these parameters as subscripts.

Algorithm 3. $\texttt{rSTAT}_{\rho,\tau,\phi}(\vec{s})$
Parameters: τ - tolerance parameter
ρ - replicability parameter
ϕ: a query $X \to [0,1]$

```
1: $\alpha = \frac{2\tau}{\rho+1-2\delta}$
2: $\alpha_{\text{off}} \leftarrow_r [0,\alpha]$
3: Split $[0,1]$ in regions: $R = \{[0,\alpha_{\text{off}}), [\alpha_{\text{off}}, \alpha_{\text{off}}+\alpha), \ldots, [\alpha_{\text{off}}+i\alpha, \alpha_{\text{off}}+(i+1)\alpha), \ldots, [\alpha_{\text{off}}+k\alpha, 1)\}$
4: $v \leftarrow \frac{1}{|\vec{s}|} \sum_{x\in\vec{s}} \phi(x)$
5: Let $r_v$ denote the region in $R$ that contains $v$
6: **return** the midpoint of region $r_v$
```

Theorem 3.2.3 upper bounds the sample complexity of $\texttt{rSTAT}_{\tau,\rho,\phi}$. In Section 3.7, we show this upper bound is tight as a function of ρ.

Theorem 3.2.3 (rSTAT simulates a statistical query oracle)**.** *Let $\tau, \delta, \rho \in [0,1]$, $\rho > 2\delta$, and let $\vec{s}$ be a sample drawn i.i.d. from distribution D. Then if*

$$|\vec{s}| \in \tilde{O}\left(\frac{1}{\tau^2(\rho-2\delta)^2}\right)$$

$\texttt{rSTAT}_{\rho,\tau,\phi}(\vec{s})$ *ρ-replicably simulates an SQ oracle $\mathscr{O}_{D,\tau,\phi}$ with failure rate δ.*

In Section 3.7, we will prove a near matching lower bound on the sample complexity of ρ-replicably estimating a statistical query with tolerance τ and success probability $1-\delta$.

Proof. We begin by showing that $\texttt{rSTAT}_{\rho,\tau,\phi}$ simulates an SQ oracle $\mathscr{O}_{D,\tau,\phi}$ with failure rate δ.

Let $\tau' = \frac{\tau(\rho-2\delta)}{\rho+1-2\delta}$. Recall $\alpha \overset{\text{def}}{=} \frac{2\tau}{\rho+1-2\delta}$, so $\frac{2\tau'}{\alpha} = \rho - 2\delta$. A Chernoff bound gives that

$$\left| \frac{1}{|\vec{s}|} \sum_{x \in \vec{s}} \phi(x) - \mathop{\mathbb{E}}_{x \sim D} \phi(x) \right| \leq \tau' = \frac{\tau(\rho - 2\delta)}{\rho + 1 - 2\delta}$$

except with failure probability δ, so long as $|\vec{s}| \geq \log(2/\delta)/(2\tau'^2)$. Outputting the midpoint of region r_v can further offset this result by at most $\alpha/2 = \frac{\tau}{\rho+1-2\delta}$. Therefore

$$|v - \mathbb{E}_{x\sim D}\phi(x)| \leq \frac{\tau(\rho-2\delta)}{\rho+1-2\delta} + \frac{\tau}{\rho+1-2\delta} = \tau,$$

except with probability δ, so long as the sample $\vec{s}$ satisfies

$$\log(2/\delta)/(2\tau'^2) = \frac{\log(2/\delta)(\rho+1-2\delta)^2}{2\tau^2(\rho-2\delta)^2} \leq \frac{4\log(2/\delta)}{2\tau^2(\rho-2\delta)^2} \leq |\vec{s}|.$$

We now show that $\texttt{rSTAT}_{\rho,\tau,\phi}$ is ρ-replicable by considering two invocations of $\texttt{rSTAT}_{\rho,\tau,\phi}$ with common randomness r on samples $\vec{s}_1, \vec{s}_2 \sim D$ respectively. The probability that either empirical estimate of $\mathbb{E}_{x\sim D}[\phi(x)]$ fails to satisfy tolerance τ' is at most 2δ. Denote by v_1 and v_2 the values returned by the parallel runs $\texttt{rSTAT}(\vec{s}_1; r)$ and $\texttt{rSTAT}(\vec{s}_2; r)$ at line 4. Conditioning on success, values v_1 and v_2 differ by at most $2\tau'$. $\texttt{rSTAT}$ outputs different values for the two runs if and only if v_1 and v_2 are in different regions of R, determined by the common randomness r. This occurs if some region's endpoint is between v_1 and v_2; since α_{off} is chosen uniformly in $[0, \alpha]$, the probability that v_1 and v_2 land in different regions is at most $2\tau'/\alpha = \rho - 2\delta$. Accounting for the 2δ probability of failure to estimate $\mathbb{E}_{x\sim D}[\phi(x)]$ to within tolerance, $\texttt{rSTAT}_{\rho,\tau,\phi}(\vec{s})$ is ρ-replicable. □

3.3 Heavy-hitters

Next, we present our replicable approximate heavy-hitters algorithm, analyzing its sample complexity and replicability. We will use this algorithm as a subroutine in later algorithms such as in the approximate-median algorithm. Also, we will show how to use this algorithm to give a generic way to boost replicability from constant ρ to arbitrarily small ρ.

Definition 3.3.1 (Heavy-Hitter)**.** *Let D be a distribution over $\mathscr{X}$. Then we say $x \in \mathscr{X}$ is a v-*heavy-hitter *of D if $\mathbf{Pr}_{x' \sim D}[x' = x] \geq v$.*

Definition 3.3.2 ((Approximate) Heavy-Hitter Problem)**.** *Let L_v be the set of $x \in supp(D)$ that are v-heavy-hitters of D. Given sample access to D, output a set L satisfying $L_{v+\varepsilon} \subseteq L \subseteq L_{v-\varepsilon}$.*

Let D be a distribution over $\mathscr{X}$. The following algorithm replicably returns a set of v'-heavy-hitters of D, where v' is a random value in $[v-\varepsilon, v+\varepsilon]$. Picking v' randomly allows the algorithm to, with high probability, avoid a situation where the cutoff for being a heavy-hitter (i.e. v') is close to the probability mass of any $x \in \text{supp}(D)$.

Algorithm 4. `rHeavyHitters`$_{\rho,v,\varepsilon}$
Input: samples $\mathscr{X}^{\text{H}}$, S from distribution D over $\mathscr{X}$ plus internal randomness r
Parameters: Target replicability ρ, target range $[v-\varepsilon, v+\varepsilon]$
Output: List of v'-heavy-hitters of D, where $v' \in [v-\varepsilon, v+\varepsilon]$

```
$\mathscr{X}^{\text{H}} \leftarrow Q_1 \overset{\text{def}}{=} \frac{\ln(6/(\rho(v-\varepsilon)))}{v-\varepsilon}$ examples from $D$  // Step 1: Find candidate heavy-hitters
$S \leftarrow Q_2 \overset{\text{def}}{=} \frac{2^6 \ln(Q_1/\rho) \cdot Q_1^2}{(\rho\varepsilon)^2}$ fresh examples from $D$// Step 2: Estimate probabilities
for all $x \in \mathscr{X}^{\text{H}}$ do
    $\widehat{p_x} \leftarrow \mathbf{Pr}_{x' \sim S}[x' = x]$            // Estimate $p_x \overset{\text{def}}{=} \mathbf{Pr}_{x' \sim D}[x' = x]$
$v' \leftarrow_r [v-\varepsilon, v+\varepsilon]$ uniformly at random      // Step 3: Remove non-$v'$-heavy-hitters
Remove from $\mathscr{X}^{\text{H}}$ all $x$ for which $\widehat{p_x} < v'$.
return $\mathscr{X}^{\text{H}}$
```

Algorithm `rHeavyHitters` returns exactly the list of v'-heavy-hitters so long as the following hold:

1. In Step 1 of Algorithm 4, all $(v-\varepsilon)$-heavy-hitters of D are included in $\mathscr{X}^{\text{H}}$.

2. In Step 2, the probabilities $\widehat{p_x}$ for all $x \in \mathscr{X}^{\mathrm{H}}$ are correctly estimated to within error $\rho\varepsilon/(3Q_1)$.

3. In Step 3, the randomly sampled v' does not fall within an interval of width $\rho\varepsilon/(3Q_1)$ centered on the true probability of a $(v-\varepsilon)$-heavy-hitter of D.

We show that these 3 conditions will hold with probability at least $1-\rho/2$, and so will hold for two executions with probability at least $1-\rho$.

Lemma 3.3.3. *For all $\varepsilon \in (0,1/2)$, $v \in (\varepsilon, 1-\varepsilon)$, with probability at least $1-\rho$,* `rHeavyHitters` *is replicable, returns a list of v'-heavy-hitters for some $v' \in [v-\varepsilon, v+\varepsilon]$, and has sample complexity $\widetilde{O}\left(\frac{1}{\rho^2\varepsilon^2(v-\varepsilon)^2}\right)$.*

Proof. We say Step 1 of Algorithm 4 succeeds if all $(v-\varepsilon)$-heavy-hitters of D are included in $\mathscr{X}^{\mathrm{H}}$ after Step 1. Step 2 succeeds if the probabilities for all $x \in \mathscr{X}^{\mathrm{H}}$ are correctly estimated to within error $\rho\varepsilon/(3Q_1)$. Step 3 succeeds if the returned $\mathscr{X}^{\mathrm{H}}$ is exactly the set of v'-heavy-hitters of D. Quantities Q_1 and Q_2 are defined in the pseudocode of Algorithm 4.

In Step 1, an individual $(v-\varepsilon)$-heavy-hitter is not included with probabilility at most $(1-v+\varepsilon)^{Q_1}$; union bounding over all $1/(v-\varepsilon)$ possible $(v-\varepsilon)$-heavy-hitters, Step 1 succeeds with probability at least $1-\frac{(1-v+\varepsilon)^{Q_1}}{v-\varepsilon} > 1-\rho/6$. Here, for clarity of presentation in the statement of Lemma 3.3.3, we make use of the inequality $v-\varepsilon < \ln(1/(1-v+\varepsilon))$.

By a Chernoff bound, each p_x is estimated to within error $\rho\varepsilon/(3Q_1)$ with all but probability $\rho/(6Q_1)$ in Step 2. Union bounding over all Q_1 possible $x \in \mathscr{X}^{\mathrm{H}}$, Step 2 succeeds except with probability $\rho/6$.

Conditioned on the previous steps succeeding, Step 3 succeeds if the randomly chosen v' is not within $\rho\varepsilon/(3Q_1)$ of the true probability of any $x \in \mathscr{X}^{\mathrm{H}}$ under distribution D. A v' chosen randomly from the interval $[v-\varepsilon, v+\varepsilon]$ lands in any given subinterval of width $\rho\varepsilon/(3Q_1)$ with probability $\rho/(6Q_1)$, and so by a union bound, Step 3 succeeds with probability at least $1-\rho/6$.

Therefore, Algorithm 4 outputs exactly the set of v'-heavy-hitters of D with probability at least $1-\rho/2$. If we consider two executions of Algorithm 4, both using the same shared randomness for chooosing v', output the set of v'-heavy-hitters of D with probability at least $1-\rho$, and so `rHeavyHitters` is ρ-replicable.

The sample complexity is $Q_1 + Q_2 \in \tilde{\Omega}\left((\rho\varepsilon(v-\varepsilon))^{-2}\right)$. □

Corollary 3.3.4. *If v, ε are constants, then* $\texttt{rHeavyHitters}_{\rho,v,\varepsilon}$ *has sample complexity* $\widetilde{O}\left(1/\rho^2\right)$.

Learning Heavy-hitters using Statistical Queries.

Next, we show that any statistical query algorithm for the v-heavy-hitters problem requires $\Omega(\log|\mathscr{X}|/\log(1/\tau))$ calls to the SQ oracle. Since Algorithm 4 has a sample complexity independent of the domain size, this implies a separation between replicable problems and problems solvable using only SQ queries.

Consider the ensemble $\{D_x\}_{x\in\mathscr{X}}$ on $\mathscr{X}$, where distribution D_x is supported entirely on a single $x \in \mathscr{X}$.

Claim 3.3.5 (Learning Heavy-hitters using Statistical Queries)**.** *Any statistical query algorithm for the v-heavy-hitters problem on ensemble $\{D_x\}_{x\in\mathscr{X}}$ requires $\Omega(\log|\mathscr{X}|/\log(1/\tau))$ calls to the SQ oracle.*

Proof. An SQ algorithm for the v-heavy-hitters problem must, for each distribution D_x, output set $\{x\}$ with high probability. An SQ oracle is allowed tolerance τ in its response to statistical query ϕ. So, for any ϕ, there must be some distribution D_x for which the following holds: at least a τ-fraction of the distributions $D_{x'}$ in the ensemble satisfy $|\phi(x') - \phi(x)| \leq \tau$. Thus, in the worst case, any correct SQ algorithm can rule out at most a $(1-\tau)$-fraction of the distributions in the ensemble with one query. If $\mathscr{X}$ is finite, then an SQ algorithm needs at least $\log_{1/\tau}(|\mathscr{X}|)$ queries. □

3.4 Approximate Median

In this section, we design a replicable algorithm for finding an approximate median in an arbitrary distribution over a finite domain. In addition to being a significant problem in its own right, and one studied extensively in the privacy literature, this is a key sub-routine for making many algorithms replicable. In particular, for any problem where the correct answers form an interval, and we have a (not-necessarily) replicable algorithm that is correct strictly more than half the time, we can run the approximate median finding algorithm on the distribution of outputs of the original to construct a reliably correct and replicable version. (In fact, we use this technique recursively within our replicable median-finding algorithm itself. Our algorithm `rMedianOfMedians` composes a mildly accurate replicable median algorithm with a generic very accurate non-replicable median algorithm.) We use a recursive technique inspired by but simpler than previous algorithms in the privacy literature [BNSV15, KLM+20], and like for these algorithms, the sample complexity of our algorithm has a non-constant but very slowly growing dependence on the domain size.

Definition 3.4.1 (τ-approximate median)**.** *Let D be a distribution over a well-ordered domain $\mathscr{X}$. $x \in \mathscr{X}$ is a τ-approximate median of D if $\mathbf{Pr}_{x' \sim D}[x' \leq x] \geq 1/2 - \tau$ and $\mathbf{Pr}_{x' \sim D}[x' \geq x] \geq 1/2 - \tau$.*

3.4.1 Replicable Approximate Median Algorithm

In this section, we present a pseudocode description of our τ-approximate median algorithm `rMedian` (Algorithm 5), and prove the following theorem.

Theorem 3.4.2 (Replicable Median)**.** *Let $\tau, \rho \in [0,1]$ and let $\delta = \rho/2$. Let D be a distribution over $\mathscr{X}$, where $|\mathscr{X}| = 2^d$. Then $\texttt{rMedian}_{\rho,d,\tau,\delta}$ (Algorithm 5) is ρ-replicable, outputs a τ-approximate median of D with all but probability δ, and has sample complexity*

$$n \in \tilde{O}\left(\left(\frac{1}{\tau^2(\rho-\delta)^2}\right) \cdot \left(\frac{3}{\tau^2}\right)^{\log^* |\mathscr{X}|}\right)$$

As an introduction to the key ideas of Algorithm 5, we consider a weighted binary tree T based on distribution D. Each internal node has two edges (a 0-edge and a 1-edge). Root-to-leaf paths represent binary representations of numbers. The weight of each internal node v is the probability that its associated binary prefix (induced by the root-to-v path) appears in an element drawn from D. If within this tree we can find a node v with weight in $[1/4, 3/4]$, then we can use the associated prefix to return an approximate median of D with approximation parameter potentially much larger than τ.

To achieve a specified approximation parameter τ, rather than using D itself to construct the binary tree T, we will use a distribution D^m over medians of D. Specifically, we use a non-replicable median algorithm to sample from τ-approximate medians of D. Identifying an approximate median of distribution D^m for even a very large approximation parameter then ensures we return a τ-approximate median of D.

The question remains of how to efficiently search T to find a node v of weight in $[1/4, 3/4]$ (under D^m). We perform this search recursively by using `rMedian` to find a prefix length ℓ such that the probability of sampling two elements from D^m agreeing on a prefix of length ℓ is large. We can then restrict our search for v to nodes near level ℓ in T (starting from the root). We apply the replicable heavy-hitters algorithm `rHeavyHitters` to find high weight nodes near level ℓ of T, and then exhaustively search the list of heavy-hitters to find an appropriate v.

We use the following non-replicable approximate median algorithm, that returns the median of its sample $\vec{s}$, as a subroutine of Algorithm 5.

Lemma 3.4.3 (Simple Median Algorithm)**.** *Let sample $\vec{s}$ be drawn from distribution D. Algorithm* `Median`$(\vec{s})$ *returns a τ-approximate median on D using $|\vec{s}| = 3(1/2-\tau)\ln(2/\delta)/\tau^2$ samples with success probability at least $1-\delta$.*

Proof. Algorithm `Median`$(\vec{s})$ fails when more than half of the elements in sample $\vec{s}$ are either i) smaller than the $(1/2-\tau)$-percentile element of D or ii) larger than the $(1/2+\tau)$-percentile element of D. Let event E_i denote the first case and event E_{ii} denote the second case. Since the elements in $\vec{s}$ are drawn i.i.d., the first event can be bounded by a Chernoff bound. Let X be a random variable denoting the number of elements in $\vec{s}$ that are smaller than the $(1/2-\tau)$-percentile element of D.

$$\begin{aligned}\mathbf{Pr}[E_i] &= \mathbf{Pr}[X \geq (1+\tau/(1/2-\tau))\,\mathbb{E}[X]] \\ &\leq \exp(-(\tau/(1/2-\tau))^2\,\mathbb{E}[X]/3) \\ &\leq \exp\left(-\frac{\tau^2}{1/2-\tau}\frac{|\vec{s}|}{3}\right) = \exp(-\ln(2/\delta)) = \delta/2\end{aligned}$$

The same argument can be used to bound the second event E_{ii}. By a union bound, the algorithm succeeds with probability at least $1-\delta$. $\square$

Before proceeding with the description of Algorithm 5, we fix some useful notation for its analysis.

- n_m - sample complexity of $\texttt{Median}_{\tau,\delta_0}$
- n_h - sample complexity of $\texttt{rHeavyHitters}_{\rho_0,\nu,\varepsilon}$
- n_{sq} - sample complexity of $\texttt{rSTAT}_{\tau,\rho_0,\phi}$
- n_d - sample complexity of $\texttt{rMedian}_{\rho,d,\tau,\delta}$
- D^m - Algorithm 5 takes as input a sample from distribution D over $\mathscr{X}$, where $|\mathscr{X}| = 2^d$. We use D^m to denote the distribution induced by sampling n_m examples from D, computing $\texttt{Median}_{\tau,\delta_0}$ on these examples, and returning the ouput
- $D_{\lceil \log d \rceil}$ - We use $D_{\lceil \log d \rceil}$ to denote the distribution induced by sampling 2 examples from D^m and returning the longest prefix ℓ on which the two medians agree. Note that this new distribution is over a new domain $\mathscr{X}'$ with $|\mathscr{X}'| = 2^{\lceil \log d \rceil} \in \Theta(d)$.
- $\rho_0 \in O(\rho/\log^*|X|)$
- $\delta_0 \in O\left((\frac{\delta}{n_h+n_{sq}})^{2\log^*|\mathscr{X}|} \cdot \left(\frac{\tau^2}{3}\right)^{2(\log^*|\mathscr{X}|)^2} \right)$

Algorithm 5. $\texttt{rMedian}(\vec{s})$
Input: $\vec{s}$ - a sample of n elements drawn i.i.d. from D
Parameters: ρ - target replicability parameter d - specifies domain size $|\mathscr{X}| = 2^d$ τ - target accuracy of median δ - target failure probability
Output: a τ-approximate median of D

```
1:  if $d = 1$ then
2:      Let $\phi_0(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & o/w \end{cases}$
3:      $p_0 \leftarrow \texttt{rSTAT}_{\rho_0,\tau/2,\phi_0}(\vec{s})$                // Base case
4:      if $p_0 \geq 1/2 - \tau/2$ then
5:          return 0
6:      else
7:          return 1
8:  Break $\vec{s}$ into $|\vec{s}|/n_m$ subsamples
9:  Run $\texttt{Median}_{\tau,\delta_0}$ on each subsample to generate a new sample $\vec{m}$ of $\tau$-approximate medians of
    $D_d$
10: Pair up elements $\vec{m}_{2i}$ and $\vec{m}_{2i-1}$, for $i \in \{1, \cdots, |\vec{m}|/2\}$
11: For each pair $(\vec{m}_{2i}, \vec{m}_{2i-1})$, let $l_i$ denote the longest prefix on which they agree
12: Let $\vec{s}_{rm}$ denote the multiset of $l_i$'s
13: $\ell \leftarrow \texttt{rMedian}_{\rho,\lceil \log d \rceil,\tau,\delta}(\vec{s}_{rm})$
14: $\vec{s}_{h0}, \vec{s}_{h1} \leftarrow n_h$ new examples from $\vec{m}$ each
15: $\vec{s}_\ell \leftarrow \{x_{|\ell} : x \in \vec{s}_{h0}\}$                // $\vec{s}_\ell$ is the set $\vec{s}_{h0}$ projected onto length $\ell$ prefixes
16: $V \leftarrow \texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}(\vec{s}_\ell)$, for $v = 5/16 + \tau$, $\varepsilon = 1/16$
17: if $\ell < d$ then
18:     $\vec{s}_{\ell+1} \leftarrow \{x_{|\ell+1} : x \in \vec{s}_{h1}\}$
19:     $V \leftarrow V \cup \texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}(\vec{s}_{\ell+1})$        // Find vertices at level $\ell$ and $\ell + 1$ with weight $\geq 1/4$
20: else
21:     return the first element of $V$
22: for $v \in V$ do
23:     Let $\phi_v(x) = \begin{cases} 1 & \text{if } x_{||v|} = v \\ 0 & \text{o/w} \end{cases}$
24:     $\vec{s}_q \leftarrow n_{sq}$ new examples from $\vec{m}$
25:     $p_v \leftarrow \texttt{rSTAT}_{\rho_0,\tau,\phi_v}(\vec{s}_q)$,                // Query $D^m$ for probability $x \leq v||1 \cdots 1$
26:     if $1/4 \leq p_v \leq 3/4$ then
27:         $s \leftarrow v$                                      // Find length $\ell$ prefix of weight in $[1/4 - \tau, 3/4 + \tau]$
28: $s_0 = s||0 \cdots 0$                                     // $s_0$ is the prefix $s$ padded with 0's to length $d$
29: $s_1 = s||1 \cdots 1$                                     // $s_1$ is the prefix $s$ padded with 1's to length $d$
30:
31: Let $\phi_{s_0}(x) = \begin{cases} 1 & \text{if } x \leq s_0 \\ 0 & \text{o/w} \end{cases}$
32: $\vec{s}_{s_0} \leftarrow n_m$ new examples from $\vec{m}$
33: $p_{s_0} \leftarrow \texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}(\vec{s}_{s_0})$
34: if $p_{s_0} \geq 1/8 - 2\tau$ then
35:     return $s_0$
36: else
37:     return $s_1$
```

Lemma 3.4.4 (Termination)**.** *Algorithm 5 terminates after $T = \log^* |\mathscr{X}|$ recursive calls.*

Proof. Algorithm 5 reaches its base case when invoked with parameter $d = 1$. At each successive recursive call (Line 13), the domain size 2^d is reduced to $2^{\lceil \log d \rceil} < 2d$, and so $d = 1$ after no more than $T = \log^* |\mathscr{X}|$ recursive calls. □

Lemma 3.4.5 (Sample Complexity)**.** *Let $\tau, \delta, \rho \in [0,1]$. Let D be a distribution over $\mathscr{X}$, with $|\mathscr{X}| = 2^d$. Then* $\texttt{rMedian}_{\rho,d,\tau,\delta}$ *has sample complexity*

$$n \in O\left(\left(\frac{1}{\tau^2(\rho-\delta)^2} \right) \cdot \left(\frac{3\log(2/\delta_0)}{\tau^2} \right)^{\log^* |\mathscr{X}|} \right)$$

Proof. We begin by arguing that, for $d > 1$, $\texttt{rMedian}_{\rho,d,\tau,\delta}$ has sample complexity $n_m(2n_{\lceil \log d \rceil} + n_h + 4n_{sq})$. First, observe that Line 9 of Algorithm 5 is the only line that uses the sample $\vec{s}$ directly, and it uses $\vec{s}$ to generate a sample $\vec{m}$ of size $|\vec{s}|/n_m$ from D^m. The remaining subroutines use subsamples from $\vec{m}$. Therefore, if the sample complexity of the remaining subroutines is bounded by some value N, then $\texttt{rMedian}_{\rho,d,\tau,\delta}$ will have sample complexity Nn_m. We now consider the sequence of subroutines and their respective complexities.

1. Line 13: $\texttt{rMedian}_{\rho,\lceil \log d \rceil,\tau,\delta}$ requires $n_{\lceil \log d \rceil}$ examples from $D_{\lceil \log d \rceil}$. Line 11 generates an example from $D_{\lceil \log d \rceil}$ from 2 examples from D^m, and so the call to $\texttt{rMedian}_{\rho,\lceil \log d \rceil,\tau,\delta}$ at Line 13 contributes $2n_{\lceil \log d \rceil}$ to the sample complexity.

2. Line 16 and Line 19: $\texttt{rHeavyHitters}_{\rho_0,\nu,\varepsilon}$ requires n_h examples from D^m

3. Line 22: the at most 3 calls to $\texttt{rSTAT}_{\rho_0,\tau,\phi_v}$ require $3n_{sq}$ examples from D^m

4. Line 33: $\texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}$ requires n_{sq} examples from D^m

Therefore $\texttt{rMedian}_{\rho,d,\tau,\delta}$ uses $n = n_m(2n_{\lceil \log d \rceil} + 2n_h + 4n_{sq})$ examples from D.

In the base case, the entire contribution to the sample complexity comes from the call to $\texttt{Median}_{\tau,\delta_0}$, which requires n_m examples from D_1. Unrolling the recursion, we have

$$
\begin{aligned}
n &\in O\left((2n_m)^{\log^*|\mathscr{X}|}(n_h+n_{sq})\right)\\
&\in \tilde{O}\left(\left(\frac{1}{\tau^2(\rho-\delta)^2}\right)\cdot\left(\frac{3\log(2/\delta_0)}{\tau^2}\right)^{\log^*|\mathscr{X}|}\right).
\end{aligned}
$$

□

Lemma 3.4.6 (Accuracy)**.** *Let $\rho, \tau, \delta \in [0,1]$ and let n denote the sample complexity proved in Lemma 3.4.5. Let $\vec{s}$ be a sample of elements drawn i.i.d. from D such that $|\vec{s}| \in \Omega(n)$. Then* `rMedian`$(\vec{s})$ *returns a τ-approximate median of D except with probability δ.*

Proof. First, we prove that `rMedian`$(\vec{s})$ returns a τ-approximate median of D, conditioned on the success of all recursive calls and subroutines. We proceed inductively. In the base case we have that $|\mathscr{X}| = 2$, and therefore at least one of the two elements in $\mathscr{X}$ must be a τ-approximate median. The statistical query performed in line 6 of Algorithm 5 uses sample $\vec{s}$ to estimate the fraction of D_1 supported on 0, to within tolerance $\tau/2$, so long as $|\vec{s}| \geq n_m$. This holds from Lemma 3.4.5, and so a τ-approximate median for D_1 is returned in the base case.

It remains to show that if a τ-approximate median for $D_{\lceil \log d \rceil}$ is returned at Line 13 of Algorithm 5, that a τ-approximate median for D is returned. We first note that, except with probability $\delta_0 \cdot |\vec{s}|/n_m$, all elements of $\vec{m}$ are τ-approximate medians of D. To generate the sample supplied to `rMedian` at Line 13, we pair up the elements of $\vec{m}$ to obtain the $|\vec{s}|/(2n_m)$ l_i, which denote the longest prefix on which a pair of elements from $\vec{m}$ agree. Then $\vec{s}_{rm}$ constitutes a sample of size $n_{\lceil \log d \rceil}$ drawn i.i.d. from $D_{\lceil \log d \rceil}$ and by inductive assumption the call to `rMedian` at Line 13 returns a τ-approximate median of $D_{\lceil \log d \rceil}$. Therefore we have that $\mathbf{Pr}_{x_1,x_2 \sim D^m}[x_{1|\ell} = x_{2|\ell}] \geq 1/2 - \tau$ and $\mathbf{Pr}_{x_1,x_2 \sim D^m}[x_{1|\ell+1} = x_{2|\ell+1}] < 1/2 + \tau$. It follows that there must exist a prefix s of length ℓ such that $\mathbf{Pr}_{x \sim D^m}[x_{|\ell} = s] \geq 1/4$.

If $\ell = d$, then $x_{|\ell} = x$, and so any prefix s such that $\mathbf{Pr}_{x \sim D^m}[x_{|\ell} = s] \geq 1/4$ is a 3/8-median of D^m and therefore a τ-median of D. In this case s is returned at Line 21.

For the remainder of the proof, we assume $\ell < d$. We argue that there must exist a prefix s of length ℓ or $\ell+1$ for which $1/4 \leq \mathbf{Pr}_{x \sim D^m}[x_{||s|} = s] \leq 3/4$. We already have that there exists a prefix s of length ℓ such that $\mathbf{Pr}_{x \sim D^m}[x_\ell = s] \geq 1/4$. Suppose that $\mathbf{Pr}_{x \sim D^m}[x_{|\ell} = s] > 3/4$. Now suppose that one of $s||0$ or $s||1$ had probability greater than 3/4 under D^m. Then it must be the case that $\mathbf{Pr}_{x_1,x_2 \sim D^m}[x_{1|\ell+1} = x_{2|\ell+1}] > 9/16$, and so $\mathbf{Pr}_{\ell' \sim D_{\lceil \log d \rceil}}[\ell' \leq \ell] < 1 - 9/16 = 7/16$, contradicting that ℓ is a τ-approximate median of $D_{\lceil \log d \rceil}$. So both $s||0$ and $s||1$ must have probability less than 3/4 under D^m. Because s has probability at least 3/4, it follows that at least one of $s||0$ and $s||1$ must have probability at least 1/4 under D^m, and so we have that there exists a prefix s' of length $\ell+1$ such that $1/4 \leq \mathbf{Pr}_{x \sim D^m}[x_{|\ell+1} = s'] \leq 3/4$.

Now that we have the existence of such a prefix, we will argue that when the loop of Line 22 terminates, s is a prefix satisfying

$$1/4 - \tau \leq \mathbf{Pr}_{x_1 \sim D^m}[x_{1|\ell} = s] \leq 3/4 + \tau.$$

Observe that the calls to `rHeavyHitters` at Line 16 and Line 19 identify a common prefix s such that $\mathbf{Pr}_{x_1 \sim D^m}[x_{1|\ell} = s] \geq 1/4$. This follows from taking $v = 5/16$, $\varepsilon = 1/16$, and the fact that the sample $\vec{s}_\ell$ and $\vec{s}_{\ell+1}$ constitute i.i.d. samples of size n_h drawn from $D^m_{|\ell}$ and $D^m_{|\ell+1}$ respectively (where we use $D^m_{|\ell}$ to indicate the distribution induced by sampling from D^m and returning only the first ℓ bits). Then we have from the proof of Lemma 3.3.3 that all $v - \varepsilon = 1/4$-heavy hitters from $D^m_{|\ell}$ and $D^m_{|\ell+1}$ are contained in the set V. The loop beginning at Line 22 will use replicable statistical queries to estimate the probability of each $v \in V$ under $D^m_{||v|}$. If the estimated probability $p_v \in [1/4, 3/4]$, then v is stored in s, and so the last such string visited by the loop is the value of s upon termination.

Now we show that if $s_0 = s||0 \cdots 0$ is returned at Line 35, then it is a τ-approximate median of D, otherwise $s_1 = s||1 \cdots 1$ is a τ-approximate median. Conceptually, we partition the domain $\mathscr{X}$ into three sets:

1. $C_{s_0} = \{x \in \mathscr{X} : x < s_0\}$
2. $C_s = \{x \in \mathscr{X} : s_0 \leq x \leq s_1\}$
3. $C_{s_1} = \{x \in \mathscr{X} : x > s_1\}$

Because s satisfies $1/4 - \tau \leq \mathbf{Pr}_{x \sim D^m}[x_{||s|} = s] \leq 3/4 + \tau$, it must be the case that D^m assigns probability mass at least $1/4 - \tau$ to the union $C_{s_0} \cup C_{s_1}$. Then it holds that at least one of C_{s_0} and C_{s_1} is assigned probability mass at least $1/8 - \tau/2$. The statistical query made at Line 33 estimates the probability mass assigned to C_{s_0} by D^m to within tolerance τ, so if s_0 is returned, it holds that $\mathbf{Pr}_{x \sim D^m}[x < s_0] \geq 1/8 - 3\tau$. Because we know $\mathbf{Pr}_{x \sim D^m}[x \in C_s] \geq 1/4 - \tau$, we then also have that $\mathbf{Pr}_{x \sim D^m}[x \geq s_0] \geq 1/4 - \tau$. Because D^m is a distribution over τ-approximate medians of D, we have that s_0 is a τ-approximate median of D as desired. If s_0 is not returned, then it must be the case that $\mathbf{Pr}_{x \sim D^m}[x > s_1] \geq 1/8 - 3\tau$, and a similar argument shows that s_1 must be a τ-approximate median of D.

Finally, we argue that all recursive calls and subroutines are successful, except with probability δ. Failures can occur exclusively at the following calls.

- Line 9: the $\log^* |\mathscr{X}| \cdot |\vec{s}|/(n_m)$ calls to $\texttt{Median}_{\tau,\delta_0}$
- Line 13: the $\log^* |\mathscr{X}|$ recursive calls to $\texttt{rMedian}_{\rho,\lceil \log d \rceil,\tau,\delta}$
- Line 16 and Line 19: the $2\log^* |\mathscr{X}|$ calls to $\texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}$
- Line 22: the (at most) $4\log^* |\mathscr{X}|$ calls to $\texttt{rSTAT}_{\rho_0,\tau,\phi_v}$
- Line 33: the $\log^* |\mathscr{X}|$ calls to $\texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}$

Calls to $\texttt{Median}_{\tau,\delta_0}$ dominate the total failure probability, and so taking $\delta_0 \in O(\frac{\delta}{|\vec{s}| \log^* |\mathscr{X}|})$ suffices to achieve failure probability δ. □

Lemma 3.4.7 (Replicability). *Let $\rho, \tau, \delta \in [0,1]$ and let n denote the sample complexity proved in Lemma 3.4.5. Let $\vec{s}$ be a sample of $O(n)$ elements drawn i.i.d. from D. Then $\texttt{rMedian}_{\rho,d,\tau,\delta}$ is ρ-replicable.*

Proof. We prove the lemma by inductive argument. First, we observe that replicability of the value returned in the base case depends only on the value $p_0 \leftarrow \texttt{rSTAT}_{\rho_0,\tau/2,\phi_0}(\vec{s})$ in Line 3. Therefore, replicability in the base case follows from the ρ_0-replicability of $\texttt{rSTAT}_{\rho_0,\tau/2,\phi_0}$.

We now argue that if the $i+1$th recursive call is ρ-replicable, that the ith recursive call is $(\rho + 5\rho_0)$-replicable.

Two parallel executions of the ith level of recursion, given samples $\vec{s}_1$ and $\vec{s}_2$ drawn i.i.d. from the same distribution D, will produce the same output so long as the following values are the same:

1. $\ell \leftarrow \texttt{rMedian}_{\rho,d,\tau,\delta}(\vec{s}_{rm})$ at Line 13
2. $V \leftarrow \texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}(\vec{s}_\ell)$ at Line 16
3. $V \leftarrow V \cup \texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}(\vec{s}_{\ell+1})$ at Line 19
4. $s \leftarrow \texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}(\vec{s}_{meds})$ when the loop at Line 22 terminates
5. $p_{s_0} \leftarrow \texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}(\vec{s}_{s_0})$ at Line 33

produce the same value. We have that 1 holds by inductive assumption.

Conditioning on 1, the calls to $\texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}$ are made on samples drawn i.i.d. from the same distribution, and so the ρ_0-replicability of $\texttt{rHeavyHitters}_{\rho_0,v,\varepsilon}$ guarantees that V contains the same list of heavy-hitters in both runs except with probability 2ρ.

Conditioning on both 1 and 2, it follows that the loop at Line 22 iterates over the same strings V, and so both runs make the same sequence of statistical queries $\texttt{rSTAT}_{\tau,\rho_0,\phi_v}$. From conditioning on 2, and the values of v and ε, we have that $|V| \leq 3$, and so the ρ_0-replicability of

$\texttt{rSTAT}_{\tau,\rho_0,\phi_{s_0}}$ gives us that sequence of values $p_v \leftarrow \texttt{rSTAT}_{\rho_0,\tau,\phi_v}(\vec{s}_q)$ is the same in both runs, except with probability 3_{ρ_0}.

Finally, conditioning on 1, 2, and 3, the values of s_0 and s_1 are the same across both runs, and so the same statistical query $\texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}$ is made in both runs. Whether s_0 or s_1 is returned depends only on the value $r_{s_0} \leftarrow \texttt{rSTAT}_{\tau,\rho_0,\phi_{s_0}}(\vec{s}_{s_0})$, and so the ρ_0-replicability of $\texttt{rSTAT}_{\rho_0,\tau,\phi_{s_0}}$ gives us that the same string is returned by both executions. A union bound over all failures of replicability then gives us that the ith recursive call will be $(\rho + 6\rho_0)$-replicable.

From Lemma 3.4.4, we have that no more than $T = \log^* |\mathscr{X}|$ recursive calls are made by the algorithm. Therefore $\texttt{rMedian}_{\rho,d,\tau,\delta}$ is replicable with parameter $\rho_0 + 5T\rho_0 \leq 6\rho_0 \log^* |\mathscr{X}| = \rho$. □

Theorem 3.4.2 then follows as a corollary of Lemma 3.4.5, Lemma 3.4.6, and Lemma 3.4.7.

Theorem 3.4.2 (Replicable Median). *Let $\tau, \rho \in [0,1]$ and let $\delta = \rho/2$. Let D be a distribution over $\mathscr{X}$, where $|\mathscr{X}| = 2^d$. Then $\texttt{rMedian}_{\rho,d,\tau,\delta}$ (Algorithm 5) is ρ-replicable, outputs a τ-approximate median of D with all but probability δ, and has sample complexity*

$$n \in \tilde{O}\left(\left(\frac{1}{\tau^2(\rho-\delta)^2} \right) \cdot \left(\frac{3}{\tau^2} \right)^{\log^* |\mathscr{X}|} \right)$$

3.5 Learning Halfspaces

In Section 3.2, we saw how combining a concentration bound with a randomized rounding technique yielded a replicable algorithm. Specifically, given a statistical query algorithm with an accuracy guarantee (with high probability) on the 1-dimensional space $[0,1]$, we can construct a replicable statistical query algorithm using randomized rounding. By sacrificing a small amount of accuracy, our replicable statistical query algorithm can decide on a canonical return value in $[0,1]$.

In this section, we extend this argument from $\mathbb{R}$ to $\mathbb{R}^d$, by way of an interesting application of a randomized rounding technique from the study of foams [KORW12]. Algorithm 1 in [KORW12] probabilistically constructs a tiling of $\mathbb{R}^d$ such that every point is rounded to a nearby integer lattice point. This tiling has an additional property that the probability that two points are not rounded to the same point by a constructed tiling is at most linear in their l_2 distance. In the usual PAC-learning setting, there is a simple weak learning algorithm for halfspaces that takes examples $(\vec{x}_i, y_i) \in \mathscr{X} \times \{\pm 1\}$, normalizes them, and returns the halfspace defined by vector $\sum_i \vec{x}_i \cdot y_i$ [Ser02]. We show a concentration bound on the sum of normalized vectors from a distribution, and then argue that all vectors within the concentration bound are reasonable hypotheses with non-negligible advantage. The combination of this concentration bound and the foam-based rounding scheme yields a replicable halfspace weak learner `rHalfspaceWkL`.

However, constructing this foam-based rounding scheme takes expected time that is exponential in the dimension d. We give an alternative rounding scheme that randomly translates the integer lattice and rounds points to their nearest translated integer lattice point. This construction yields another replicable halfspace weak learner $\texttt{rHalfspaceWkL}^{\text{box}}$ with roughly an additional factor of d in the sample complexity, but with polynomial runtime. In Section 3.6, we show how to combine these replicable weak learners with a replicable boosting algorithm, yielding an polynomial-time replicable strong learner for halfspaces.

3.5.1 Replicable Halfspace Weak Learner: An Overview

Let D be a distribution over $\mathbb{R}^d$, and let EX be an example oracle for D and f, where $f : \mathbb{R}^d \to \{\pm 1\}$ is a halfspace that goes through the origin. Let $||\vec{x}||$ denote the l_2 norm of vector $\vec{x}$. We assume that D satisfies a (worst-case) margin assumption with respect to f.

Definition 3.5.1. *[Margin] Let D be a distribution over $\mathbb{R}^d$. We say D has margin τ_f with respect to halfspace $f(\vec{x}) \overset{\text{def}}{=} \operatorname{sign}(\vec{w} \cdot \vec{x})$ if $\frac{\vec{x} \cdot f(\vec{x})}{||\vec{x}||} \cdot \frac{\vec{w}}{||\vec{w}||} \geq \tau_f$ for all $x \in supp(D)$. Additionally, we say D has (worst-case) margin τ if $\tau = \sup_f \tau_f$.*

Our replicable halfspace weak learner `rHalfspaceWkL` uses its input to compute an empirical estimation $\vec{z}$ of the expected vector $\mathbb{E}_{\vec{x}\sim D}[\vec{x}\cdot f(x)]$. Then, `rHalfspaceWkL` uses its randomness to construct a rounding scheme R via Algorithm `ConstructFoams`. R is used to round our (rescaled) empirical estimation $\vec{z}$, and the resulting vector defines the returned halfspace. The algorithm relies on the margin assumption to ensure that the weak learner's returned hypothesis is positively correlated with the true halfspace f.[2]

Algorithm 6. `rHalfspaceWkL`$(\vec{s};r)$ a ρ-replicable halfspace weak learner using foams
Parameters: ρ - desired replicability
d - dimension of halfspace
τ - assumed margin
a - a constant, $a = .05$
Input: A sample $\vec{s}$ of $m = \left(\frac{896\sqrt{d}}{\tau^2\rho}\right)^{1/(1/2-a)}$ examples $(\vec{x}_i, y_i)$ drawn i.i.d. from distribution D
Output: A hypothesis with advantage $\tau/4$ on D against f

```
$k \leftarrow \frac{1}{m}\frac{8\sqrt{d}}{\tau^2} = 8\cdot\left(\frac{\rho}{896}\left(\frac{\tau^2}{\sqrt{d}}\right)^{1/2+a}\right)^{1/(1/2-a)}$    // Scaling factor
$\vec{z} \leftarrow \sum_S \frac{\vec{x}_i}{\|\vec{x}_i\|}\cdot y_i$
$R \leftarrow_r$ ConstructFoams$(d)$ (Algorithm 7)    // Rounding scheme $R : \mathbb{R}^d \to \mathbb{Z}^d$
$\vec{w} \leftarrow R(k\cdot\vec{z})$
return Hypothesis $h(\vec{x}) \overset{\text{def}}{=} \frac{\vec{x}}{\|\vec{x}\|}\cdot\frac{\vec{w}}{\|\vec{w}\|}$
```

The subroutine `ConstructFoams` previously appeared as Algorithm 1 in [KORW12]. For completeness, we include a description below (Algorithm 7).

[2]The parameter a is a constant, but we leave it in variable form for convenience in the analysis; we choose $a = .05$ in this proof for clarity of presentation, but one could optimize the choice of a to yield a slightly better sample complexity.

Algorithm 7. `ConstructFoams`(d) // Algorithm 1 in [KORW12]
Input: dimension d
Output: rounding scheme $R : \mathbb{R}^d \to \mathbb{Z}^d$

```
Let $f : [0,1]^d \to \mathbb{R}$ s.t. $f(x_1,\ldots,x_d) \overset{\text{def}}{=} \prod_{i=1}^{d}(2\sin^2(\pi x_i))$
Let all points in $\mathbb{R}^d$ be unassigned
for stage $t = 1,2,\ldots$ until all points are assigned do
    Uniformly at random sample $Z_t, H_t$ from $[0,1)^d \times (0,2^d)$.
    Let droplet $D_i$ be the set of points $\{x | x \in -Z_t + [0,1)^d, f(x+Z_t) > H_t\}$.
    Let $R$ map all currently unassigned points in $D_i$ to $(0,0,\ldots,0)$ and extend this assignment
    periodically to all integer lattice points.
return $R$
```

The following is the main result of this section.

Theorem 3.5.2. *Let D be a distribution over $\mathbb{R}^d$, and let $f : \mathbb{R}^d \to \{\pm 1\}$ be a halfspace with margin τ in D. Then* `rHalfspaceWkL`$(\vec{s}; r)$ *is a $(\rho, \tau/4, \rho/4)$-weak learner for halfspaces. That is, Algorithm 6 ρ-replicably returns a hypothesis h such that, with probability at least $1-\rho/2$, $\frac{1}{2}\mathbb{E}_{\vec{x}\sim D} h(\vec{x}) f(\vec{x}) \geq \tau/4$, using a sample of size $m = \left(\frac{896\sqrt{d}}{\tau^2\rho}\right)^{20/9}$.*

Proof. **Correctness (Advantage):** We argue correctness in two parts. First, we show the expected weighted vector $\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}\right]$ defines a halfspace with good advantage (see Lemma 3.5.8), following the arguments presented in Theorem 3 of [Ser02]. Then, we argue that rounding the empirical weighted vector $\vec{z}$ in Algorithm 6 only slightly rotates the halfspace. By bounding the possible loss in advantage in terms of the amount of rotation (Lemma 3.5.9), we argue that the rounded halfspace $\vec{w}/|\vec{w}|$ also has sizable advantage.

By Lemma 3.5.8, the expected weighted vector $\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}\right]$ has advantage $\tau/2$ on D and f. The martingale-based concentration bound in Corollary 3.5.11 implies that the distance between

$\vec{z}$ and $\mathbb{E}[\vec{z}] = m \cdot \mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}\right]$ is less than $4m^{1/2+a}$ with probability at least $1-e^{-m^{2a}/2}$ for any $a \in (0,1/2)$ (chosen later). Then, the vector is scaled by k and rounded. Any rounding scheme R randomly generated by `ConstructFoams` always rounds its input to a point within distance $\sqrt{d}$ (Observation 3.5.4). Combining, the total distance between vectors $\frac{\vec{w}}{k\cdot\|\mathbb{E}[\vec{z}]\|}$ and $\frac{\mathbb{E}[\vec{z}]}{\|\mathbb{E}[\vec{z}]\|}$ is at most

$$\frac{4m^{1/2+a}+\sqrt{d}/k}{\|\mathbb{E}[\vec{z}]\|}.$$

As D has margin τ with respect to f, for all $\vec{x} \in \text{supp}(D)$, $\frac{\vec{x}}{\|\vec{x}\|}\cdot f(\vec{x})$ has length at least τ in the direction of the expected weighted vector $\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}\right]$. Thus, $\|\mathbb{E}[\vec{z}]\| \geq \tau m$, and the above quantity is at most $\frac{4m^{1/2+a}+\sqrt{d}/k}{\tau m}$. Simplifying, $\frac{4m^{1/2+a}}{\tau m} = \frac{4}{\tau m^{1/2-a}} = \frac{4\tau}{896}\frac{\rho}{\sqrt{d}} < \tau/8$ and $\frac{\sqrt{d}/k}{\tau m} = \frac{\sqrt{d}}{\tau}\frac{\tau^2}{8\sqrt{d}} = \tau/8$. By applying Lemma 3.5.9 with $\theta = \tau/8+\tau/8 = \tau/4$, we can conclude that h has advantage at least $\tau/2-\tau/4 = \tau/4$, as desired.[3]

Replicability: Let $\vec{z}_1$ and $\vec{z}_2$ denote the empirical sums of vectors $\vec{x}_i y_i$ from two separate runs of `rHalfspaceWkL`. It suffices to show that the rounding scheme R constructed by `ConstructFoams` rounds $k\cdot\vec{z}_1$ and $k\cdot\vec{z}_2$ to the same vector $\vec{w}$ with high probability. The distance between $\vec{z}_1$ and $\vec{z}_2$ is at most $2\cdot 4m^{1/2+a}$ with probability at least $1-2e^{-m^{2a}/2}$, by Corollary 3.5.11, the triangle inequality, and a union bound. After scaling by k, this distance is at most $8km^{1/2+a}$. By Lemma 3.5.3, the probability that R does not round $k\cdot\vec{z}_1$ and $k\cdot\vec{z}_2$ to same integer lattice point is at most $7\cdot 8km^{1/2+a}$. Altogether, the replicability parameter is at most

$$2e^{-m^{2a}/2}+56km^{1/2+a}.$$

[3] A dedicated reader may notice that the scaling factor k is subconstant. A possible error may arise if the scaling factor is so small that the halfspace vector $\vec{z}\cdot k$ gets rounded to 0 by the rounding function R (constructed by `ConstructFoams`). Fortunately, with our choice of parameters, this turns out to not be an issue. The empirical vector sum $\vec{z}$ has norm at least $\tau\cdot m$, where τ is the margin size and m is the sample complexity. As we have chosen scaling factor k such that $m\cdot k = 8\sqrt{d}/\tau^2$, the input given to R has norm at least $8\sqrt{d}/\tau$. Every rounding function R constructed by `ConstructFoams` rounds its input to a point at distance at most $\sqrt{d}$ away (Observation 3.5.4), so we can be sure that R never rounds our vector to the zero vector.

The second term satisfies $56km^{1/2+a} = 448 \cdot \frac{\rho}{896} \cdot 1 = \rho/2$, and the first term $2e^{-m^{2a}/2} \le \rho/2$ when $m \ge (2\ln(4/\rho))^{1/(2a)}$. As long as a is chosen such that $m = \left(\frac{896\sqrt{d}}{\tau^2\rho}\right)^{1/(1/2-a)} \ge (2\ln(4/\rho))^{1/(2a)}$, the algorithm is ρ-replicable. This occurs if $\left(\frac{896}{\rho}\right)^{2a/(1/2-a)} \ge 2\ln(4/\rho)$, which is true for all values of $\rho \in (0,1)$ when $a = .05$.[4]

Failure rate: The algorithm succeeds when the martingale concentration bound holds. So, the failure probability of `rHalfspaceWkL` is at most $e^{-m^{2a}/2} \le \rho/4$.

Sample complexity: Plugging in $a = .05$ in the expression $m = \left(\frac{896\sqrt{d}}{\tau^2\rho}\right)^{1/(1/2-a)}$ yields the conclusion. □

3.5.2 Replicable Weak Halfspace Learner – Definitions and Lemmas

Foams-Based Rounding Scheme from [KORW12]

For completeness, we restate relevant results from [KORW12] for our construction.

Lemma 3.5.3 (Combining Theorem 1 and Theorem 3 of [KORW12]). *Let $R : \mathbb{R}^d \to \mathbb{Z}^d$ be the randomized rounding scheme constructed by Algorithm 7 (Algorithm 1 in [KORW12]). Let $x, y \in \mathbb{R}^d$, and let $\varepsilon \overset{\text{def}}{=} d_{l_2}(x,y)$. Then $\mathbf{Pr}[R(x) = R(y)] \ge 1 - O(\varepsilon)$, where the probability is over the randomness used in the algorithm.*

Proof. Theorem 3 of [KORW12] states that $f(\vec{x}) = \Pi_{i=1}^d (2\sin^2(\pi x_i))$ is a *proper* density function and $\int_{[0,1)^d} |\langle \nabla f, u\rangle| \le 2\pi$ for all unit vectors u. Theorem 1 of [KORW12] states the following. Let f be a proper density function, and points $x, y \in \mathbb{R}^d$ such that $y = x + \varepsilon \cdot u$, where $\varepsilon > 0$ and u is a unit vector. Let N denote the number of times the line segment $\overline{xy}$ crosses the boundary between different droplets (potentially mapping to the same integer lattice point) in an execution of `ConstructFoams`. Then $\mathbb{E}[N] \approx \varepsilon \cdot \int_{[0,1)^d]} |\langle \nabla f, u\rangle|$, where the $\approx$ notation is hiding a $W\varepsilon^2$ term, where $W > 0$ is a universal constant depending only on f. The authors refine this statement ([KORW12], page 24) to show that the $W\varepsilon^2$ term can be made arbitrarily small. Combining, $\mathbb{E}[N] \le 2\pi\varepsilon + W\varepsilon^2 < 6.3\varepsilon$. By Markov's inequality, $\mathbf{Pr}[N = 0] < 1 - 6.3\varepsilon$. □

[4]The constant a can be improved slightly if a is chosen as a function of ρ.

Observation 3.5.4. `ConstructFoams` *always outputs a rounding scheme R with the following property: the (l_2) distance between any vector $\vec{v} \in \mathbb{R}^d$ and $R(\vec{v})$ is at most $\sqrt{d}$.*

This follows from noticing that R maps each coordinate of $\vec{v}$ to its floor or ceiling.

Theorem 3.5.5 (Runtime of `ConstructFoams`; [KORW12], page 25)**.** *There are universal constants $1 < c < C$ such that Algorithm 7, when run with $f(\vec{x}) = \Pi_{i=1}^{d}(2\sin^2(\pi x_i))$, takes between c^d and C^d stages except with probability at most c^{-d}.*

Weak Learning Definitions

Definition 3.5.6 (Weak Learning Algorithm (in the Filtering Model))**.** *Let $\mathscr{C}$ be a concept class of functions from domain $\mathscr{X}$ to $\{\pm 1\}$, and let $f \in \mathscr{C}$. Let D be a distribution over $\mathscr{X}$. Let* `WkL` *be an algorithm that takes as input a labeled sample $S = \{(x_i, f(x_i))\}_m$ drawn i.i.d. from D, and outputs a hypothesis $h : \mathscr{X} \to [-1,1]$. Then* `WkL` *is a (γ, δ)-weak learner for $\mathscr{C}$ with sample complexity m if, for all f, D, with probability at least $1-\delta$,* `WkL`(S) *outputs a hypothesis $h : \mathscr{X} \to [-1,1]$ such that $\mathbb{E}_{x \sim D} f(x)h(x) \geq 2\gamma$, where S is a sample of size $|S| = m$ drawn i.i.d. from D.*

We say a (γ, δ)-weak learner has *advantage* γ. Equivalently, if a hypothesis h satisfies $\frac{1}{2}\mathbb{E}_{x \sim D} f(x)h(x) \geq \gamma$, then we say h has advantage γ (on D and f).

Definition 3.5.7 (Replicable Weak Learning Algorithm)**.** *Algorithm* `rWkL` *is a (ρ, γ, δ)-weak learner if* `rWkL` *is ρ-replicable and a (γ, δ)-weak learner.*

Halfspaces and Their Advantage

Definition 3.5.1. *[Margin] Let D be a distribution over $\mathbb{R}^d$. We say D has margin τ_f with respect to halfspace $f(\vec{x}) \overset{\text{def}}{=} \text{sign}(\vec{w} \cdot \vec{x})$ if $\frac{\vec{x} \cdot f(\vec{x})}{\|\vec{x}\|} \cdot \frac{\vec{w}}{\|\vec{w}\|} \geq \tau_f$ for all $x \in supp(D)$. Additionally, we say D has (worst-case) margin τ if $\tau = \sup_f \tau_f$.*

Lemma 3.5.8 (Advantage of Expected Weighted Vector Hypothesis [Ser02])**.** *Let $f(\vec{x}) \overset{\text{def}}{=} \text{sign}(\vec{w} \cdot \vec{x})$ be a halfspace, and let D be a distribution over $\mathbb{R}^d$ with margin τ with respect to f. Let $\vec{z} = \mathbb{E}_{\vec{v} \sim D}\left[\frac{\vec{v}}{\|\vec{v}\|} f(\vec{v})\right]$. Then the hypothesis $h_{\vec{z}}(\vec{x}) = \frac{\vec{x}}{\|\vec{x}\|} \cdot \frac{\vec{z}}{\|\vec{z}\|}$ has advantage at least $\tau/2$.*

Proof. The advantage of $h_{\vec{z}}$ is $\frac{1}{2}\mathbb{E}_{\vec{x}\sim D}[h_{\vec{z}}(\vec{x})f(\vec{x})] = \frac{1}{2}\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}}{\|\vec{x}\|}\cdot\frac{\vec{z}}{\|\vec{z}\|}\cdot f(\vec{x})\right] = \frac{\vec{z}\cdot\vec{z}}{2\|\vec{z}\|} = \frac{\|\vec{z}\|}{2} \geq \frac{\vec{z}\cdot\vec{w}}{2\|\vec{w}\|}$, by the Cauchy-Schwarz inequality. Vector $\vec{z}$ is a convex combination of $\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}$ terms, for $\vec{x}\in\mathrm{supp}(D)$. By the margin assumption, $\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}\cdot\frac{\vec{w}}{\|\vec{w}\|} \geq \tau$ for all $x\in\mathrm{supp}(D)$. Thus, $\frac{\vec{z}\cdot\vec{w}}{2\|\vec{w}\|}\geq\frac{\tau}{2}$.

□

Lemma 3.5.9 (Advantage of Perturbed Halfspaces)**.** *Consider a halfspace defined by unit vector $\vec{w}$, and let $h(\vec{x}) = \frac{\vec{x}}{\|\vec{x}\|}\cdot\vec{w}$. Assume h has advantage γ, i.e. $\frac{1}{2}\mathbb{E}_{x\sim D}f(x)h(x)\geq\gamma$. Let $\vec{u}$ be any vector such that $\|\vec{u}\|\leq\theta$, where $\theta\in[0,\sqrt{3}/2)$. Let perturbed vector $\vec{w'} = \frac{\vec{w}+\vec{u}}{\|\vec{w}+\vec{u}\|}$, and let $h'(\vec{x}) = \frac{\vec{x}}{\|\vec{x}\|}\cdot\vec{w'}$. Then h' has advantage at least $\gamma-\theta$.*

Proof. First, we bound the maximum distance between $\vec{w}$ and $\vec{w'}$. Then, we apply Cauchy-Schwarz to bound the advantage loss. $\vec{w'}$ is constructed by perturbing $\vec{w}$ by a vector $\vec{u}$, and then normalizing to norm 1. $\vec{w'}$ is furthest away from $\vec{w}$ when the vector $\vec{w'}$ is tangent to the ball of radius $\|\vec{u}\|$ around $\vec{w}$. In this case, $\|\vec{w'}-\vec{w}\|^2 = (1-\sqrt{1-\theta^2})^2+\theta^2 = 2-2\sqrt{1-\theta^2}$. Since $\theta<\sqrt{3}/2$, $2-2\sqrt{1-\theta^2}<4\theta^2$. So, $\|\vec{w'}-\vec{w}\|^2<4\theta^2$. The advantage of h' is

$$\begin{aligned}\frac{1}{2}\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}}{\|\vec{x}\|}\cdot\vec{w'}\cdot f(\vec{x})\right] &= \frac{1}{2}\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}}{\|\vec{x}\|}\cdot(\vec{w}+(\vec{w'}-\vec{w}))\cdot f(\vec{x})\right]\\ &= \gamma+\frac{1}{2}(\vec{w'}-\vec{w})\cdot\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}}{\|\vec{x}\|}\cdot f(\vec{x})\right]\end{aligned}$$

By Cauchy-Schwarz, the second term has magnitude at most $\sqrt{4\theta^2\cdot 1}/2$, so the advantage of h' is at least $\gamma-\theta$. □

Concentration Bound on Sum of Normalized Vectors

Let D be a distribution on $\mathbb{R}^n$. Let $\mathbf{v} = \{\mathbf{v_1},\ldots,\mathbf{v_T}\}\in D^T$ be a random sample of T vectors from D with the following properties:

1. $\mathbb{E}_{\mathbf{v}\in D^T}[\sum_{i=1}^T\mathbf{v_i}] - \mathbb{E}_{v\in D}[v] = 0$.

2. $\forall v\in D$, $||v||_2\leq c$.

Lemma 3.5.10. *Let $D, \mathbf{v} \in D^T$ satisfy properties (1) and (2) above, and let $\mathbf{v}^{\leq \mathbf{T}} = \sum_{i=1}^{T} \mathbf{v_i}$. Then for all $\Delta > 0$,*

$$\mathbf{Pr}_{\mathbf{v}}[||\mathbf{v}^{\leq \mathbf{T}}||_2 \geq \sqrt{T}(1+c/2)+\Delta] \leq e^{-\Delta^2/2c^2T}.$$

For a proof, see Appendix 3.9.

Corollary 3.5.11. *Let D be a distribution supported on the unit ball in d dimensions, and let f be a halfspace. Let S be a sample of T examples $(\vec{x}_i, f(\vec{x}_i))$ drawn i.i.d. from D, and let $\vec{z} = \sum_S \vec{x}_i \cdot f(\vec{x}_i)$. Let $a \in (0,1/2)$. Then $\mathbf{Pr}_{S\sim D}\left[\|\vec{z} - T\mathbb{E}_{\vec{v}\sim D}[\vec{v}f(\vec{v})]\| \geq 4T^{1/2+a}\right] \leq e^{-T^{2a}/2}$.*

Proof. In order to have D satisfy the properties (1) and (2) above, we must translate D by the expectation $\mathbb{E}_{\vec{v}\sim D}[\vec{v}f(\vec{v})]$. After this translation, the maximum length of a vector in the support is $c = 2$. Plugging in $\Delta = 2T^{1/2+a}$ and noting $2T^{1/2+a} \geq 2T^{1/2}$ yields the conclusion. $\square$

3.5.3 Coordinate-Based Rounding Scheme

Algorithm `rHalfspaceStL` uses polynomial sample complexity and runs in polynomial time except for subroutine `ConstructFoams`, which runs in expected exponential time in the dimension d (Theorem 3.5.5). Next, we consider a simpler rounding scheme that rounds points coordinate-by-coordinate to a randomly shifted integer lattice. This rounding scheme requires tighter concentration bounds, resulting in approximately another factor of d in the sample complexity. In return, it can be constructed by `ConstructBoxes` and executed in linear time in sample complexity m and dimension d.

Algorithm 8. `ConstructBoxes`(d) // constructs coordinate-based rounding schemes
Input: dimension d
Output: rounding scheme $R : \mathbb{R}^d \to \mathbb{R}^d$

```
  Uniformly at random draw $Z$ from $[0,1)^d$.
  Let box $B$ be the set of points $\{x | \forall i \in [d], x_i \in [-1/2 + z_i, 1/2 + z_i)\}$
  Let $R$ map all points in $B$ to point $Z$ and extend this assignment periodically by integer lattice points
  return $R$
```

Lemma 3.5.12. *Let $R : \mathbb{R}^d \to \mathbb{R}^d$ be the randomized rounding scheme output by* `ConstructBoxes`. *Let $\vec{x}, \vec{y} \in \mathbb{R}^d$, and let $\varepsilon \overset{\text{def}}{=} d_{l_2}(\vec{x}, \vec{y})$. Then $\mathbf{Pr}[R(\vec{x}) = R(\vec{y})] \geq 1 - d\varepsilon$.*

Proof. We bound this probability by a crude l_2 to l_1 distance conversion. If $\vec{x}$ and $\vec{y}$ have l_2 distance ε, then the distance between x_i and y_i is at most ε for all coordinates $i \in [d]$. The i'th coordinate of $\vec{x}$ and $\vec{y}$ are not rounded to the same point with probability $|x_i - y_i|$. By a union bound, $\mathbf{Pr}[R(\vec{x}) = R(\vec{y})] \geq 1 - d\varepsilon$. $\square$

Observation 3.5.13. `ConstructBoxes` *always outputs a rounding scheme R with the following property: the (l_2) distance between any vector $\vec{v} \in \mathbb{R}^d$ and $R(\vec{v})$ is at most $\sqrt{d}/2$.*

This follows from noticing that R maps each coordinate of $\vec{v}$ to value within distance $1/2$.

Replicable Halfspace Weak Learner using Boxes

Algorithm 9. $\texttt{rHalfspaceWkL}^{\text{box}}(\vec{s};r)$ // a ρ-replicable halfspace weak learner
Parameters: desired replicability ρ, dimension d, assumed margin τ, constant $a = .1$
Input: A sample S of $m = \left(\frac{64d^{3/2}}{\tau^2\rho}\right)^{1/(1/2-a)}$ examples $(\vec{x}_i, y_i)$ drawn i.i.d. from distribution D
Output: A hypothesis with advantage $\gamma/4$ on D against f

```
$k \leftarrow \frac{1}{m}\frac{4\sqrt{d}}{\tau^2} = 4 \cdot \left(\frac{\rho \cdot \tau^{1+2a}}{64 \cdot d^{5/4+a/2}}\right)^{1/(1/2-a)}$    // Scaling factor
$\vec{z} \leftarrow \sum_S \frac{\vec{x}_i}{\|\vec{x}_i\|} \cdot y_i$
$R \leftarrow_r$ ConstructBoxes$(d)$ (Algorithm 8)    // Rounding scheme $R : \mathbb{R}^d \to \mathbb{R}^d$
$\vec{w} \leftarrow R(k \cdot \vec{z})$
return Hypothesis $h(\vec{x}) \overset{\text{def}}{=} \frac{\vec{x}}{\|\vec{x}\|} \cdot \frac{\vec{w}}{\|\vec{w}\|}$
```

Theorem 3.5.14. *Let D be a distribution over $\mathbb{R}^d$, and let $f : \mathbb{R}^d \to \{\pm 1\}$ be a halfspace with margin τ in D. Then* $\texttt{rHalfspaceWkL}^{box}(\vec{s};r)$ *is a $(\rho, \tau/4, \rho/4)$-weak learner for halfspaces. That is, Algorithm 9 ρ-replicably returns a hypothesis h such that, with probability at least $1-\rho/2$, $\mathbf{Pr}_{\vec{x}\sim D} h(\vec{x})f(\vec{x}) \geq \tau/4$, using a sample of size $m = \left(\frac{64d^{3/2}}{\tau^2\rho}\right)^{5/2}$.*

Proof. **Correctness (Advantage):** The proof proceeds almost identically to the proof of Theorem 3.5.2. By Lemma 3.5.8, the expected weighted vector $\mathbb{E}_{\vec{x}\sim D}\left[\frac{\vec{x}\cdot f(\vec{x})}{\|\vec{x}\|}\right]$ has advantage $\tau/2$ on D and f. Any rounding scheme R randomly generated by `ConstructBoxes` always rounds its input to a point within distance $\sqrt{d}/2$, so the distance between vectors $\frac{\vec{w}}{k\cdot\|\mathbb{E}[\vec{z}]\|}$ and $\frac{\mathbb{E}[\vec{z}]}{\|\mathbb{E}[\vec{z}]\|}$ is at most

$$\frac{4m^{1/2+a} + \sqrt{d}/2k}{\tau m}.$$

Simplifying, $\frac{4m^{1/2+a}}{\tau m} = \frac{4}{\tau m^{1/2-a}} = \frac{4\tau}{64}\frac{\rho}{d^{3/2}} < \tau/8$ and $\frac{\sqrt{d}/2k}{\tau m} = \frac{\sqrt{d}}{2\tau}\frac{\tau^2}{4\sqrt{d}} = \tau/8$. By applying Lemma 3.5.9 with $\theta = \tau/8 + \tau/8$, we can conclude that h has advantage at least $\tau/2 - (\tau/8 + \tau/8) = \tau/4$, as desired.

Replicability: Let $\vec{z}_1$ and $\vec{z}_2$ denote the empirical sums of vectors $\vec{x}_i y_i$ from two separate runs of `rHalfspaceWkL`. It suffices to show that the rounding scheme R constructed by `ConstructBoxes` rounds $k \cdot \vec{z}_1$ and $k \cdot \vec{z}_2$ to the same vector $\vec{w}$ with high probability. The distance between $\vec{z}_1$ and $\vec{z}_2$ is at most $2 \cdot 4m^{1/2+a}$ with probability at least $1 - 2e^{-m^{2a}/2}$, by Corollary 3.5.11, the triangle inequality, and a union bound. After scaling by k, this distance is at most $8km^{1/2+a}$. By Lemma 3.5.12, the probability that R does not round $k \cdot \vec{z}_1$ and $k \cdot \vec{z}_2$ to same integer lattice point is at most $d \cdot 8km^{1/2+a}$. Altogether, the replicability parameter is at most

$$2e^{-m^{2a}/2} + 8dkm^{1/2+a}.$$

The second term satisfies $8dkm^{1/2+a} = 8d(km/m^{1/2-a}) = \rho/2$, and the first term $2e^{-m^{2a}/2} \leq \rho/2$ when $m \geq (2\ln(4/\rho))^{1/(2a)}$. So, as long as a is chosen such that $m = \left(\frac{64d^{3/2}}{\tau^2\rho}\right)^{1/(1/2-a)} \geq (2\ln(4/\rho))^{1/(2a)}$, the algorithm is ρ-replicable. This occurs if $\left(\frac{64}{\rho}\right)^{2a/(1/2-a)} \geq 2\ln(4/\rho)$, which is true for all values of $\rho \in (0,1)$ when $a = .07$. For simpler constants, we use $a = .1$.

Failure rate: The algorithm succeeds when the martingale concentration bound holds. So, the failure probability of `rHalfspaceWkL` is at most $e^{-m^{2a}/2} \leq \rho/4$.

Sample complexity: Plugging in $a = .1$ in the expression $m = \left(\frac{64d^{3/2}}{\tau^2\rho}\right)^{1/(1/2-a)}$ yields the conclusion. □

3.6 Replicable Boosting

In this section, we argue that a small modification of the boosting algorithm in [Ser03] is a replicable boosting algorithm. Given access to a replicable weak learner, this boosting algorithm ρ-replicably outputs a hypothesis. Boosting algorithms are a natural candidate for constructing replicable algorithms — many boosting algorithms in the standard PAC-setting are deterministic, and the final classifier returned is often a simple function of the weak learner hypotheses (e.g. a majority vote). Combining this replicable boosting algorithm with our replicable halfspace weak learner from Section 3.5 yields a replicable strong learner for halfspaces.

Specifically, we modify the smooth boosting algorithm described in [Ser03] in the batch setting, presenting it in the filtering setting [BS07]. This boosting algorithm has three main components, all of which can be made replicable: (i) checking for termination (via a statistical query), (ii) running the weak learner (replicable by assumption), and (iii) updating the weighting function (deterministic). The final classifier is a sum of returned weak learner hypotheses. With high probability over two runs, our boosting algorithm `rBoost` collects the exact same hypotheses $h_1, \dots, h_T$ from its replicable weak learner.

3.6.1 Replicable Boosting Algorithm: An Overview

In smooth boosting algorithms, a "measure" function $\mu : \mathscr{X} \to [0,1]$ determines a reweighting of distribution D. The induced reweighted distribution, denoted D_μ, is defined by the probability density function $D_\mu(x) = \mu(x) \cdot D(x)/d(\mu)$, where $d(\mu)$ is a normalizing factor $\mathbb{E}_{x \sim D}\mu(x)$. We refer to $d(\mu)$ as the *density* of measure μ. A sample $\vec{s}$ is drawn from D_μ and passed to the weak learner `rWkL`. Sampling from D_μ using example oracle EX is done by rejection sampling — draw a sample (x,y) from EX and a random $b \in_r [0,1]$; if $r \leq \mu(x)$, keep (x,y); otherwise, reject x and loop until we keep (x,y). On expectation, we require $m/d(\mu)$ examples from D to sample m examples from D_μ.

At the beginning of the algorithm, $\mu(x) = 1$ for all $x \in \text{supp}(D)$. Weak learner hypotheses h_t are used to modify update μ (and thus D_μ) for future weak learner queries. The algorithm terminates when the density $d(\mu)$ drops below the desired accuracy parameter ε — at this point, the majority vote hypothesis $\mathbf{h} = \text{sign}(\sum_t h_t)$ has accuracy at least $1 - \varepsilon$ over D.

More specifically, we define $\mu_{t+1}(x) = M(g_t(x))$ using a base measure function $M : \mathbb{R} \to [0,1]$ and score function $g : \mathscr{X} \to \mathbb{R}$. As in [Ser03], we use a capped exponential function as our base measure function $M(a) = \begin{cases} 1 & a \le 0 \\ (1-\gamma)^{a/2} & a > 0 \end{cases}$. The score function is $g_t(x) = \sum_{i=1}^{t}(h_i(x)f(x) - \theta)$, where $\theta < \gamma$ is chosen as a function of γ.

Algorithm 10. $\texttt{rBoost}^{\texttt{rWkL}}(\vec{s};r)$ // a ρ-replicable boosting algorithm
Input: A sample $\vec{s}$ of m examples $(\vec{x}_i, y_i)$ drawn i.i.d. from distribution D.
Access to replicable weak learner $\texttt{rWkL}$ with advantage γ and sample complexity $m_{\texttt{rWkL}}$.
Parameters: desired replicability ρ, accuracy ε, constant $\theta \overset{\text{def}}{=} \gamma/(2+\gamma)$, round complexity $T = O(1/\varepsilon\gamma^2)$
Output: A hypothesis $\mathbf{h} = \text{sign}\left(\sum_{t=1}^{T} h_t\right)$, where the h_t's are weak learner hypotheses.

```
$g_0(x) \overset{\text{def}}{=} 0$
$\mu_1(x) \overset{\text{def}}{=} M(g_0) = 1$                          // “Measure” function for reweighting
$t \leftarrow 0$
while 1 do
    $t \leftarrow t+1$
    $D_{\mu_t}(x) \overset{\text{def}}{=} \mu_t(x) \cdot D(x)/d(\mu_t)$                // Reweighted distribution
    $\vec{s}_1 \leftarrow \widetilde{O}(m_{\texttt{rWkL}}/\varepsilon)$ fresh examples from $\vec{s}$    // Samples for rejection sampling
    $\vec{s}_{\texttt{rWkL}} \leftarrow \texttt{RejSamp}(\vec{s}_1, m_{\texttt{rWkL}}, \mu_t; r_1)$
    Hypothesis $h_t \leftarrow \texttt{rWkL}(\vec{s}_{\texttt{rWkL}}; r_2)$
    $g_t(x) \overset{\text{def}}{=} g_{t-1}(x) + h_t(x)f(x) - \theta$           // Reweight distribution using $h_t$
    $\mu_{t+1}(x) \overset{\text{def}}{=} M(g_t(x))$
    $\vec{s}_2 \leftarrow \widetilde{O}\left(\frac{1}{\rho^2\varepsilon^3\gamma^2}\right)$ fresh examples from $\vec{s}$    // Run $\texttt{rSTAT}$ to check if $d(\mu_{t+1}) \le \varepsilon$
    if $\texttt{rSTAT}_{\tau,\rho_0,\phi}(\vec{s}_2; r_3) \le 2\varepsilon/3$ then     // tolerance $\tau = \varepsilon/3$, replicability $\rho_0 = \rho/(3T)$
        Exit while loop                              // failure rate $\rho/(12T)$, query $\phi(x,y) = \mu(x)$
return $\mathbf{h} \leftarrow \text{sign}(\sum_t h_t)$
```

Algorithm 11. `RejSamp`$(\vec{s}_{\text{all}}, m_{\text{target}}, \mu; r)$ // draw a sample from distribution D_μ
Input: sample $\vec{s}_{\text{all}}$ drawn i.i.d. from distribution D, target size of output sample $m_{\text{target}} \in [|\vec{s}_{\text{all}}|]$, and description of measure function $\mu : \mathscr{X} \to [0,1]$.
Output: $\perp$ or a sample $\vec{s}_{\text{kept}}$ of size $|\vec{s}_{\text{kept}}| = m_{\text{target}}$

```
$\vec{s}_{\text{kept}} \leftarrow \emptyset$
**for** $i = 1$ to $i = |\vec{s}_{\text{all}}|$ **do**
    Use randomness $r$ to randomly pick a $b \in [0,1]$
    **if** $\mu(x_i) \geq b$ **then**                    // Reject $(x_i, y_i)$ w. p. $1 - \mu(x)$
        $\vec{s}_{\text{kept}} \leftarrow \vec{s}_{\text{kept}} || (x_i, y_i)$      // Add example $(x_i, y_i)$ to $\vec{s}_{\text{kept}}$
    **if** $|\vec{s}_{\text{kept}}| = m_{\text{target}}$ **then**
        **return** $\vec{s}_{\text{kept}}$
**return** $\perp$                                    // Ran out of fresh samples in $\vec{s}_{\text{all}}$
```

A subtle note is that this boosting algorithm must precisely manage its sample $\vec{s}$ and random string r when invoking subroutines. In order to utilize the replicability of subroutines (e.g. `rWkL`), the boosting algorithm needs to ensure that it uses random bits from the same position in r. A first-come first-serve approach to managing r (i.e. each subroutine uses only the amount of randomness it needs) fails immediately for `rBoost` — the amount of randomness `RejSamp` needs is dependent on the sample, so the next subroutine (in this case, `rWkL`) may not be using the same randomness across two (same-randomness r) runs of `rBoost`.

If one can precisely upper bound the amount of randomness needed for each of L subroutines, then r can be split into chunks $r_1 || r_2 || \ldots || r_L$, avoiding any desynchronization issues. Alternatively, one can split r into L equally long random strings by only using bits in positions equivalent to l mod L for subroutine $l \in [L]$.

3.6.2 Analysis of `rBoost` (Algorithm 10)

As before, function f in concept class C is a function from domain $\mathscr{X}$ to $\{\pm 1\}$. D is a distribution over $\mathscr{X}$.

Theorem 3.6.1 (Replicable Boosting). *Let $\varepsilon > 0, \rho > 0$. Let* `rWkL` *be a $(\rho_r, \gamma, \delta_{\texttt{rWkL}})$-weak learner. Then $\texttt{rBoost}^{\texttt{rWkL}}(\vec{s}; r)$ is ρ-replicable and with probability at least $1-\rho$, outputs a hypothesis $\mathbf{h}$ such that $\mathbf{Pr}_{x \sim D}[\mathbf{h}(x) = f(x)] \geq 1 - \varepsilon$.* `rBoost` *runs for $T = O(1/(\varepsilon \gamma^2_{\texttt{rWkL}}))$ rounds and uses $\widetilde{O}\left(\frac{m_{\texttt{rWkL}(\rho/(6T))}}{\varepsilon^2 \gamma^2} + \frac{1}{\rho^2 \varepsilon^3 \gamma^2}\right)$ samples, where the $\widetilde{O}$ notation hides $\log(1/(\rho \varepsilon \gamma^2))$ factors and $m_{\texttt{rWkL}(\rho/(6T))}$ denotes the sample complexity of* `rWkL` *with replicability parameter $\rho/(6T)$.*

For readability, we break the proof into components for round complexity, correctness, replicability, sample complexity, and failure probability.

Proof. **Round Complexity:** Theorem 3 in [Ser03] gives a $T = O(1/(\varepsilon \gamma^2_{\texttt{rWkL}}))$ round complexity bound for this boosting algorithm in the batch setting. Analogous arguments hold in the filtering setting, so we defer to [Ser03] for brevity.

Correctness: Similarly, since replicable weak learner `rWkL` satisfies the definitions of a weak learner, the correctness arguments in [Ser03] also hold. A small difference is the termination condition — rather than terminate when the measure satisfies $d(\mu) < \varepsilon$, our algorithm terminates when the density estimated by `rSTAT` is less than $2\varepsilon/3$. We run `rSTAT` on query $\phi(x) = \mu(x)$ with tolerance parameter $\varepsilon/3$. Thus, when the `rBoost` terminates, $d(\mu_t) < \varepsilon$.

Replicability: We show this boosting algorithm not only replicably outputs the same hypothesis $\mathbf{h}$, but that each returned weak learner hypothesis h_t is identical across two runs of the boosting algorithms (using the same randomness r) with high probability. The reweighted distribution D_{μ_t} depends only on the previous weak learner hypotheses $h_1, \ldots, h_{t-1}$, so the only possibilities for loss of replicability are: (i) returning $\perp$ while rejection sampling from D_μ; (ii) running the replicable weak learner; and (iii) using a statistical query to decide to exit the while loop. We note that our choice of parameters adds non-replicabilty at most $\rho/(3T)$ for each and apply a union bound over at most T rounds of boosting.

1. By Lemma 3.6.2, $O(\frac{m_{\texttt{rWkL}}}{\varepsilon} \cdot \log(T/\rho))$ examples suffice to guarantee `RejSamp` outputs $\perp$ with probability at most $\rho/(6T)$. Union bounding over two runs, this is at most $\rho/(3T)$.

2. By Lemma 3.6.4, running `rWkL` with replicability parameter $\rho/(6T)$ will add a $\rho/(3T)$ contribution to the non-replicability.

3. We run `rSTAT` with replicability parameter $\rho/(3T)$.

Sample Complexity: There are two contributions to the sample complexity: samples used for the weak learner `rWkL`, and samples used by `rSTAT` to estimate the density of measure μ_t. Fresh samples are used for each of T rounds of boosting. Together, by Theorem 3.2.3 and the definition of $\vec{s}_1$ (in Algorithm 10), the sample complexity is

$$O\left(T \cdot \left(\frac{m_{\texttt{rWkL}(\rho/(6T))}}{\varepsilon} \cdot \log(T/\rho) + \frac{\log(T/\rho)}{(\varepsilon^2)(\rho)^2}\right)\right) = \widetilde{O}\left(\frac{m_{\texttt{rWkL}(\rho/(6T))}}{\varepsilon^2\gamma^2} + \frac{1}{\rho^2\varepsilon^3\gamma^2}\right)$$

where the $\widetilde{O}$ notation hides $\log(1/(\rho\varepsilon\gamma^2))$ factors and $m_{\texttt{rWkL}(\rho/(6T))}$ denotes the sample complexity of `rWkL` with replicability parameter $\rho\varepsilon\gamma^2$.

Failure Probability: Assuming the weak learner returns correct hypotheses when it is replicable, the boosting algorithm `rBoost` is correct when it is replicable, so the failure probability is bounded above by ρ.[5] $\square$

3.6.3 Rejection Sampling Lemmas

Next, we show that replicability composes well with rejection sampling throughout the execution of `rBoost`.

[5] A more precise sample complexity statement in terms of the failure probability δ can be obtained by unboxing the error probabilities. The algorithm can fail if `RejSamp` outputs $\perp$, if `rWkL` fails, and if `rSTAT` fails. Bounding each of these quantities by $\delta/(3T)$ ensures that the `rBoost` has failure rate δ.

Lemma 3.6.2 (Failure Rate of RejSamp)**.** *Let measure μ have density $d(\mu) \geq \varepsilon/3$. Let $\vec{s}_{all}$ be a sample drawn i.i.d. from distribution D. If $|\vec{s}_{all}| \geq \frac{24m_{target}}{\varepsilon} \cdot \log(1/\delta)$, then* RejSamp$(\vec{s}_{all}, m_{target}, \mu; r)$ *outputs $\perp$ with probability at most δ.*

Proof. The probability RejSamp outputs $\perp$ is precisely the probability a binomial random variable $X \sim B(|\vec{s}_{\text{all}}|, d(\mu))$ is at most m_{target}. By a Chernoff bound, $\mathbf{Pr}[X \leq (1-.5)|\vec{s}_{\text{all}}| \cdot d(\mu)] \leq \exp(-|\vec{s}_{\text{all}}| \cdot d(\mu)/8) \leq \exp(-|\vec{s}_{\text{all}}| \cdot \varepsilon/24)$. Thus, $\mathbf{Pr}[X \leq m_{\text{target}}] \leq \delta$. □

Remark 3.6.3. *The following is a justification of why we may assume $d(\mu) \geq \varepsilon/3$ in the previous lemma.*

When RejSamp *is first called in round* 1 *of* rBoost*, $\mu(x) = 1$ for all x, so $d(\mu) = 1$. In subsequent rounds $t \geq 2$,* RejSamp *is only called if, in previous round $t-1$,* rSTAT *estimated $d(\mu)$ to be at least $2\varepsilon/3$.* rSTAT *is run with tolerance $\varepsilon/3$, so $d(\mu) \geq \varepsilon/3$ whenever* rSTAT *succeeds. Whenever we apply the above lemma, we are assuming the success of previous subroutines (by keeping track of and union bounding over their error).*

The following lemma shows that rejection sampling before running a replicable algorithm only increases the non-replicability ρ by a factor of 2. To be precise, we let p denote the probability the rejection sampler returns $\perp$. However, when we apply this Lemma in the proof of Theorem 3.6.1, we will have already accounted for this probability.

Lemma 3.6.4 (Composing Replicable Algorithms with Rejection Sampling)**.** *Let $\mathscr{A}(\vec{s}, r)$ be a ρ-replicable algorithm with sample complexity m Let $\mu : \mathscr{X} \to [0,1]$. Consider $\mathscr{B}$, the algorithm defined by composing* RejSamp$(\vec{s}', m, \mu; r')$ *with $\mathscr{A}(\vec{s}; r)$. Let q be the probability that* RejSamp *returns $\perp$. Then $\mathscr{B}$ is a $2q + 2\rho$-replicable algorithm.*

Proof. Since $\mathscr{A}$ is ρ-replicable, $\mathbf{Pr}_{\vec{s}_1, \vec{s}_2, r}[\mathscr{A}(\vec{s}_1; r) = \mathscr{A}(\vec{s}_2; r)] \geq 1 - \rho$. However, the rejection sampling is done with correlated randomness, so $\vec{s}_1$ and $\vec{s}_2$ are not independent. Consider an imaginary third run of algorithm $\mathscr{A}(\vec{s}_3; r)$, where $\vec{s}_3$ is drawn from D_μ using separate randomness.

We will use a triangle-inequality-style argument (and a union bound) to derive the conclusion. Conditioned on `RejSamp` not returning $\perp$, algorithm $\mathscr{B}(\vec{s}_1'; r'||r)$ returns the same result as $\mathscr{A}(\vec{s}_3, r)$ (when both algorithms use randomness r for the execution of $\mathscr{A}$) with probability at least $1-\rho$. The same statement holds for the second run $\mathscr{B}(\vec{s}_1'; r'||r)$. Thus,

$$\mathbf{Pr}_{\vec{s}_1', \vec{s}_2', r'||r}\left[\mathscr{B}(\vec{s}_1'; r'||r) = \mathscr{B}(\vec{s}_2'; r'||r) \mid \text{neither run outputs } \perp\right] \geq 1-2\rho.$$

Finally, $\mathscr{B}$ may fail to be replicable if either `RejSamp` call returns $\perp$, so we union bound over this additional $2q$ probability. □

3.6.4 Replicable Strong Halfspace Learner

We give two replicable strong learners for halfspaces by combining boosting algorithm `rBoost` with replicable weak halfspace learners `rHalfspaceWkL` and $\texttt{rHalfspaceWkL}^{\text{box}}$.

Corollary 3.6.5. *Let D be a distribution over $\mathbb{R}^d$, and let $f : \mathbb{R}^d \to \{\pm 1\}$ be a halfspace with margin τ in D. Let $\varepsilon > 0$. Then*

- *Algorithm* `rBoost` *run with weak learner* `rHalfspaceWkL` *ρ-replicably returns a hypothesis* $\mathbf{h}$ *such that, with probability at least $1-\rho$, $\mathbf{Pr}_{\vec{x}\sim D}[\mathbf{h}(\vec{x}) = f(\vec{x})] \geq 1-\varepsilon$, using a sample of size $\widetilde{O}\left(\frac{d^{10/9}}{\tau^{76/9}\rho^{20/9}\varepsilon^{28/9}}\right)$.*
- *Algorithm* `rBoost` *run with weak learner* $\texttt{rHalfspaceWkL}^{box}$ *ρ-replicably returns a hypothesis* $\mathbf{h}$ *such that, with probability at least $1-\rho$, $\mathbf{Pr}_{\vec{x}\sim D}[\mathbf{h}(\vec{x}) = f(\vec{x})] \geq 1-\varepsilon$, using a sample of size $\widetilde{O}\left(\frac{d^{15/4}}{\tau^{10}\rho^{5/2}\varepsilon^{9/2}}\right)$.*

Proof. For the first strong learner, we compose Theorem 3.6.1 with Theorem 3.5.2. `rHalfspaceWkL` has advantage $\gamma = \tau/4$, so `rBoost` has round complexity $T = O(1/(\varepsilon\gamma^2)) = O(1/(\varepsilon\tau^2))$. `rBoost` runs `rHalfspaceWkL` with parameter $\rho_{\texttt{rWkL}} = \rho/6T$, so the sample complexity $m_{\texttt{rHalfspaceWkL}}$ is $O\left(\frac{d^{10/9}}{\tau^{58/9}\rho^{20/9}\varepsilon^{10/9}}\right)$. Thus, the sample complexity for the boosting algorithm is

$$\widetilde{O}\left(\frac{d^{10/9}}{\tau^{76/9}\rho^{20/9}\varepsilon^{28/9}}\right)$$

For the second strong learner, we compose Theorem 3.6.1 with Theorem 3.5.14. As before, `rBoost` has round complexity $T = O(1/(\varepsilon\tau^2))$ and runs $\texttt{rHalfspaceWkL}^{\text{box}}$ with parameter $\rho_{\texttt{rWkL}} = \rho/6T$. The sample complexity $m_{\texttt{rHalfspaceWkL}^{\text{box}}}$ is $O\left(\left(\frac{d^{3/2}}{\tau^2\rho_{\texttt{rWkL}}}\right)^{5/2}\right) = \left(\left(\frac{d^{3/2}}{\tau^4\rho\varepsilon}\right)^{5/2}\right)$. Thus, the sample complexity for the boosting algorithm is

$$\widetilde{O}\left(\frac{m_{\texttt{rWkL}(\rho/(6T))}}{\varepsilon^2\gamma^2} + \frac{1}{\rho^2\varepsilon^3\gamma^2}\right) = \widetilde{O}\left(\frac{1}{\varepsilon^2\gamma^2}\left(\frac{d^{3/2}}{\tau^2\rho_{\texttt{rWkL}}}\right)^{5/2}\right) = \widetilde{O}\left(\frac{d^{15/4}}{\tau^{10}\rho^{5/2}\varepsilon^{9/2}}\right).$$

□

Remark 3.6.6. $\texttt{rHalfspaceWkL}^{box}$ *runs in time polynomial in the input parameters. So, the strong learner obtained by running boosting algorithm* `rBoost` *with weak learner* $\texttt{rHalfspaceWkL}^{box}$ *is a* $\text{poly}(1/\varepsilon, 1/\rho, 1/\tau, d)$*-time algorithm. However, the other weak learner* `rHalfspaceWkL` *uses a foams construction subroutine from [KORW12] that takes expected exponential in d runtime. The corresponding strong learner runs in time polynomial in* $1/\varepsilon, 1/\rho$*, and* $1/\tau$*, but exponential in d.*

3.6.5 Discussion

Algorithm 10 follows the smooth boosting framework of Servedio [Ser03], which also shows how to boost a weak halfspace learner under a margin assumption on the data. They show that their boosted halfspace learner obtains a hypothesis with good margin on the training data, and then apply a fat-shattering dimension argument to show generalization to the underlying distribution with sample complexity $\tilde{O}(1/(\tau\varepsilon)^2)$. Notably, this gives sample complexity independent of d. Moreover, their smooth boosting algorithm is tolerant to malicious noise perturbing an $\eta \in O(\tau\varepsilon)$ fraction of its sample.

A generic framework for differentially private boosting was given in [BCS20], with an application to boosting halfspaces. Their boosting algorithm also follows the smooth boosting framework, but uses a variant of the round-optimal boosting algorithm given in [BHK09]. Their halfspace learner similarly requires a margin assumption on the data and tolerates random classification noise at a rate $\eta \in O(\tau\varepsilon)$. They give two generalization arguments for their halfspace learner, both of which are dimension-independent. The first follows from prior work showing that differential privacy implies generalization [BNS$^+$16a] and gives sample complexity $\tilde{O}(\frac{1}{\varepsilon\alpha\tau^2} + \frac{1}{\varepsilon^2\tau^2} + \alpha^{-2} + \varepsilon^{-2})$ for approximate differential privacy parameters (α, β). The second follows from a fat-shattering dimension argument and gives a tighter bound of $\widetilde{O}\left(\frac{1}{\varepsilon\alpha\tau^2}\right)$.

Boosting algorithms have been thoroughly studied over the past few decades, and there are many types of boosting algorithms (e.g. distribution-reweighting, branching-program, gradient boosting) with different properties (e.g. noise-tolerance, parallelizability, smoothness, batch vs. filtering). It would be interesting to see which of these techniques can be made replicable, and at what cost.

3.7 SQ–Replicability Lower Bound

How much does it cost to make a nonreplicable algorithm into a replicable one? In this section, we show a lower bound for replicable statistical queries via a reduction from the coin problem.

Theorem 3.7.1 (SQ–Replicability Lower Bound)**.** *Let $\tau > 0$ and let $\delta \leq 1/16$. Let query $\phi : \mathscr{X} \to [0,1]$ be a statistical query. Let $\mathscr{A}$ be a ρ-replicable SQ algorithm for ϕ with tolerance less than τ and success probability at least $1-\delta$. Then $\mathscr{A}$ has sample complexity at least $m \in \Omega(1/(\tau^2\rho^2))$.*

Note that this nearly matches the replicable statistical query upper bound in Theorem 3.2.3, in the case that $\delta \in \Theta(\rho)$.

Recall the coin problem: promised that a 0-1 coin has bias either $1/2 - \tau$ or $1/2 + \tau$ for some fixed $\tau > 0$, how many flips are required to identify the coin's bias with high probability?

Proof of Theorem 3.7.1. A τ-tolerant ρ-replicable SQ algorithm $\mathscr{A}$ for ϕ naturally induces a ρ-replicable algorithm $\mathscr{B}$ for the τ-coin problem — $\mathscr{B}$ runs $\mathscr{A}$ (the results of the coin flips are the $\phi(x)$'s), and $\mathscr{B}$ accepts (outputs 1) if $\mathscr{A}$'s output is $\geq 1/2$, otherwise rejects. The success probability of $\mathscr{B}$ is at least that of $\mathscr{A}$. As $\mathscr{A}$ is replicable for all distributions, $\mathscr{B}$ also satisfies the assumption in Lemma 3.7.2 that $\mathscr{B}$ is ρ-replicable for coins with bias in $(1/2-\tau, 1/2+\tau)$. By Lemma 3.7.2, any replicable algorithm solving the coin problem with these parameter has sample complexity $m \in \Omega(1/(\tau^2\rho^2))$, implying the lower bound. $\square$

Lemma 3.7.2 (Sample Lower Bound for the Coin Problem)**.** *Let $\tau < 1/4$ and $\rho < 1/16$. Let $\mathscr{B}$ be a ρ-replicable algorithm that decides the coin problem with success probability at least $1-\delta$ for $\delta = 1/16$. Furthermore, assume $\mathscr{B}$ is ρ-replicable, even if its samples are drawn from a coin $\mathbf{C}$ with bias in $(1/2-\tau, 1/2+\tau)$. Then $\mathscr{B}$ requires sample complexity $m \in \Omega(1/(\tau^2\rho^2))$, i.e. $\rho \in \Omega(1/\tau\sqrt{m})$.*

Proof. Assume we have an algorithm $\mathscr{B}(b_1..b_m;r)$ of sample complexity m so that (i) if the b_i's are chosen i.i.d. in $\{0,1\}$ with bias $1/2-\tau$, $\mathscr{B}$ accepts with at most δ probability (over both random r and the b_i's), and (ii) if the b_i's are drawn i.i.d. with bias $1/2+\tau$, $\mathscr{B}$ accepts with at least $1-\delta$ probability.

Let $p \in [0,1]$ denote the bias of a coin. Since $\mathscr{B}$ is ρ-replicable, $\mathscr{B}$ is ρ-replicable for any distribution on p. In particular, pick $p \in_U [1/2-\tau, 1/2+\tau]$. Let $\mathbf{C}_{-\tau}$ denote a coin with bias $1/2-\tau$, and let $\mathbf{C}_{+\tau}$ denote a coin with bias $1/2+\tau$. By Markov's inequality, each of the following is true with probability at least $1-1/4$ over choice of r:

- $\mathbf{Pr}_{b_1..b_m \sim_{\text{i.i.d.}} \mathbf{C}_{-\tau}}[\mathscr{B}(b_1..b_m;r) \text{ accepts}] \leq 4\delta$
- $\mathbf{Pr}_{b_1..b_m \sim_{\text{i.i.d.}} \mathbf{C}_{+\tau}}[\mathscr{B}(b_1..b_m;r) \text{ accepts}] \geq 1-4\delta$
- When p is chosen between $1/2-\tau$ and $1/2+\tau$ uniformly, and then $b_1..b_m, b'_1..b'_m$ are sampled i.i.d. with expectation p, $\mathbf{Pr}[\mathscr{B}(b_1..b_m;r) = A(b'_1..b'_m;r)] \geq 1-4\rho$.

By a union bound, there exists an r^* so that every above statement is true. Note that for any p, given $\sum b_i = j$, the samples $b_1..b_m$ are uniformly distributed among all Boolean vectors of Hamming weight j. Let $a_j \overset{\text{def}}{=} \mathbf{Pr}[\mathscr{B}(b_1..b_m;r^*) \text{ accepts } | \sum b_i = j]$. Then the probability $\mathscr{B}$ accepts using r^* on bits with bias p is $\texttt{Acc}(p) = \sum_j a_j \binom{m}{j} p^j(1-p)^{m-j}$. In particular, this is a continuous and differentiable function.

Since $\texttt{Acc}(1/2-\tau) < 4\delta < 1/4$ and $\texttt{Acc}(1/2+\tau) > 1-4\delta > 3/4$, there is a $q \in (1/2-\tau, 1/2+\tau)$ with $\texttt{Acc}(q) = 1/2$. We show that $\texttt{Acc}(p)$ is close to $1/2$ for all p close to q by bounding the derivative $\texttt{Acc}'(p)$ within the interval $[1/4, 3/4]$, which contains $[1/2-\tau, 1/2+\tau]$.

By the standard calculus formulas for derivatives,

$$\begin{aligned}\texttt{Acc}'(p) &= \sum_j a_j \binom{m}{j} (jp^{j-1}(1-p)^{m-j} - (m-j)p^j(1-p)^{m-j-1})\\ &= \sum_j a_j \binom{m}{j} p^j(1-p)^{m-j}(j/p - (m-j)/(1-p))\\ &= \sum_j a_j \binom{m}{j} p^j(1-p)^{m-j}(j-mp)/(p(1-p)).\end{aligned}$$

Since $1/4 < p < 3/4$, $p(1-p) > 3/16 > 1/6$, and $0 \leq a_j \leq 1$. So this sum is at most

$$\sum_j \binom{m}{j} p^j(1-p)^{m-j} 6|j-mp| = 6\mathbb{E}_j[|j-mp|]$$

where the last expectation is over j chosen as the sum of m random Boolean variables of expectation p. This expectation is $O(m^{1/2})$ because the expectation of the absolute value of the difference between any variable and its expectation is at most the standard deviation for the variable.

Since the derivative is at most $O(\sqrt{m})$, there is an interval I of length $\Omega(1/\sqrt{m})$ around q so that $1/3 < \texttt{Acc}(p) < 2/3$ for all p in this interval. Since $\texttt{Acc}(p) \notin (1/3, 2/3)$ at $p = 1/2-\tau$ and $p = 1/2+\tau$, interval I is entirely contained in $(1/2-\tau, 1/2+\tau)$. So, there is an $\Omega(1/\tau\sqrt{m})$ chance that a random $p \in_U [1/2-\tau, 1/2+\tau]$ falls in interval I. For $p \in I$, there is a $2\texttt{Acc}(p)(1-\texttt{Acc}(p)) > 4/9$

conditional probability of non-replicability for $\mathscr{B}$. Therefore, $\rho \geq \Omega(1/\tau\sqrt{m})$ and $m \in \Omega(1/\tau^2\rho^2)$.

$\square$

3.8 Appendix: Replicability: Alternative Definitions and Properties

In this section, we consider a few alternative criteria for replicability and show how they relate to our definition of replicability. We also demonstrate other robustness properties of replicability such as amplifying the parameters, as well as data/randomness reuse.

Alternative Definitions and Amplification. In the main body of the paper, we have chosen to define $\mathscr{A}$ as have two sources of random inputs: samples $\vec{s}$ drawn from distribution D and internal randomness r. $\mathscr{A}$ has no additional inputs. However, we could more generally define $\mathscr{A}$ to have additional, nonrandom inputs. In this more general definition, we define $\mathscr{A}(x;\vec{s};r)$ where $\vec{s}$ and r are as defined previously, and x is an auxiliary input (or tuple of inputs). $\mathscr{A}(x;\vec{s};r)$ is ρ-replicable with respect to distribution D if for every input x, $\mathscr{A}(x;\vec{s};r)$ is ρ-replicable. This definition generalizes both pseudodeterministic algorithms (in which there is no underlying distribution, so $\vec{s}$ is empty) as well as our definition of replicable learning algorithms (in which there are no additional inputs, so x is empty).

Rather than parameterize replicability by a single parameter ρ, one could use two variables (η,ν).

Definition 3.8.1 ((η,ν)-replicability)**.** *Let $\mathscr{A}(x;\vec{s};r)$ be an algorithm, where $\vec{s}$ are samples from D, and r is the internal randomness. We say that a particular random string r is η-good for $\mathscr{A}$ on x with respect to D if there is a single "canonical" output Z_r such that* $\mathbf{Pr}[\mathscr{A}(x;\vec{s};r) = Z_r] \geq 1-\eta$. *Then $\mathscr{A}$ is (η,ν)-replicable with respect to D if, for each x, the probability that a random r is η-good for $\mathscr{A}$ (on x and D) is at least* $1-\nu$.

(η, ν)-replicability is qualitatively the same as ρ-replicability, but might differ by polynomial factors. If $\mathscr{A}$ is (η, ν)-replicable, then $\mathscr{A}$ is ρ-replicable on D, where $\rho \leq 2\eta + \nu$. The probability that two runs of $\mathscr{A}$, using the same internal randomness r, output different results is at most $\mathbf{Pr}[r \text{ not } \eta\text{-good}]$ plus the probability that at least one run is not the special output Z_r (conditioned on r being η-good). In the other direction, if $\mathscr{A}$ is ρ-replicable, then $\mathscr{A}$ is $(\rho/\nu, \nu)$-replicable for any $\rho \leq \nu < 1$. Say there is a ν probability that r is not η-good. Conditioned on picking a not η-good r, there is a conditional (at least) η probability of the second run of $\mathscr{A}$ returning something different than the first run.[6] Thus, $\rho \geq \eta\nu$.

A similar definition called "pseudo-global stability", developed independently to our work, appears in [GKM21]. That definintion parametrizes by the sample complexity m and does not explicitly parametrize by auxiliary input x. Additionally, their definition includes an (α, β)-accuracy guarantee on Z_r, the very likely output. To keep both definitions consistent with their original conventions, we write $\eta' \overset{\text{def}}{=} 1 - \eta$ and $\nu' \overset{\text{def}}{=} 1 - \nu$.

Definition 3.8.2 (Pseudo-global stability, Definition 15 in [GKM21])**.** *A learning algorithm $\mathscr{A}$ with sample complexity m is said to be (α, β)-accurate, (η', ν')-pseudo-globally stable if there exists a hypothesis h_r for every $r \in supp(R)$ (depending on D) such that $\mathbf{Pr}_{r \sim R}[\mathrm{err}_D(h_r) \leq \alpha] \geq 1 - \beta$ and*

$$\mathbf{Pr}_{r \sim R}\left[\mathbf{Pr}_{\vec{s} \sim D^m}[\mathscr{A}(\vec{s}; r) = h_r] \geq \eta'\right] \geq \nu'$$

where $\vec{s}$ is a sample of m (labeled) examples (x_i, y_i) drawn from distribution D.

The final condition of Definition 3.8.2 is equivalent to saying that i) a randomly chosen r is η'-good with probability at least ν', and ii) for every r, h_r is the output that witnesses the η-goodness. Carrying the accuracy guarantee through the previous argument, an (α, β)-accurate

[6]If $1 - \eta \geq 1/2$, then the probability that two runs of $\mathscr{A}$ using the same randomness returns the same result is at most $(1 - \eta)^2 + \eta^2$, i.e., when $\mathscr{A}$ has only two possible outputs. This is less than $1 - \eta$, assuming $1 - \eta \geq 1/2$. If $1 - \eta < 1/2$, then there must be more than two outputs, and the probability of nonreplicability is again larger than η.

(η', ν')-pseudo-globally-stable algorithm $\mathscr{A}$ implies a $(2(1-\eta')+(1-\nu')) = (2\eta+\nu)$-replicable algorithm $\mathscr{A}$ also with (α, β)-accuracy.

If we are willing to increase the sample complexity of $\mathscr{A}$, we can make the connection stronger:

Theorem 3.8.3 (Amplification of Replicability)**.** *Let* $0 < \eta, \nu, \beta < 1/2$ *and* $m > 0$*. Let* $\mathscr{A}$ *be an* (η, ν)*-replicable algorithm for distribution D with sample complexity m and failure rate* β*. If* $\rho > 0$ *and* $\nu + \rho < 3/4$*, then there is a* ρ*-replicable algorithm* $\mathscr{A}'$ *for D with sample complexity* $m' = \widetilde{O}(m(\log 1/\beta)^3/(\rho^2(1/2-\eta)^2)$ *and failure rate at most* $O(\beta+\rho)$*. The construction of* $\mathscr{A}'$ *does not depend on D.*

Proof. Set $k = 3\log 1/\beta$. For each random string r, let D_r be the distribution on outputs of $\mathscr{A}(x;\vec{s};r)$ (over random $\vec{s}$). Algorithm $\mathscr{A}'$ randomly picks k-many strings $r_1, \ldots, r_k$, runs the replicable heavy-hitters algorithm (Algorithm 4) on the distributions $D_{r_1}, \ldots, D_{r_k}$, and outputs the first returned heavy-hitter (or $\perp$ if each subroutine returns the empty list). We say there are k *rounds* of $\mathscr{A}'$, one per random string r.

The replicability of `rHeavyHitters` implies the replicability of $\mathscr{A}'$. We show that a heavy-hitter in D_r for randomly chosen r is often a correct answer, except with probability comparable to β.

By definition, r is η-good iff D_r has a $1-\eta$ heavy-hitter. Since $\eta < 1/2$, this heavy-hitter will be unique, and there will be no other $1-\eta > 1/2$ heavy-hitters. Given any r, we can draw from distribution D_r by running algorithm $\mathscr{A}$ with fresh samples $\vec{s}$. Consider running the replicable heavy-hitters algorithm with parameters $v = (3/2-\eta)/2$, $\varepsilon = (1/2-\eta)/2$, and replicability $\rho' = \rho/k$. These are chosen so that $v+\varepsilon = 1-\eta$ and $v-\varepsilon = 1/2$. If r is η-good, then `rHeavyHitters` will return the (unique) majority element for D_r with probability at least $1-\rho/k$. If r is not $1/2$-good[7], the replicable heavy-hitters algorithm with the same parameters will return the empty list with

[7]Since $\eta < 1/2$ by assumption, r being not $1/2$-good implies r is not η-good.

probability at least $1-\rho/k$.

Next, we compute the conditional probability that the first element that `rHeavyHitters` returns is correct. The probability that `rHeavyHitters` produces an empty list in one round is at most ν (when the randomly chosen r is not η-good) plus ρ/k (when r is η-good but the heavy-hitters algorithm fails). At most a (2β)-fraction of random strings r satisfy both of the following two conditions: i) D_r has a majority element Z_r and ii) Z_r is an incorrect output. Thus, the conditional probability of outputting an incorrect answer, given `rHeavyHitters` produces a non-empty output, is at most $(2\beta+\rho/k)/(1-\nu-\rho/k)$. By assumption, $\nu+\rho/k<3/4$, so this is $O(\beta+\rho)$.

So far, we have bounded the probability that $\mathscr{A}'$ returns an incorrect answer. $\mathscr{A}'$ could also fail if `rHeavyHitters` returns the empty list in each of k rounds. Since $\nu+\rho/k<3/4$, this happens with probability at most $(3/4)^k\leq\beta$. So, the overall probability of error is at most $O(\beta+\rho)$.

If two runs of $\mathscr{A}'$ use the same r_i's and same randomness for each heavy-hitters call, they only produce different answers if a pair of `rHeavyHitters` calls produces different answers in the same round. By the replicability of `rHeavyHitters`, this only happens with probability ρ/k each round, for a total non-replicability probability at most ρ.

$\mathscr{A}'$ calls `rHeavyHitters` $k=O(\log 1/\beta)$ times. Each example used by `rHeavyHitters` is created by running $\mathscr{A}$, which has sample complexity m. By Lemma 3.3.3, $\texttt{rHeavyHitters}_{\rho',v,\varepsilon}$ has sample complexity $\widetilde{O}\left(\frac{1}{\rho'^2\varepsilon^2(v-\varepsilon)^2}\right)$. Substituting in $\rho'=\rho/k, \varepsilon=(1/2-\eta)/2$, and $v-\varepsilon=1/2$, $\mathscr{A}$ has sample complexity $km\cdot\widetilde{O}\left(\frac{k^2}{\rho^2(1/2-\eta)^2}\right)=\widetilde{O}\left(\frac{m\log^3(1/\beta)}{\rho^2(1/2-\eta)^2}\right)$. □

Corollary 3.8.4. *Let $\alpha>0$ and $\rho<1/4-\alpha$. Let $\mathscr{A}$ be a ρ-replicable algorithm using m samples is correct except with error at most β. Then for arbitrary ρ' satisfying $\rho>\rho'>0$, there is a ρ'-replicable algorithm $\mathscr{A}'$ with sample complexity $m'=\widetilde{O}\left(\frac{m\log^3(1/\beta)}{\rho'^2\alpha^2}\right)$ that is correct except with error at most $O(\beta+\rho')$.*

Proof. By the arguments immediately after Definition 3.8.1, a ρ-replicable algorithm implies a $(\rho/x,x)$-replicable algorithm. Choosing $x=1/2-\alpha$ allows us to apply Theorem 3.8.3 for any

$\rho < 1/4$. The $(1/2-\eta)$ term in Theorem 3.8.3 simplifies to $\alpha/(1-2\alpha)$ in this context. When α can be chosen as a constant, the sample complexity simplifies to $m' = \widetilde{O}(m\log^3(1/\beta)/\rho'^2)$. $\square$

Public versus Private Randomness. We define replicability as the probability that when run twice using the same (public) randomness, but with independently chosen data samples, the algorithm returns the same answer. In [GL19], the authors define a related concept, but divide up the randomness into two parts, where only the first randomness part gets reused in the second run of the algorithm. In their applications, there are no data samples, so re-running the algorithm using identical randomness would always give identical results; rather, they were trying to minimize the amount of information about the random choices that would guarantee replicability, i.e., minimize the length of the first part.

Similarly, we could define a model of replicability that involved two kinds of random choices. Define $\mathscr{A}(x;\vec{s};r_{pub},r_{priv})$, $\vec{s}=(s_1,\ldots,s_m)$ to be ρ-replicable with respect to r_{pub} and D if for every x, random $\vec{s_1}$ and $\vec{s_2}$ drawn from D^m, and random $r_{pub},r_{priv},r'_{priv}$,

$$\mathbf{Pr}[\mathscr{A}(x;\vec{s_1};r_{pub},r_{priv}) = \mathscr{A}(x;\vec{s_2};r_{pub},r'_{priv})] \geq 1-\rho.$$

If we want to minimize the amount of information we need to store to guarantee replicability, keeping r_{priv} and r_{pub} distinct may be important. However, if all we want is to have a maximally replicable algorithm, the following observation shows that it is always better to make the entire randomness public.

Lemma 3.8.5. *If $\mathscr{A}(x;\vec{s};r_{pub},r_{priv})$ is ρ-replicable w.r.t. r_{pub} over D, then $\mathscr{A}(x;\vec{s};r_{pub},r_{priv})$ is ρ-replicable with respect to (r_{pub},r_{priv}) over D.*

Proof. We show for each value of x and r_{pub},

$$\mathbf{Pr}[\mathscr{A}(x;\vec{s_1};r_{pub},r_{priv}) = \mathscr{A}(x;\vec{s_2};r_{pub},r'_{priv})] \leq \mathbf{Pr}[\mathscr{A}(x;\vec{s_1};r_{pub},r_{priv}) = \mathscr{A}(x;\vec{s_2};r_{pub},r_{priv})].$$

Fix x and r_{pub}. For each possible value R of r_{priv} and output Z, let $q_{R,Z} = \mathbf{Pr}[\mathscr{A}[(x;\vec{s}_1;r_{pub},R)] = Z]$, and let $\vec{q}_R$ be the vector indexed by Z whose Z^{th} coordinate is $q_{R,Z}$. Then

$$\mathbf{Pr}[\mathscr{A}[(x;\vec{s}_1;r_{\text{pub}},R) = \mathscr{A}(x;\vec{s}_2;r_{pub},R)] = \sum_Z (q_{R,Z})^2 = ||\vec{q}_R||_2^2,$$

and

$$\mathbf{Pr}[\mathscr{A}(x;\vec{s}_1;r_{pub},R) = \mathscr{A}(x;\vec{s}_2;r_{\text{pub}},R')] = \sum_Z (q_{R,Z}q_{R',Z}) = \langle \vec{q}_R, \vec{q}_{R'} \rangle.$$

Thus,

$$\begin{aligned}
\mathbf{Pr}[\mathscr{A}(x;\vec{s}_1;r_{pub},r_{priv}) = \mathscr{A}(x;\vec{s}_2;r_{pub},r'_{priv})] &= \mathbb{E}_{R,R'}[\langle \vec{q}_R, \vec{q}_{R'} \rangle] \\
&\leq \mathbb{E}_{R,R'}[||\vec{q}_R||_2 ||\vec{q}_{R'}||_2] \\
&= (\mathbb{E}_R[||\vec{q}_R||_2])^2 \\
&\leq \mathbb{E}_R[||\vec{q}_R||_2^2] \\
&= \mathbf{Pr}[\mathscr{A}(x;\vec{s}_1;r_{\text{pub}},r_{\text{priv}}) = \mathscr{A}(x;\vec{s}_2;r_{pub},r_{\text{priv}})].
\end{aligned}$$

□

We will implicitly use this observation in the boosting algorithm section, since it will be convenient to think of the two runs of the boosting algorithm as picking samples each step independently, when using the same random string would create some correlation.

Replicability Implies Generalization. We show that a hypothesis generated by a replicable algorithm has a high probability of having generalization error close to the empirical error. Let h be a hypothesis, c be a target concept, and D be a distribution. The *risk* (generalization error) of $R(h) \overset{\text{def}}{=} \mathbf{Pr}_{x \sim D}[h(x) \neq c(x)]$. If $\vec{s}$ is a sample drawn i.i.d. from D, then the *empirical risk* $\widehat{R}_{\vec{s}}(h) \overset{\text{def}}{=} \mathbf{Pr}_{x \in \vec{s}}[h(x) \neq c(x)]$.

Lemma 3.8.6 (Replicability Implies Generalization)**.** *Let sample $\vec{s} \sim D^n$, and let $\delta > 0$. Let h be a* *hypothesis output by ρ-replicable learning algorithm $\mathscr{A}(\vec{s};r)$, where r is a random string. Then, with probability at least $1-\rho-\delta$ over the choice of $\vec{s}$ and r, $R(h) \leq \widehat{R}_{\vec{s}}(h) + \sqrt{\ln(1/\delta)/(2n)}$.*

Proof. Consider running $\mathscr{A}(\vec{s}_2;r)$, where $\vec{s}_2$ is an independent sample of size m drawn from D, but r is the same as before. Let h_2 denote the returned hypothesis. Since h_2 is independent of $\vec{s}$, $\mathbf{Pr}_{\vec{s}\sim D^n}[\widehat{R}_{\vec{s}}(h_2) - R(h_2) \geq \varepsilon] \leq \exp(-2n\varepsilon^2)$ for $\varepsilon > 0$ by Hoeffding's inequality. By the replicability of $\mathscr{A}$, $h_2 = h$ with probability at least $1-\rho$. By a union bound, $R(h) \geq \widehat{R}_{\vec{s}}(h) + \sqrt{\ln(1/\delta)/2n}$ with probability at least $1-\rho-\delta$. $\square$

In the above argument, we use the definition of replicability to create independence between $\vec{s}$ and h, allowing us to use Hoeffding's inequality.

Connections to Data Reuse. We consider the adaptive data analysis model that appears in [DFH$^+$15b] and [DFH$^+$15a], and we prove that replicable algorithms are resiliant against adaptive queries (Lemma 3.8.7). The proof is via a hybrid argument.

Lemma 3.8.7 (Replicability $\Longrightarrow$ Data Reusability)**.** *Let D be a distribution over domain $\mathscr{X}$. Let $\mathscr{M}$ be a mechanism that answers queries of the form $q : \mathscr{X} \to \{0,1\}$ by drawing a sample S of n i.i.d. examples from D and returning answer a. Let $\mathscr{A}$ denote an algorithm making m* adaptive *queries, chosen from a set of queries Q, so that the choice of q_i may depend on q_j, a_j for all $j < i$. Denote by $[\mathscr{A},\mathscr{M}]$ the distribution over transcripts $\{q_1, a_1, \dots q_m, a_m\}$ of queries and answers induced by $\mathscr{A}$ making queries of $\mathscr{M}$. Let $\mathscr{M}'$ be a mechanism that behaves identically to $\mathscr{M}$, except it draws a single sample S' of n i.i.d. examples from D and answers all queries with S'.*

If $\mathscr{M}$ answers all queries $q \in Q$ with ρ-replicable procedures, then

$$SD_{\Delta}([\mathscr{A},\mathscr{M}],[\mathscr{A},\mathscr{M}']) \leq (m-1)\rho,$$

where $SD_{\Delta}(D_1, D_2)$ denotes the statistical distance between distribtuions D_1 and D_2.

Proof. For $i \in [m]$, let $[\mathscr{A}, \mathscr{M}_i]$ denote the distribution on transcripts output by algorithm $\mathscr{A}$'s interaction with $\mathscr{M}_i$, where $\mathscr{M}_i$ is the analogous mechanism that draws new samples $S_1, \ldots, S_i$ for the first i queries, and reuses sample S_i for the remaining $m-i$ queries. Note that $\mathscr{M}' = \mathscr{M}_1$ and $\mathscr{M} = \mathscr{M}_m$.

For $i \in [m-1]$, consider distributions $[\mathscr{A}, \mathscr{M}_i]$ and $[\mathscr{A}, \mathscr{M}_{i+1}]$. We will bound the statistical distance by a coupling argument. Let $S_1, \ldots, S_{i+1}$ denote random variables describing the samples used, and let r denote the randomness used over the entire procedure. $[\mathscr{A}, \mathscr{M}_i]$ can be described as running the entire procedure (with randomness R) on $S_1, \ldots, S_{i-1}, S_{i+1}, S_{i+1}, \ldots, S_{i+1}$, and $[\mathscr{A}, \mathscr{M}_{i+1}]$ can be similarly described as running the entire procedure (with randomness R) on $S_1, \ldots, S_{i-1}, S_i, S_{i+1}, S_{i+1}, \ldots, S_{i+1}$.

These distributions are identical for the first $i-1$ queries and answers, so the i'th query q_i is identical, conditioned on using the same randomness. Both S_i and S_{i+1} are chosen by i.i.d. sampling from D, so $\mathbf{Pr}_{S_i, S_{i+1}, r}[\mathscr{A}(q_i, S_{i+1}; r) = A(q_i, S_i; r)] \geq 1 - \rho$ by replicability. Conditioned on both transcripts including the same $(i+1)$'th answer a_{i+1} (and continuing to couple S_{i+1} and r for both runs), the remaining queries and answers $q_{i+1}, a_{i+1}, \ldots, q_m, a_m$ is identical. Therefore, $SD_\Delta([\mathscr{A}, \mathscr{M}_i], [\mathscr{A}, \mathscr{M}_{i+1}]) \leq \rho$ for all $i \in [m-1]$. Unraveling, $SD_\Delta([\mathscr{A}, \mathscr{M}], [\mathscr{A}, \mathscr{M}']) \leq (m-1)\rho$. □

Remark 3.8.8. *This connection may be helpful for showing that replicability cannot be achieved efficiently in contexts where data reuse is not efficiently achievable.*

3.9 Appendix: Concentration of Sum of Vectors

In this Section, we use Azuma's inequality to prove a concentration bound on the sum of vectors from a distribution.

Let D be a distribution on $\mathbb{R}^n$. Let $\mathbf{v} = \{\mathbf{v_1}, \ldots, \mathbf{v_T}\} \in D^T$ be a random sample of T vectors from D with the following properties:

1. $\mathbb{E}_{\mathbf{v}\in\ D^T}[\sum_{i=1}^T \mathbf{v_i}] - \mathbb{E}_{v\in\ D}[v] = 0.$

2. $\forall v \in\ D$, $||v||_2 \leq c$.

The following lemma shows that the length of $\mathbf{v^1} + \mathbf{v^2} + \ldots + \mathbf{v^T}$ is tightly concentrated.

Lemma 3.5.10. *Let $D, \mathbf{v} \in D^T$ satisfy properties (1) and (2) above, and let $\mathbf{v^{\leq T}} = \sum_{i=1}^T \mathbf{v_i}$. Then for all $\Delta > 0$,*

$$\mathbf{Pr_v}[||\mathbf{v^{\leq T}}||_2 \geq \sqrt{T}(1+c/2)+\Delta] \leq e^{-\Delta^2/2c^2T}.$$

The intuition behind Lemma 3.5.10 is similar to the one-dimensional case, where D is a distribution over $(-1,1)$, $\mathbf{v} \in\ D^T$, and $\sum_{i=1}^T \mathbf{v_i}$ is concentrated around zero, with standard deviation $\sqrt{T}$. Let $\mathbf{v^{\leq i}}$ denote $\sum_{i=1}^i \mathbf{v_i}$. In the one-dimensional case, we can prove concentration of $\mathbf{v^{\leq T}}$ via a Chernoff or martingale argument since the expected value of $\mathbf{v^{\leq i}}$ (the sum of the first i numbers) is equal to $\mathbf{v^{\leq i-1}}$. However for the higher dimensional case, $\mathbf{v^{\leq i}}$ is now the sum of the first i vectors, and it is in general not the case that the expected length of $\mathbf{v^{\leq i}}$ is equal or even not much larger than the length of $\mathbf{v^{\leq i-1}}$. However, if the length of $\mathbf{v^{\leq i-1}}$ is sufficiently large (greater than $\sqrt{T}$), then $\mathbb{E}[||\mathbf{v^{\leq i}}||_2 \mid \mathbf{v^{\leq i-1}}]$ can be upper bounded (approximately) by $||\mathbf{v^{\leq i-1}}||_2 + 1/\sqrt{T}$. Therefore, if we want to bound the probability that the length of $\mathbf{v^{\leq T}}$ is large (at least $\sqrt{T}+\Delta$), there must be some time t such that the vector $\mathbf{v^{\leq t}}$ is outside of the ball of radius $\sqrt{T}$ around the origin, and never returns. So we can bound the probability that $||\mathbf{v^{\leq T}}||_2 \geq \sqrt{t}+\Delta$, by considering the sequence of random variables $\mathbf{x^{\leq t}}, \ldots, \mathbf{x^{\leq T}}$ such that $\mathbf{x^{\leq t}}$ is equal to the length of $\mathbf{v^{\leq t}}$, and for each $t' \geq t$, $\mathbf{x^{\leq t'}}$ is the length of $\mathbf{v^{\leq t'}}$ minus a correction term (so that we can upper bound $\mathbb{E}[\mathbf{x^{\leq t'+1}} \mid \mathbf{x^{\leq t'}}]$ by $\mathbf{x^{\leq t'}}$.) We will show that $\mathbf{x^{\leq t}}, \ldots, \mathbf{x^{\leq T}}$ is a supermartingale where $|\mathbf{x^{\leq t'+1}} - \mathbf{x^{\leq t'}}|$ is bounded by a constant, and then the concentration inequality will follow from Azuma's Lemma.

Definition 3.9.1. *Let D be a distribution over $\mathbb{R}^n$ satisfying the above two properties.*

1. *Let $\mathbf{v} = \{\mathbf{v_1}, \ldots, \mathbf{v_{T'}}\} \in\ D^{T'}$ be a sequence of $T' \leq T$ random variables, and let $\mathbf{v_0} \in \mathbb{R}^n$ have length $\sqrt{T}$. For $0 \leq i \leq T'$, let $\mathbf{v^{\leq i}} = \sum_{i=1}^{T'} \mathbf{v_i}$.*

2. *The stopping time $\tau \in [T']$ (with respect to $\{\mathbf{v}^{\leq \mathbf{i}}\}$) is equal to:*

$$min\{\{i \in [T'] \mid ||\mathbf{v}^{\leq \mathbf{i}}||_2 < \sqrt{T}\} \cup \{T'\}\}.$$

That is, τ is the first time i such that the length of $\mathbf{v}^{\leq \mathbf{i}}$ drops below $\sqrt{T} + \frac{i}{3\sqrt{T}}$ (and otherwise $\tau = T'$).

3. *For each $i \in [T']$, we define the sequence of random variables $\mathbf{x}^{\leq \mathbf{0}}, \mathbf{x}^{\leq \mathbf{1}}, \mathbf{x}^{\leq \mathbf{2}}, \ldots, \mathbf{x}^{\leq \mathbf{T}'}$ where $\mathbf{x}^{\leq \mathbf{0}} = ||\mathbf{v}^{\mathbf{0}}||_2 = \sqrt{T}$, and for all $i \geq 1$, $\mathbf{x}^{\leq \mathbf{i}}$ will be the adjusted length of the first i vectors, $||\mathbf{v}^{\leq \mathbf{i}}||$ with stopping condition τ:*

$$\mathbf{x}^{\leq \mathbf{i}} = \begin{cases} ||\mathbf{v}^{\leq \mathbf{i}}||_2 - \frac{ci}{2\sqrt{T}} & \text{if } \tau > i \\ \mathbf{x}^{\leq \tau} & \text{otherwise} \end{cases}$$

Claim 3.9.2. *The sequence of random variables $\mathbf{x}^{\leq \mathbf{1}}, \ldots, \mathbf{x}^{\leq \mathbf{T}'}$ is a supermartingale.*

Proof. We need to show that for every $i \in [T']$, $\mathbb{E}[\mathbf{x}^{\leq \mathbf{i}} \mid \mathbf{x}^{\leq \mathbf{i-1}}] \leq \mathbf{x}^{\leq \mathbf{i-1}}$. Fix $i \in [T']$; if $\tau \leq i-1$ then $\mathbf{x}^{\leq \mathbf{i}} = \mathbf{x}^{\leq \mathbf{i-1}}$ so the condition holds. Otherwise assume that $\tau \geq i$. Since

$$\mathbb{E}[\mathbf{x}^{\leq \mathbf{i}} \mid \mathbf{x}^{\leq \mathbf{i-1}}] = \mathbb{E}[\mathbf{x}^{\leq \mathbf{i}} \mid \mathbf{v}^{\leq \mathbf{i-1}}] = \mathbb{E}[||\mathbf{v}^{\leq \mathbf{i-1}} + \mathbf{v_i}||_2] - \frac{ci}{2\sqrt{T}}$$

and $\mathbf{x}^{\leq \mathbf{i-1}} = ||\mathbf{v}^{\leq \mathbf{i-1}}||_2 - \frac{c(i-1)}{2\sqrt{T}}$, it suffices to show that $\mathbb{E}[||\mathbf{v}^{\leq \mathbf{i-1}} + \mathbf{v_i}||_2 \leq ||\mathbf{v}^{\leq \mathbf{i-1}}||_2 + \frac{c}{2\sqrt{T}}$.

To prove this, we can write $\mathbf{v_i} = \mathbf{v_i}^{||} + \mathbf{v_i}^{\perp}$ where $\mathbf{v_i}^{||}$ is the component of $\mathbf{v_i}$ in the direction of $\mathbf{v}^{\leq \mathbf{i-1}}$, and $\mathbf{v_i}^{\perp}$ is the orthogonal component. Since the expected length of $\mathbf{v}^{\leq \mathbf{i-1}} + \mathbf{v_i}^{||}$ is equal to the length of $\mathbf{v}^{\leq \mathbf{i-1}}$ (by property 1), we just have to show that the expected length of $\mathbf{v}^{\leq \mathbf{i-1}} + \mathbf{v_i}^{\perp}$ is at most $\frac{c}{2\sqrt{T}}$. Since $\mathbf{v_i}$ has length at most c, so does $\mathbf{v_i}^{\perp}$, so we have:

$$\mathbb{E}[||\mathbf{v}^{\leq \mathbf{i-1}} + \mathbf{v_i}^{\perp}||_2] \leq (||\mathbf{v}^{\leq \mathbf{i-1}}||_2^2 + c)^{1/2} \leq \frac{c}{2\sqrt{T}}$$

where the last inequality holds since $\tau \geq i$ implies $||\mathbf{v}^{\leq \mathbf{i-1}}||_2 \geq \sqrt{T}$. $\square$

Claim 3.9.3. *For all i,* $|\mathbf{x}^{\leq \mathbf{i}} - \mathbf{x}^{\leq \mathbf{i-1}}| \leq c$.

Proof. Since $\mathbf{v_i}$ has length at most c the absolute value of the difference between $||\mathbf{v}^{\leq \mathbf{i}}||_2$ and $||\mathbf{v}^{\leq \mathbf{i-1}}||_2$ is at most 2. The claim easily follows since $\mathbf{x}^{\leq \mathbf{i}} = ||\mathbf{v}^{\leq \mathbf{i}}|| + \frac{ci}{2\sqrt{T}}$. $\square$

The above two Claims together with Azuma's inequality gives:

$$Pr[|\mathbf{x}^{\leq \mathbf{T}'} - \mathbf{x}^{\leq \mathbf{0}}| \geq \Delta] \leq e^{-\Delta^2/2c^2T}.$$

Proof. (of Lemma 3.5.10)

In order for $\mathbf{v}^{\leq \mathbf{T}}$ to have length at least $\sqrt{T}(1+c/2)+\Delta$, there must be some largest time $t \in [T]$ such that $||\mathbf{v}^{\leq \mathbf{t}}||_2 \in (\sqrt{T}, \sqrt{T}+1]$. That is, at all times $t' \geq t$ the vector $\mathbf{v}^{\leq \mathbf{t}'}$ is outside the ball of radius $\sqrt{T}$. Thus by the above argument, the random variables $\mathbf{x}^{\leq \mathbf{i}}{}_{i=t}^{T}$ are a supermartingale where the absolute value of the difference between successive variables is at most c, and by Azuma, $\mathbf{Pr}[\mathbf{x}^{\leq \mathbf{T}} \geq \sqrt{T}+\Delta]$ is at most $e^{-\Delta^2/2c^2T}$. Since $\mathbf{x}^{\leq \mathbf{T}} = ||\mathbf{v}^{\leq \mathbf{T}}||_2 - \frac{Tc}{2\sqrt{T}} = ||\mathbf{v}^{\leq \mathbf{T}}||_2 - \frac{\sqrt{T}c}{2}$, $\mathbf{Pr}[||\mathbf{v}^{\leq \mathbf{T}}||_2 \geq \sqrt{T}(1+c/2)+\Delta]$ is at most $e^{-\Delta^2/2c^2T}$.

$\square$

Acknowledgements.

The authors would like to thank Mark Bun, Cynthia Dwork, Max Hopkins, Pasin Manurangsi, Rahul Santhanam, Geelon So, Ryan Williams, and TPDP 2021 participants for interesting discussions.

Chapter 3, in full, is a reprint of the material as it appeared on Arxiv at https://arxiv.org/abs/2201.08430v2. This material is the full version of the paper that appeared in the Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing 2022. Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sorrell, Jessica. "Reproducibility in Learning". For this dissertation,

the term “reproducibility” was modified to “replicability” where appropriate, and minor formatting edits were made to improve readability. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Massart Boosting

We study the problem of boosting the accuracy of a weak learner in the (distribution-independent) PAC model with Massart noise. In the Massart noise model, the label of each example x is independently misclassified with probability $\eta(x) \leq \eta$, where $\eta < 1/2$. The Massart model lies between the random classification noise model and the agnostic model. Our main positive result is the first computationally efficient boosting algorithm in the presence of Massart noise that achieves misclassification error arbitrarily close to η. Prior to our work, no non-trivial booster was known in this setting. Moreover, we show that this error upper bound is best possible for polynomial-time black-box boosters, under standard cryptographic assumptions. Our upper and lower bounds characterize the complexity of boosting in the distribution-independent PAC model with Massart noise. As a simple application of our positive result, we give the first efficient Massart learner for unions of high-dimensional rectangles.

4.1 Introduction

4.1.1 Background and Motivation

Boosting is a general learning technique that combines the outputs of a weak base learner — a learning algorithm with low but non-trivial accuracy — to obtain a hypothesis of higher accuracy. Boosting has been extensively studied in machine learning and statistics since initial work by Schapire [Sch90]. The reader is referred to [Sch03] for an early survey from the theoretical machine learning community, [BH07] for a statistics perspective, and [SF12] for a book on the topic. Here we study boosting in the context of learning classes of Boolean functions with a focus on Valiant's distribution-independent PAC model [Val84]. During the past three decades, several efficient boosting procedures have been developed in the *realizable* PAC model, i.e., when the data is consistent with a function in the target class. On the other hand, boosting in the presence of *noisy data* remains less understood.

In this work, we study the complexity of boosting in the presence of *Massart noise*. In the Massart (or bounded noise) model, the label of each example x is flipped independently with probability $\eta(x) \le \eta$, for some parameter $\eta < 1/2$. The flipping probability $\eta(x)$ is bounded but is unknown to the learner and can depend on the example x in a potentially adversarial manner. Formally, we have the following definition.

Definition 4.1.1 (PAC Learning with Massart Noise)**.** *Let $\mathscr{C}$ be a concept class over $X = \mathbb{R}^n$, D_x be any fixed but unknown distribution over X, and $0 \le \eta < 1/2$ be the noise parameter. Let $f \in \mathscr{C}$ be the unknown target concept. A* noisy example oracle, $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, *works as follows: Each time $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ is invoked, it returns a labeled example (x, y), where $x \sim D_x$, $y = f(x)$ with probability $1 - \eta(x)$ and $y = -f(x)$ with probability $\eta(x)$, for an* unknown *function $\eta(x) : X \to [0, \eta]$. Let D denote the joint distribution on (x, y) generated by the above oracle. A learning algorithm is given i.i.d. samples from D and its goal is to output a hypothesis h such that with high probability the misclassification error $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y]$ is as small as possible. We will use $\mathrm{OPT} \overset{\mathrm{def}}{=} \inf_{g \in \mathscr{C}} \mathbf{Pr}_{(x,y)\sim D}[g(x) \neq y]$ to denote the optimal misclassification error.*

Background on Massart Noise.

The Massart model is a natural semi-random input model that is more realistic and robust than random classification noise. Noise can reflect computational difficulty or ambiguity, as well as random factors. For example, a cursive "e" might be substantially more likely to be misclassified as "a" than an upper case Roman letter. Massart noise allows for these variations in misclassification rates, while not requiring precise knowledge of which instances are more likely to be misclassified. That is, algorithms that learn in the presence of Massart noise are likely to be less brittle than those that depend on uniformity of misclassification noise. Agnostic learning is of course even more robust, but unfortunately, it can be computationally infeasible to design agnostic learners for many applications.

In its above form, the Massart noise model was defined in [MN06]. An essentially equivalent noise model had been defined in the 80s by Sloan and Rivest [Slo88, Slo92, RS94, Slo96], and a very similar definition had been considered even earlier by Vapnik [Vap82]. The Massart model is a generalization of the Random Classification Noise (RCN) model [AL88] and appears to be easier than the agnostic model [Hau92, KSS94]. Perhaps surprisingly, until very recently, no progress had been made on the efficient, distribution-free PAC learnability in the presence of Massart noise for any non-trivial concept class.

In more detail, the existence of an efficient distribution-independent PAC learning algorithm with non-trivial error guarantee for any concept class in the Massart model had been posed as an open question in a number of works, including [Slo88, Coh97], and was highlighted in A. Blum's FOCS'03 tutorial [Blu03]. Recent work [DGT19] made the first algorithmic progress in this model for the concept class of halfspaces. Specifically, [DGT19] gave a polynomial-time learning algorithm for Massart halfspaces with misclassification error $\eta + \varepsilon$. We note that the information-theoretically optimal error is $\mathrm{OPT} = \mathbb{E}_{x \sim D_x}[\eta(x)]$, which is at most η but could be much smaller. Thus, the error achieved by the aforementioned algorithm can be very far from optimal. Very recent follow-up work [CKMY20] showed that obtaining the optimal error of $\mathrm{OPT} + \varepsilon$ for halfspaces requires super-polynomial time in Kearns' Statistical Query (SQ) model [Kea98]. Contemporaneous to the results of the current paper, [DK20] showed an SQ lower bound ruling out any constant factor or even polynomial factor approximation for this problem. The approximability of learning Massart halfspaces remains a challenging open problem of current investigation.

Comparison to RCN and Agnostic Noise.

Random Classification Noise (RCN) [AL88] is the special case of Massart noise where the label of each example is independently flipped with probability *exactly* $\eta < 1/2$. RCN is a fundamentally easier model algorithmically. Roughly speaking, RCN is predictable which allows us to cancel out the effect of the noise on any computation, in expectation. A formalization of this intuition is that *any* Statistical Query (SQ) algorithm [Kea98] is automatically robust to RCN. This fact inherently fails in the presence of Massart noise. Roughly speaking, the ability of the Massart adversary to choose *whether* to flip a label and if so, with what probability, makes this model algorithmically challenging. Moreover, the uniform noise assumption in the RCN model is commonly accepted to be unrealistic, since in practical scenarios some instances are harder to classify than others [FV13]. For example, in the setting of human annotation noise [BK09], it has been observed that the flipping probabilities are not uniform.

The agnostic model [Hau92, KSS94] is the most challenging noise model in the literature, in which an adversary can arbitrarily flip an $\mathrm{OPT} < 1/2$ fraction of the labels. It is well-known that (even weak) learning in this model is computationally intractable for simple concept classes, including halfspaces [Dan16].

The Massart model can be viewed as a reasonable compromise between RCN and the agnostic model, in the sense that it is a realistic noise model that may allow for efficient algorithms in settings where agnostic learning is computationally hard. This holds in particular for the important concept class of halfspaces. As already mentioned, even weak learning of halfspaces is hard in the agnostic model [Dan16], while an efficient Massart learner with non-trivial accuracy is known [DGT19].

Boosting With Noisy Data.

An important research direction, which was asked in Schapire's original paper [Sch90], is to design boosting algorithms in the presence of noisy data. This broad question has been studied in the past two decades by several researchers. See Section 4.1.4 for a detailed summary of related work. Specifically, prior work has obtained efficient boosters for RCN [KS03] and agnostic noise [Ser03, Fel10]. It should be emphasized that these prior works do not immediately extend to give boosters for the Massart noise setting. For example, while the agnostic model is stronger than the Massart model, an agnostic booster does not imply a Massart booster, as it relies on a much stronger assumption — the existence of a weak *agnostic* learner. That is, the complexity of noisy boosting is not "monotone" in the difficulty of the underlying noise model. More broadly, it turns out that the complexity of boosting with inconsistent data, and the underlying boosting algorithms, crucially depend on the choice of the noise model.

In this work, we ask the following question:

Can we develop efficient boosting algorithms for PAC learning with Massart noise?

Our focus is on the distribution-independent setting. Given a distribution-independent Massart weak learner for a concept class $\mathscr{C}$, we want to design a distribution-independent Massart learner for $\mathscr{C}$ with high(er) accuracy. Prior to this work, no progress had been made on this front. *In this paper, we resolve the complexity of the aforementioned problem by providing (1) an efficient boosting algorithm and (2) a matching computational lower bound on the error rate of any black-box booster.*

This work is the first step of the broader agenda of developing a general algorithmic theory of boosting for other "benign" semi-random noise models, lying between random and fully adversarial corruptions.

4.1.2 Our Results

Our main result is the first computationally efficient boosting algorithm for distribution-independent PAC learning in the presence of Massart noise that guarantees misclassification arbitrarily close to η, where η is the upper bound on the Massart noise rate. To state our main result, we will require the definition of a Massart weak learner (see Definition 4.2.5 for additional details).

Definition 4.1.2 (Massart Weak Learner)**.** *Let $\alpha, \gamma \in (0, 1/2)$. An (α, γ)-Massart weak learner* `WkL` *for concept class $\mathscr{C}$ is an algorithm that, for any distribution D_x over examples, any function $f \in \mathscr{C}$, and any noise function $\eta(x)$ with noise bound $\eta < 1/2 - \alpha$, outputs a hypothesis h that with high probability satisfies $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y] \leq 1/2 - \gamma$, where D is the joint Massart noise distribution.*

We prove two versions of our main algorithmic result. In Section 4.3, we present our Massart noise-tolerant booster (Algorithm 12). In Appendix 4.6, we analyze this algorithm and show that it converges within $O(1/(\eta\gamma^2))$ rounds of boosting (Theorem 4.6.1). In Section 4.7, we give a more careful analysis of convergence, showing that the same algorithm in fact converges in $O(\log^2(1/\eta)/\gamma^2)$ rounds (Theorem 4.7.1). In fact, the latter upper bound is nearly optimal for distribution-independent boosters (see, e.g., Chapter 13 of [SF12]). We now state our main result:

Theorem 4.1.3 (Main Result)**.** *There exists an algorithm* `Massart-Boost` *that for every concept class $\mathscr{C}$, given samples to a Massart noise oracle $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, where $f \in \mathscr{C}$, and black-box access to an (α, γ)-Massart weak learner* `WkL` *for $\mathscr{C}$,* `Massart-Boost` *efficiently computes a hypothesis h that with high probability satisfies $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y] \leq \eta(1 + O(\alpha))$. Specifically,* `Massart-Boost` *makes $O(\log^2(1/\eta)/\gamma^2)$ calls to* `WkL` *and draws*

$$\mathrm{polylog}(1/(\eta\gamma))/(\eta\gamma^2)\, m_{\texttt{WkL}} + \mathrm{poly}(1/\alpha, 1/\gamma, 1/\eta)$$

samples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, where $m_{\texttt{WkL}}$ is the number of samples required by `WkL`.

Prior to this work, no such boosting algorithm was known for PAC learning with Massart noise. Moreover, as we explain in Section 4.1.4, previous noise-tolerant boosters do not extend to the Massart noise setting. In Section 4.1.3, we provide a detailed overview of our new algorithmic ideas to achieve this.

Some additional comments are in order. First, we note that the $\eta + \varepsilon$ error guarantee achieved by our efficient booster can be far from the information-theoretic minimum of $\mathrm{OPT} + \varepsilon$. The error guarantee of our generic booster matches the error guarantee of the best known polynomial-time learning algorithm for Massart halfspaces [DGT19]. Interestingly, the learning algorithm of [DGT19] can be viewed as a specialized boosting algorithm for the class of halfspaces, in which the halfspace structure is used to downweight specific regions on which the current classifier achieves high accuracy. Theorem 4.1.3 is a broad generalization of this result that applies to *any* concept class. This connection was one of the initial motivations for this work.

A natural question is whether the error upper bound achieved by our booster can be improved. Perhaps surprisingly, we show that our guarantee is best possible for black-box boosting algorithms (under cryptographic assumptions). Specifically, we have the following theorem:

Theorem 4.1.4 (Lower Bound on Error of Black-Box Massart Boosting)**.** *Assuming one-way functions exist, no polynomial-time boosting algorithm, given black-box access to an (α, γ)-Massart weak learner, can output a hypothesis h with misclassification error $\mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y] \leq \eta(1 + o(\alpha))$, where η is the upper bound on the Massart noise rate. In particular, this statement remains true on Massart distributions with optimal misclassification error* $\mathrm{OPT} \ll \eta$.

The reader is referred to Theorem 4.4.1 for a detailed formal statement. Our lower bound establishes that the error upper bound achieved by our boosting algorithm is best possible. It is worth pointing out a related lower bound shown in [KS03] in the context of RCN. Specifically, [KS03] showed that any efficient black-box booster tolerant to RCN must incur error at least η (with respect to the target function f), where η is the RCN noise rate. Since RCN is the special case of Massart noise where $\eta(x) = \eta$ for all x, the lower bound of [KS03] suggests a lower bound of OPT for black-box Massart boosting. Importantly, our lower bound is significantly stronger, as it shows a lower bound of η, even when OPT is much smaller than η.

Intriguingly, Theorem 4.1.4 shows that the error guarantee of the [DGT19] learning algorithm for Massart halfspaces cannot be improved using boosting, and ties with recent work [DK20] providing evidence that learning with Massart noise (within error relative to OPT) is computationally hard.

Application: Massart Learning of Unions of Rectangles.

As an application of Theorem 4.1.3, we give the first efficient learning algorithm for unions of (axis-aligned) rectangles in the presence of Massart noise. Interestingly, weak agnostic learning of a single rectangle is computationally hard in the agnostic model (see, e.g., [FGRW09]). Recall that a rectangle $R \in \mathbb{R}^d$ is an intersection of inequalities of the form $x \cdot v < t$, where $v \in \{\pm e_j : j \in [d]\}$ and $t \in \mathbb{R}$. Formally, we show:

Theorem 4.1.5. *There exists an efficient algorithm that learns unions of k rectangles on $\mathbb{R}^d$ with Massart noise bounded by η. The algorithm has sample complexity $kd^{O(k)}\mathrm{poly}(1/\varepsilon, 1/\eta)$, runs in time $(kd^k/\varepsilon)^{O(k)}\mathrm{poly}(1/\eta)$, and achieves misclassification error $\eta + \varepsilon$, for any $\varepsilon > 0$.*

See Theorem 4.9.4 for a more detailed statement. Theorem 4.1.5 follows by an application of Theorem 4.1.3 coupled with a simple weak learner for unions of rectangles that we develop. Our weak learner finds a rectangle entirely contained in the negative region to gain some advantage over a random guess.

It is worth pointing out that the Massart SQ lower bound of [CKMY20] applies to learning monotone conjunctions. This rules out efficient SQ algorithms with error $\mathrm{OPT} + \varepsilon$, even for a single rectangle.

4.1.3 Overview of Techniques

In this section, we provide a brief overview of our approach.

Boosting Algorithm Approach.

We start with our Massart noise boosting algorithm. Let D be the Massart distribution $\mathtt{Mas}\{f, D_x, \eta(x)\}$ from which our examples are drawn. The distribution D_x on examples is fixed but arbitrary and the function $\eta(x)$ is a Massart noise function satisfying $\eta(x) \leq \eta < 1/2$ with respect to the target function $f \in \mathscr{C}$. As is standard in distribution-independent boosting, our boosting algorithm adaptively generates a sequence of distributions $D^{(i)}$, invokes the weak learner on samples from these distributions, and incrementally combines the corresponding weak hypotheses to obtain a hypothesis with higher accuracy.

The technical challenge of distribution-independent boosting is the adaptive generation of new distributions $D^{(i)}$ that effectively use the weak learner to acquire new and useful information about the target function f. To see why this requires some care, consider an adversarial weak learner that attempts to give the booster as little information about f as possible, while still satisfying its definition as a weak learner. Such an adversarial weak learner might, whenever possible, produce hypotheses that correctly classify the same, small set of examples P, while classifying all other examples randomly. Assuming the function f is balanced, and the intermediate distributions $D^{(i)}$ assign probability at least γ to P, these adversarial hypotheses will have accuracy $1/2+\gamma$ on their corresponding distributions, while providing no new information about the target function to the booster. To thwart this behavior, the booster must eventually restrict its distributions to assign sufficiently small probability to P to ensure that the weak learner can no longer meet its promised accuracy lower-bound by correctly classifying only the set P. In this way, the booster can force the weak learner to output hypotheses correlating with f on other subsets of its domain. Under reasonable conditions on the specific strategy for reweighting distributions, boosters that incrementally decrease the probability assigned to examples as they are more frequently correctly classified by weak hypotheses are known to eventually converge to high-accuracy hypotheses, by reduction to iterated two-player zero-sum games [FS97]. This general approach to reweighting intermediate distributions is common to all distribution-independent boosters, even in the noiseless setting.

Our booster follows the smooth boosting framework [Ser03] with some crucial modifications that are necessary to handle Massart noise. A smooth boosting algorithm generates intermediate distributions that do not put too much weight on any individual point, and so do not compel the weak learner to generate hypotheses having good correlation only with noisy examples. This makes the smooth boosting framework a natural starting point for the design of a Massart noise-tolerant booster, though smoothness of the intermediate distributions alone is not a sufficient condition for preservation of the Massart noise property.

To see why, note that to preserve the Massart noise property of the intermediate distributions, it is not enough to enforce an upper bound on the probability that any (potentially noisy) example can be assigned. We require an upper bound on the *relative* probabilities of sampling noisy and correct labels for a given point, to ensure we always have a noise upper bound $\eta^{(i)} < 1/2$. This seems to suggest that preserving the Massart noise property requires a corresponding lower bound on the probability assigned to any given example, so that we do not inadvertently assign more probability to $(x, -f(x))$ than $(x, f(x))$. This is at odds with our strategy for making use of an adversarial weak learner, since guaranteeing progress requires that our distributions can assign arbitrarily small probability to some examples. So, we must use alternative techniques to manage noise.[1]

The fix for this is to simply not include examples (x, y) in the support of $D^{(i)}$ whenever including them could violate the Massart noise property or permit an adversarial weak learner to tell us only what we already know. If many of the weak hypotheses obtained by our booster agree with the label y on x, then we learn little from a marginal weak hypothesis that agrees with y on x. So, we exclude (x, y) from the support of $D^{(i)}$. We must also symmetrically exclude $(x, -y)$, otherwise we risk violating the Massart noise property for $D^{(i)}$, since we have assigned no probability to (x, y), and it may be the case that $-y \neq f(x)$. Withholding these examples allows the booster to get new information from the weak learner in each round, without ever invoking it on an excessively noisy sample.

[1]We note that vanilla smooth boosting has been shown to succeed in the agnostic model. Interestingly, the above subtle issue for Massart boosting does not arise in agnostic boosting, since agnostic noise is easy to preserve.

This balance comes at the cost of updates from the weak learner on withheld examples. This may not seem to pose a significant problem for our booster at first. After all, points on which many hypotheses agree are points where our algorithm is already fairly confident about the correct value of $f(x)$. Unfortunately, this confidence may not be sufficiently justified to ensure an $\eta+\varepsilon$ error at the end of the day. In order to deal with this, our algorithm will need to make use of one further idea. We directly check the empirical error of our aggregated hypotheses on the set of withheld examples. If this error is too large (i.e., larger than $\eta+\varepsilon$), we conclude we are "overconfident" and have more to learn about the withheld examples after all. Since even an adversarial weak learner will give us new information about these examples in expectation, we include them in subsequent distributions, with appropriate upper and lower bounds on their probabilities to preserve the Massart noise property. If the empirical error is not too large, we are content to learn nothing new about these examples, and so continue to withhold them for the next round of boosting.

Overall, our algorithm will alternate between the two steps of applying the weak learner to an appropriately reweighted version of the underlying distribution, and checking the consistency of our hypotheses with the set of withheld examples. Each step will allow us to make progress in the sense of decreasing a relevant potential function. We iterate these steps until almost all points are consistently being withheld from the weak learner. Once we reach this condition, we will have produced a hypothesis with appropriately small error, and can terminate the algorithm. We analyze the convergence of our algorithm to a low-error hypothesis via a novel potential function that can be easily adapted to analyze other smooth boosting algorithms.

Error Lower Bound.

We show that no "black-box" generic boosting algorithm for Massart noise can have significantly better error than that for our algorithm, i.e., $\eta+\Theta(\eta\alpha)$. While this seemingly matches the lower bound for RCN boosting from [KS03], the RCN bound only implies a lower bound for RCN weak learners in the special case of Massart noise when $\eta = \mathrm{OPT}$. We show a similar lower bound in the Massart noise setting for a small but polynomial value of OPT. That is, Massart noise boosting algorithms cannot be improved even when only a very small fraction of instances are actually noisy.

To prove our lower bound, we consider a situation where the function to be learned is highly biased, and there is a tiny fraction of inputs with the majority value that are noisy and indistinguishable from non-noisy inputs. If the distribution queried by a boosting algorithm does not reweight values in some way to favor the minority answer, an uncooperative weak learner can return the majority answer and have advantage γ. On the other hand, if the boosting algorithm does reweight values, it risks adding too much noise to the small fraction of already noisy examples, violating the Massart condition. Specifically, we exhibit an adversarial weak learner `rWkL` that has a stability property called *replicability*. `rWkL` returns a hypothesis h that outputs the maximum likelihood label for each heavy-hitter of given distribution D' and outputs a constant value for non-heavy-hitters. Using replicability, we argue that i) boosting with `rWkL` can be efficiently simulated without knowing the function f and ii) `rWkL` satisfies the definition of a Massart noise weak learner. We conclude that a black-box boosting algorithm must be able to efficiently learn pseudorandom functions in order to extract useful information from `rWkL`.

4.1.4 Comparison with Prior Work

The literature on boosting is fairly extensive. Since the initial work of Schapire [Sch90], boosting has become one of the most studied areas in machine learning — encompassing both theory and practice. Early boosting algorithms [Sch90, Fre95, FS97] were not tolerant in the presence of noisy data. In this section, we summarize the most relevant prior work with a focus on boosting techniques that have provable noise tolerance guarantees.

Efficient boosting algorithms have been developed for PAC learning in the agnostic model [Hau92, KSS94] and in the presence of Random Classification Noise (RCN) [AL88]. The notion of agnostic boosting was introduced in [BDLM01]. Subsequently, a line of work [Ser03, Gav03, KMV08, KK09, Fel10] developed efficient agnostic boosters with improved error guarantees, culminating in the optimal bound. These agnostic boosters rely on one of two techniques: smooth boosting, introduced in [Ser03], or boosting via branching programs, developed in [MM02]. While both of these techniques have been successful in the agnostic model, known RCN-tolerant boosters from [KS03, LS05, LS08] are all based on the branching program technique [MM02]. In the following paragraphs, we briefly summarize these two techniques.

Smooth boosting [Ser03] is a technique that produces intermediate distributions which do not assign too much weight on any single example. The technique was inspired by Impagliazzo's hard-core set constructions in complexity theory [Imp95] (see also [KS99, Hol05, BHK09]) and is closely related to convex optimization. Roughly speaking, smooth boosting algorithms are reminiscent of first-order methods in convex optimization. Smooth boosting methods have been shown to be tolerant to agnostic noise [Ser03, Gav03, KK09, Fel10]. Interestingly, [LS10] established a lower bound against potential-based convex boosting techniques in the presence of RCN. While we do not prove any relevant theorems here, we believe that our technique can be adapted to give an efficient booster in the presence of RCN.

Another important boosting technique relies on branching programs [MM02]. The main idea is to iteratively construct a branching program in which each internal node is labeled with a hypothesis generated by some call to the weak learner. This technique is quite general and has led to noise tolerant boosters for both RCN [KS03] (see also [LS05, LS08] for refined and simplified boosters relying on this technique) and agnostic noise [KMV08]. Roughly speaking, the branching programs methodology leads to "non-convex algorithms" and is quite flexible.

It is worth pointing out that the aforementioned branching program-based boosters do not succeed with Massart noise in their current form. Specifically, the RCN booster in [KS03] crucially relies on the uniform noise property of RCN, which implies that agreement with the true target function is proportional to agreement with the observed labels. On the other hand, for the agnostic booster of [KMV08], the generated distributions on which the weak learner is invoked do not preserve the Massart noise property — a crucial requirement for any such booster. While it should be possible to adapt the branching program technique to work in the Massart noise model, we believe that the smooth-boosting technique developed in this paper leads to simpler and significantly more efficient boosters that are potentially practical.

Finally, we acknowledge existing work developing efficient learning algorithms for Massart halfspaces (and related noise models) in the *distribution-specific* PAC model [ABHU15, ABHZ16, ZLC17, DKTZ20a, ZSA20, DKTZ20b, DKK$^+$20]. These works are technically orthogonal to the results of this paper, as they crucially leverage a priori structural information about the distribution on examples (e.g., log-concavity).

4.1.5 Organization

The structure of this paper is as follows: Section 4.2 contains preliminary definitions and fixes notation. In Section 4.3 and Appendix 4.6 we present our Massart noise-tolerant boosting algorithm. In Appendix 4.7, we prove an improved round complexity for our booster. In Section 4.4 and Appendix 4.8, we show that the error achieved by our booster is optimal by proving a lower-bound on the error of any black-box Massart-noise-tolerant booster. In Appendix 4.9, we give an application of our boosting algorithm to learning unions of rectangles. In Appendix 4.10, we give a glossary of symbols.

4.2 Preliminaries

Throughout this work, we are primarily concerned with large finite domains $\mathscr{X}$. For a distribution D over $\mathscr{X}$, let $\mathrm{supp}(D)$ be the set of all $x \in \mathscr{X}$ such that $D(x) \neq 0$. Let $S \,||\, z$ denote appending z to sequence S. For $f : \mathscr{X} \to \mathbb{R}$, $x \in \mathscr{X}$, we define $\mathrm{sign}(f)(x) = 1$ if $f(x) \geq 0$ and -1 otherwise.

4.2.1 Massart Noise Model

Let $\mathscr{C}$ be a class of Boolean-valued functions over some domain $\mathscr{X}$, and let D_x be a distribution over $\mathscr{X}$. Let $f \in \mathscr{C}$ be an unknown target function, and let $\eta(x) : \mathscr{X} \to [0, 1/2)$ be an unknown function.

Definition 4.2.1 (Noisy Example Oracle)**.** *When invoked, noisy example oracle* $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ *produces a labeled example* (x, y) *as follows:* $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ *draws* $x \sim D_x$*. With probability* $\eta(x)$*, it returns* $(x, -f(x))$*, and otherwise returns* $(x, f(x))$*.*

Definition 4.2.2 (Massart Distribution)**.** *Massart distribution* $D = \texttt{Mas}\{f, D_x, \eta(x)\}$ *over* $(\mathscr{X}, \pm 1)$ *is the distribution induced by sampling from* $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$*.*

We refer to $\eta(x)$ in this context as the *Massart noise function*. We say a Massart distribution D has *noise rate* η if $\eta(x) \leq \eta$ for all $x \in \mathrm{supp}(D_x)$. The *noise bound* of a Massart noise function is η if $\max_{x \in \mathrm{supp}(D_x)} \eta(x) = \eta$. We emphasize that this model restricts the noise bound to be $\eta < 1/2$.

4.2.2 Learning under Massart Noise

Let $f : \mathscr{X} \to \{\pm 1\}$ be a function in concept class $\mathscr{C}$. Let $D = \texttt{Mas}\{f, D_x, \eta(x)\}$ be a Massart distribution over $\mathscr{X}$.

Definition 4.2.3 (Misclassification Error, Function Error)**.** *The misclassification error of hypothesis* $h : \mathscr{X} \to \{\pm 1\}$ *over* D *is* $\mathrm{err}_{0\text{-}1}^{D}(h) = \mathbf{Pr}_{(x,y)\sim D}[h(x) \neq y]$*. The error of hypothesis* $h : \mathscr{X} \to \{\pm 1\}$ *with respect to* f *over* D *is* $\mathrm{err}_{0\text{-}1}^{D_x, f}(h) = \mathbf{Pr}_{x \sim D_x}[h(x) \neq f(x)]$*.*

Definition 4.2.4 (Advantage)**.** *Hypothesis* $h : \mathscr{X} \to \{\pm 1\}$ *has advantage* $\gamma > 0$ *against distribution* D *if* $\mathrm{err}_{0\text{-}1}^{D}(h) \leq 1/2 - \gamma$*.*

We use the notation $\mathrm{adv}^{D}(h)$ to denote the largest $\gamma \in [0, 1/2]$ for which $\mathrm{err}_{0\text{-}1}^{D}(h) \leq 1/2 - \gamma$.

4.2.3 Boosting and Weak Learners

Definition 4.2.5 (Massart Noise Weak Learner). *Let $\mathscr{C}$ be a concept class of functions $f : \mathscr{X} \to \{\pm 1\}$. Let $\alpha \in [0, 1/2)$. Let $\gamma : \mathbb{R} \to \mathbb{R}$ be a function of α. A Massart noise (α, γ)-weak learner* `WkL` *for $\mathscr{C}$ is an algorithm such that, for any distribution D_x over $\mathscr{X}$, function $f \in \mathscr{C}$, and noise function $\eta(x)$ with noise bound $\eta < 1/2 - \alpha$,* `WkL` *outputs a hypothesis $h : \mathscr{X} \to \{\pm 1\}$ such that $\mathbf{Pr}_S[\mathrm{adv}^D(h) \geq \gamma] \geq 2/3$, where the sample S is drawn from Massart distribution $D =$* $\mathtt{Mas}\{f, D_x, \eta(x)\}$.

We let γ be a function of α because a given weak learning algorithm may satisfy stronger advantage guarantees if its input distributions are less noisy. For example, the advantage guarantee for our rectangle weak learner (Appendix 4.9) depends quadratically on α.

Our boosting algorithm operates in the *filtering* model of [BS07]. It generates intermediate distributions by drawing examples (x, y) from its oracle $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ and keeping them with probability $\mu(x, y)$, according to a function $\mu : \mathscr{X} \times \{\pm 1\} \to [0, 1]$. We refer to μ informally as a *measure* to emphasize that it induces a distribution D_μ (Definition 4.5.3), but need not be one itself. The expectation of the μ with respect to D is a useful quantity for analyzing distribution-independent boosting algorithms. It affects the sample complexity of making calls to the weak learner and is used to bound the error of the final hypothesis output by our algorithm.

Definition 4.2.6 (Density of a measure). *Let D be a Massart distribution, and let μ be a measure. The* density *of μ with respect to D is $d(\mu) = \mathbb{E}_{(x,y)\sim D}[\mu(x, y)]$.*

4.3 Boosting Algorithm

In this section, we describe our Massart noise-tolerant boosting algorithm `Massart-Boost` (Algorithm 12) and state our boosting theorem. More detailed pseudocode can be found in Appendix 4.6, along with a proof of convergence and sample complexity bounds. A proof of the tighter round and sample complexity bounds in Theorem 4.3.5 below can be found in Appendix 4.7.

Our algorithm maintains a working hypothesis $\text{sign}(G)$ for $G : \mathscr{X} \to \mathbb{R}$, initialized to 0. We use G to determine measure $\mu(x,y)$, which induces a distribution D_μ on which we query the weak learner. We update G with the resulting weak hypothesis, defining a new distribution over examples by decreasing μ on examples (x,y) for which $\text{sign}(G(x)) = y$ and increasing μ otherwise.

However, to preserve the Massart noise property, we must guarantee μ never assigns too little weight to an example $(x, f(x))$ relative to the weight of $(x, -f(x))$. To ensure we maintain a noise bound below $1/2 - \alpha$ for all intermediate distributions D_μ, we define $\mu(x,y) = 0$ for any example (x,y) at risk of violating this constraint. Since we have defined $\mu(x,y)$ to be anticorrelated with $yG(x)$, noisy example $(x, -f(x))$ may dominate clean example $(x, f(x))$ when $|G(x)|$ is large and $\text{sign}(G(x)) = f(x)$. To preserve the Massart noise property, we pick a threshold s for $|G(x)|$ and partition $\mathscr{X}$ into two sets, $\mathscr{X}^r$ and $\mathscr{X}^s$, based on whether $|G(x)| \geq s$.

The set $\mathscr{X}^r$ contains all $x \in \mathscr{X}$ for which $|G(x)| \geq s$. These x may have high effective noise rate, so they "risk" violating the Massart noise property. Thus, we assign $\mu(x,y) = 0$ for all $x \in \mathscr{X}^r$, regardless of y, removing them from the support of D_μ. The set $\mathscr{X}^s$ contains all $x \in \mathscr{X}$ for which $|G(x)| < s$, ensuring that the effective noise rate of each $x \in \mathscr{X}^s$ (in distribution D_μ) is bounded above by $1/2 - \alpha$. Thus, it is "safe" to call the weak learner on examples (x,y) where $x \in \mathscr{X}^s$. Initially, all $x \in \mathscr{X}$ are in $\mathscr{X}^s$. Our weak hypothesis h is guaranteed to have advantage against D_μ, so we can use h to improve our predictions $G(x)$ for $x \in \mathscr{X}^s$.

To improve our predictions on $x \in \mathscr{X}^r$, `Massart-Boost` performs an additional calibration step. If the working hypothesis $\text{sign}(G)$ misclassifies too many risky examples, it must be "overconfident" in its predictions on $\mathscr{X}^r$; so, we can improve G by adding hypothesis $-\text{sign}(G)$ (similar to the balancing step of [Fel10]). This recalibration step decreases $|G(x)|$ for $x \in \mathscr{X}^r$, moves all $x \in \mathscr{X}^r$ back into $\mathscr{X}^s$, and allows us to again call the weak learner on these examples. As more examples are correctly classified by $\text{sign}(G)$, the density of the measure μ decreases. When this density is small, the algorithm terminates and returns the classifier $\text{sign}(G)$.

4.3.1 Definitions

Let function $M : \mathbb{R} \to [0,1]$ satisfy $M(v) = 1$ when $v < 0$, and $M(v) = e^{-v}$ when $v \geq 0$. This "base" measure function is used to both define μ, the measure function used for reweighting intermediate distributions D_μ, and Φ, the potential function used to analyze convergence. We consider a measure function $\mu : \mathscr{X} \times \{\pm 1\} \to [0,1]$ that is parameterized by $s \in \mathbb{R}_{>0}$ and a real-valued function $F : \mathscr{X} \to \mathbb{R}$. In particular, μ assigns no weight to examples (x,y) such that $|F(x)| \geq s$.

Definition 4.3.1 (Measure function)**.** *Let* $\mu_{F,s}(x,y) \overset{\text{def}}{=} M(yF(x))$ *if* $|F(x)| < s$ *and* 0 *otherwise.*

At the start of the algorithm, we set $s = \log\left(\frac{1-\eta}{\eta+c}\right)$, where $c = \frac{4\eta\alpha}{1-2\alpha}$. This ensures that the noise rate of distribution D_μ (see Definition 4.5.3) is at most $1/2 - \alpha$. Our algorithm uses the measure function $\mu_{G_t,s}$ in round $t \in [T]$, where G_t is the working hypothesis G after t rounds of boosting. To simplify notation, let $\mu_t(x,y) \overset{\text{def}}{=} M(yG_t(x))$ if $|G_t(x)| < s$ and 0 otherwise.

In each round of boosting, we use G_t and s to partition the domain $\mathscr{X}$ into two sets: $\mathscr{X}_t^s$ and $\mathscr{X}_t^r$. If it is "safe" to run the weak learner on a sample containing x, we say $x \in \mathscr{X}_t^s$. Otherwise, $x \in \mathscr{X}_t^r$.

Definition 4.3.2 ($\mathscr{X}_t^s$, $\mathscr{X}_t^r$)**.** *For all* $x \in \mathscr{X}$, $x \in \mathscr{X}_t^s$ *if* $|G_t(x)| < s$ *and* $x \in \mathscr{X}_t^r$ *if* $|G_t(x)| \geq s$.

4.3.2 Pseudocode and Boosting Theorem

Algorithm 12. `Massart-Boost`$^{\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x)), \mathtt{WkL}}(\eta, \varepsilon, \gamma)$
η: Massart noise rate, ε: Target error in excess of η, γ: Weak learner advantage guarantee

```
$G \leftarrow 0$,
**while** $d(\mu) > \eta$ **do**
    $S \leftarrow$ sample from $D_\mu$
    $h \leftarrow \mathtt{WkL}(S)$
    $h^s(x) \leftarrow h(x)$ if $x \in \mathscr{X}_G^s$ and 0 otherwise
    $G \leftarrow G + \lambda h^s$
    **if** error of sign$(G)$ on $\mathscr{X}_G^r$ exceeds $\eta + \varepsilon$ **then**
        $h^r(x) \leftarrow -\mathrm{sign}(G(x))$ if $x \in \mathscr{X}_G^r$ and 0 otherwise
        $G \leftarrow G + \lambda h^r$
    update $\mu$ according to Definition 4.3.1
$H \leftarrow \mathrm{sign}(G_t)$
**return** $H$
```

Algorithm 12 is a simplified psuedocode description of our Massart noise-tolerant boosting algorithm. To prove that Algorithm 12 converges, we show that it make progress in each round of boosting against the following potential function (Lemma 4.3.3).

$$\Phi(t) = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{yG_t(x)}^{\infty} M(z)dz.$$

To see how this function allows us to capture the incremental progress made at each round, consider how the potential $\Phi(t)$ changes as we take a step of size λ in hypothesis space in the direction of some hypothesis h, starting from G_t. If we take λ sufficiently small, then we have from the mean value theorem that the change in potential should be not too much smaller than $\mathbb{E}_{(x,y)\sim D}\lambda y h_t(x) M(yG_t(x))$. Hypothetically, if the function $\mu_{G,s}(x,y)$ used for reweighting were *exactly* $M(yG(x))$, then a hypothesis h with advantage γ would guarantee a drop in potential of roughly

$$\mathop{\mathbb{E}}_{(x,y)\sim D} \lambda y h_t(x) \mu_t(x,y) = \mathop{\mathbb{E}}_{(x,y)\sim D} \lambda y h_t(x) D_{\mu_t}(x,y) d(\mu_t) = \lambda \gamma d(\mu_t).$$

In other words, the advantage guarantee of the weak learner ensures that the potential will drop by an amount proportional to the density of the current measure μ_t. Because $d(\mu_0) = 1$, and the algorithm terminates once $d(\mu_t)$ falls below η, this guaranteed potential drop would allow us to prove termination.

We cannot take $\mu_t(x,y) = M(yG_t(x))$ for all (x,y), however, because such a measure function might overweight noisy examples. Instead, our measure function is exactly $M(yG_t(x))$ only on examples in $\mathscr{X}_t^s$. The proof idea above then guarantees progress on such examples. If an example is not in $\mathscr{X}_t^s$, we permit the algorithm to make no progress or even regress. But, progress is made in expectation over all examples in $\mathscr{X}_t^r$, implying the following lemma (proved in Appendix 4.6).

Lemma 4.3.3 (Potential Drop)**.** *Take $\lambda = \gamma/8$, $\delta_{\texttt{WkL}} = \delta\eta\gamma^2/1536$, and assume $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha}$. Then for every round of boosting t, with all but probability $\delta\eta\gamma^2/768$,*

$$\Phi(t) - \Phi(t+1) \geq \frac{\gamma^2}{32}\left(d(\mu_t) - \frac{\eta}{2}\right)$$

The potential function Φ is bounded below by 0 and is initially equal to 1. The algorithm terminates if $d(\mu) < \eta$, so Lemma 4.3.3 implies that `Massart-Boost` terminates in $O(1/(\eta\gamma^2))$ rounds with high probability. A tighter analysis presented in Appendix 4.7 improves the round complexity to $O(\log^2(1/\eta)/\gamma^2)$ rounds.

Next, we show that a low density measure implies small misclassification error. Among

$x \in \mathscr{X}^s$, the misclassification error is at most $d(\mu)$, since $\mu(x,y) = 1$ for any example such that $\text{sign}(G(x)) \neq y$. To analyze the misclassification error on $x \in \mathscr{X}^r$, we proceed by casework on the rebalancing step being applied in the last round of boosting. We show that either i) $\mathscr{X}^r$ is at most an $\varepsilon/2$-fraction of distribution D, or ii) the misclassification error on $\mathscr{X}^r$ is at most $\eta + \varepsilon$. In the first case, we can assume $\text{sign}(G)$ misclassifies every $x \in \mathscr{X}^r$ and still achieves error $\eta + \varepsilon$. In the second case, the error on both $\mathscr{X}^s$ and $\mathscr{X}^r$ is at most $\eta + \varepsilon$.

Lemma 4.3.4 (Label error). *When* `Massart-Boost` *terminates, with all but probability $\delta/4$, trained classifier G satisfies* $\text{err}_{0\text{-}1}^{D}(\text{sign}(G)) \leq \eta + \varepsilon$, *assuming* $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha}$.

The following theorem combines these arguments with analyses of sample complexity, failure rate, and runtime to show that `Massart-Boost` efficiently converges to a low-error classifier.

Theorem 4.3.5 (Boosting Theorem). *Let* `WkL` *be an (α,γ)-weak learner requiring a sample of size $m_{\texttt{WkL}}$. Then for any $\delta \in (0,1/2]$, any Massart distribution D with noise rate $\eta < 1/2$, and any $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha}$, taking $\lambda = \gamma/8$ and $\kappa = \eta$,* $\texttt{Massart-Boost}^{\texttt{WkL}}(\lambda,\kappa,\eta,\varepsilon,\delta,\gamma,\alpha,m_{\texttt{WkL}})$ *will, with probability $1-\delta$, run for $T \in O\left(\log^2(1/\eta)/\gamma^2\right)$ rounds, output a hypothesis H such that* $\text{err}_{0\text{-}1}^{D}(H) \leq \eta + \varepsilon$ *and* $\text{err}_{0\text{-}1}^{D_x,f}(H) \leq \frac{\eta+\varepsilon}{1-\eta}$, *make no more than*

$$m \in O\left(\frac{\log^2(1/\eta)}{\gamma^2}\left(\frac{\log(1/(\delta\eta\gamma))}{\varepsilon^3} + \frac{\log(1/(\delta\eta\gamma))}{\eta^2} + \frac{m_{\texttt{WkL}}\log(1/(\delta\eta\gamma))}{\eta} + \frac{\log(1/(\delta\eta\gamma))}{\eta\gamma^2}\right)\right)$$

calls to $\text{EX}^{\text{Mas}}(f, D_x, \eta(x))$, *and run in time*

$$m \in O\left(\frac{\log^4(1/\eta)}{\gamma^4}\left(\frac{\log(1/(\delta\eta\gamma))}{\varepsilon^3} + \frac{\log(1/(\delta\eta\gamma))}{\eta^2} + \frac{m_{\texttt{WkL}}\log(1/(\delta\eta\gamma))}{\eta} + \frac{\log(1/(\delta\eta\gamma))}{\eta\gamma^2}\right)\right),$$

neglecting the runtime of the weak learner.

4.4 Lower Bound on Error for Massart Boosting

In this section, we show that no generic "black-box" boosting algorithm can achieve significantly better misclassification error than that of our algorithm, i.e., $\eta + \Theta(\eta\alpha)$, given a Massart noise oracle $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$. While this seemingly matches the lower bound for RCN boosting from [KS03], the RCN bound only implies a lower bound for the special case of Massart noise when $\eta = \mathrm{OPT}$. Moreover, that bound is not directly applicable in the Massart noise setting, since an RCN weak learner is not required to tolerate Massart noise.

We show that a lower bound of η still holds in the Massart noise setting even if OPT is much smaller than η. That is, boosting algorithms cannot be improved even when only a very small fraction of instances are actually noisy.

We consider the case where the target function $f \in \mathscr{C}$ is a pseudorandom function that is biased towards -1 labels. The noise function $\eta(x)$ is non-zero only on a small, random subset of preimages of -1 under f, and so there is a small fraction of examples $(x,1)$ where it cannot be distinguished if $f(x) = 1$ and $\eta(x) = 0$ or if $f(x) = -1$ and $\eta(x) > 0$. If the distributions D_μ queried by the boosting algorithm never satisfy $\mathbb{E}_{(x,y)\sim D_\mu}[y] > -2\gamma$, an uncooperative weak learner can return the constant function -1 and have advantage γ. On the other hand, if the boosting algorithm does query the weak learner on a distribution such that $\mathbb{E}_{(x,y)\sim D_\mu}[y] > -2\gamma$, it risks overweighting a noisy example $(x,1)$ and violating the Massart condition. Our adversarial, "rude" weak learner `rWkL` exploits this tension by only providing information attainable without knowing f.

4.4.1 Adversarial Weak Learner

Let `BlackBoxBoost` be a black-box boosting algorithm that draws m examples from EX and queries the weak learner T times. Let $\mathtt{SP}_t$ be a sampling procedure defining $D^{\mathtt{SP}_t}$, the t'th distribution queried to the weak learner. The following "rude" weak learner $\mathtt{rWkL}_{m,T}(S)$ attempts to only give information that `BlackBoxBoost` could discover alone: `rWkL` identifies the heavy-hitters of D^{SP_t} and defines $h(x)$ as x's majority label under D^{SP_t}. On non-heavy-hitters, $h(x)$ is the constant -1 function. For more details, see Appendix 4.8.3. For pseudocode, see Algorithm 19.

4.4.2 Error Lower Bound Theorem

Theorem 4.4.1 (Error Lower Bound Theorem)**.** *Let $\eta \in [0, 1/2), \alpha \in (0, 1/2 - \eta)$. Let $\{f_s\}$ be an η'-biased pseudorandom function family with security parameter n, where $\eta' = \eta(1 + \alpha/5)$. Let η, α be at least inversely polynomially in n bounded away from $1/2$. Then, for random s, no efficient black-box boosting algorithm* `BlackBoxBoost` *with example bound m running for T rounds, given query access to $(\alpha, \gamma(\alpha) \overset{\text{def}}{=} \alpha/20)$-weak learner* $\mathtt{rWkL}_{m,T}$ *and* $\mathrm{poly}(n, 1/(1 - 2\eta), 1/\gamma)$ *examples from example oracle* $\mathrm{EX}(U_n, f_s, \eta(x))$*, can output a hypothesis with label error at most $\eta(1 + o(\alpha))$. In particular, for all polynomials q, for all polynomial time black-box Massart boosting algorithms* `BlackBoxBoost` *with query access to* `rWkL` *and example oracle* EX*, for n sufficiently large, $\mathbf{Pr}_{s \in U_n}\left[\mathrm{err}_{0\text{-}1}^{U_n, f_s}(H) \leq \eta'\right] < \frac{1}{q(n)}$, where H is the trained classifier output by* `BlackBoxBoost`.

A detailed proof is presented in Appendix 4.8. Intuitively, since `rWkL` can be simulated without knowing f, `rWkL` cannot help the boosting algorithm learn f. So, no efficient algorithm can use `rWkL` to learn pseudorandom f. The primary focus of the proof is showing that `rWkL` is a valid Massart noise weak learner.

Adversarial weak learner `rWkL` has a stability property we call *replicability*. Assuming D^{SP_t} is Massart, `rWkL` returns the same exact hypothesis h^v with high probability over its sample $S \sim D^{\mathsf{SP}_t}$. Using replicability, we argue that i) boosting with `rWkL` can be efficiently simulated without knowing the function f and ii) `rWkL` satisfies the definition of a Massart noise weak learner. By Massart-ness, the labels of heavy hitters $x \in \mathscr{X}$ of D^{SP_t} must be biased towards the true label $f(x)$, guaranteeing advantage on heavy-hitters. To show `rWkL` also handles non-heavy-hitters gracefully, we first show that boosting with `rWkL` can be efficiently simulated, and then we appeal to the pseudorandomness of f. `rWkL` only fails to return a hypothesis with γ advantage if D^{SP_t} is supported on many non-heavy-hitters x whose true label $f(x) = 1$. We can conclude by observing that finding many such non-heavy-hitters implies a violation of the pseudorandomness assumption.

Lemma 4.4.2 (Advantage of `rWkL`)**.** *Let $D^{\mathtt{SP}_t}$ denote the distribution induced by the sampling procedure $\mathtt{SP}_t$ and* `hEG` *at round $t \in [T]$ of boosting. Similarly, let $D_r^{\mathtt{SP}_t}$ denote the distribution induced by $\mathtt{SP}_t$ and* `rEG`*. Let S_t denote a sample drawn i.i.d. from $D_r^{\mathtt{SP}_t}$. Then for all* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$ *rounds of boosting* `rWkL` *with* `rEG`*, if $D^{\mathtt{SP}_t}$ is Massart, then with probability $1 - O(1/(mT))$ over its internal randomness, $\mathtt{rWkL}(S_t)$ outputs a hypothesis h_t with advantage at least γ against $D^{\mathtt{SP}_t}$, except with negligible probability in m over the choice of $\mathtt{SP}_t$.*

4.5 Appendix: Additional Definitions and Proofs for Section 4.2

4.5.1 Boosting and Weak Learners

Recall the definition of a Massart noise weak learner.

Definition 4.2.5 (Massart Noise Weak Learner)**.** *Let $\mathscr{C}$ be a concept class of functions $f : \mathscr{X} \to \{\pm 1\}$. Let $\alpha \in [0,1/2)$. Let $\gamma : \mathbb{R} \to \mathbb{R}$ be a function of α. A Massart noise (α,γ)-weak learner* `WkL` *for $\mathscr{C}$ is an algorithm such that, for any distribution D_x over $\mathscr{X}$, function $f \in \mathscr{C}$, and noise function $\eta(x)$ with noise bound $\eta < 1/2 - \alpha$,* `WkL` *outputs a hypothesis $h : \mathscr{X} \to \{\pm 1\}$ such that $\mathbf{Pr}_S[\mathrm{adv}^D(h) \geq \gamma] \geq 2/3$, where the sample S is drawn from Massart distribution $D = \mathtt{Mas}\{f, D_x, \eta(x)\}$.*

Note that we define an (α,γ)-Massart noise weak learner to have failure probability at most $1/3$. For any desired $\delta \in (0,1/3)$, such a weak learner can be used to obtain a hypothesis with advantage $\gamma/2$, with all but probability δ, by standard repetition techniques demonstrated in Lemma 4.5.1.

Lemma 4.5.1 (`WkL` repetition)**.** *Let* `WkL` *be an (α,γ)-Massart noise weak learner requiring a sample of size $m_{\mathtt{WkL}}$. Then for any $\delta \in (0,1/3)$, $2\log(2/\delta)$ calls to* `WkL` *and $2\log(2/\delta)(m_{\mathtt{WkL}} + 1/\gamma^2)$ examples suffice to obtain a hypothesis with advantage at least $\gamma/2$ with all but probability δ.*

Proof. To drive down the failure probability of `WkL`, we draw $2\log(2/\delta)$ samples of size $m_{\mathtt{WkL}}$ and run `WkL` on each of them to obtain a list of hypotheses, at least one of which has advantage γ with all but probability $\delta/2$. We then draw a sample of size $2\log(2/\delta)/\gamma^2$ to test each hypothesis in our list, keeping the best. The Chernoff-Hoeffding inequality guarantees that testing our hypotheses overestimates the advantage by more than $\gamma/2$ with probability no greater than $\delta/2$, and so we obtain a hypothesis with advantage at least $\gamma/2$ with all but probability δ. $\square$

We are primarily interested in *efficient* Massart noise weak learners (Definition 4.5.2).

Definition 4.5.2 (Efficient Massart Noise Weak Learner)**.** *Let* $\texttt{WkL}$ *be an* (α,γ)*-Massart noise weak learner. Let n be the maximum bit complexity of a single example* $(x,y) \in \mathscr{X} \times \{\pm 1\}$*, and let* $m_{\texttt{WkL}}$ *denote the number of examples comprising sample S.* $\texttt{WkL}(S)$ *is* efficient *if*

1. $\texttt{WkL}$ *uses* $m_{\texttt{WkL}}(n,\eta,\gamma) = \mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$ *examples.*
2. $\texttt{WkL}$ *outputs a hypothesis h in time* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$.
3. *Hypothesis* $h(x)$ *has bit complexity* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$.
4. *For all* $x \in \mathscr{X}$*, the hypothesis* $h(x)$ *can be evaluated in time* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$.

Boosting algorithms can utilize the advantage guarantee of the weak learner by cleverly reweighting its input distributions. To sample from these reweighted distributions, we sample from the underlying distribution D via $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ and reject examples according to a function $\mu : \mathscr{X} \times \{\pm 1\} \to [0,1]$. We refer to μ informally as a *measure* to emphasize that it induces a distribution, but need not be one itself.

Definition 4.5.3 (Rejection Sampled Distribution D_μ)**.** *Let D be a Massart distribution, and let* $\mu : \mathscr{X} \times \{\pm 1\} \to [0,1]$ *be an efficiently computable measure. We define* D_μ *as the distribution generated from D by the following rejection sampling procedure: draw an example* $(x,y) \sim D$*. With probability* $\mu(x,y)$*, keep this example. Otherwise, repeat this process (until an example is kept).*

Note that some choices of μ may induce a distribution D_μ which is *not* Massart, as reweighting examples may distort $\eta(x)$, and so it is possible that we no longer have a noise bound less than $1/2$. In particular, if there is an $x \in \mathrm{supp}(D_x)$ for which $\mu(x, -f(x))\eta(x) \gg \mu(x, f(x))(1-\eta(x))$, then D_μ is not a Massart distribution and running the weak learner on a sample from this distribution is not guaranteed to return a hypothesis with good advantage. In designing our boosting algorithm, we will choose μ carefully to ensure that this never happens.

Lemma 4.5.4 (Sampling from D_μ). *For any $m > 0$, $\delta \in (0, 1/2)$, obtaining a sample of size m from D_μ by rejection sampling from D requires no more than $\frac{\log(1/\delta)}{d(\mu)^2} + \frac{2m}{d(\mu)}$ examples from distribution D, with all but probability δ.*

Proof. From the definition of D_μ, we can sample from D_μ by drawing an example (x, y) from D and keeping it with probability $\mu(x, y)$. By Definition 4.2.6, we expect to keep an example with probability $d(\mu)$. Then the Chernoff-Hoeffding inequality allows us to conclude that, following this procedure, if we draw $\frac{\log(1/\delta)}{d(\mu)^2} + \frac{2m}{d(\mu)}$ examples from D, we keep at least m of them with all but probability δ. □

4.6 Appendix: Additional Lemmas and Proofs for Section 4.3

4.6.1 Boosting Algorithm

In this appendix, we present our Massart noise-tolerant boosting algorithm `Massart-Boost` (Algorithm 13) and prove the following theorem:

Theorem 4.6.1 ((Simplified) Boosting Theorem). *Let `WkL` be an (α, γ)-weak learner requiring a sample of size $m_{\texttt{WkL}}$. Then for any Massart distribution D with noise rate $\eta < 1/2$, and any $\varepsilon > \frac{8\eta\alpha}{1-2\alpha}$, `Massart-Boost` will*

- *make at most $T \in \tilde{O}\left(1/(\eta\gamma^2)\right)$ calls to `WkL`*
- *output a hypothesis H such that $\mathrm{err}_{0\text{-}1}^{D}(H) \leq \eta + \varepsilon$ and $\mathrm{err}_{0\text{-}1}^{D_x, f}(H) \leq \frac{\eta+\varepsilon}{1-\eta}$*
- *make*
$$m \in \tilde{O}\left(\frac{1}{\eta\gamma^2\varepsilon^3} + \frac{m_{\texttt{WkL}}}{\eta^2\gamma^2} + \frac{1}{\eta^2\gamma^4}\right)$$
calls to its example oracle $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$
- *run in time*
$$\tilde{O}\left(\frac{m_{\texttt{WkL}}}{\eta^3\gamma^4} + \frac{1}{\eta^3\gamma^6} + \frac{1}{\eta^2\gamma^4\varepsilon^3}\right),$$

neglecting the runtime of the weak learner.

4.6.2 Description of Boosting Algorithm

The pseudocode for our boosting algorithm is Algorithm 13. It makes calls to three subroutines in addition to the weak learner: `Samp` (Routine 14), `Est-Density` (Routine 15), and `OverConfident` (Routine 16), which we first describe informally.

The `Samp` subroutine captures the procedure by which our algorithm draws samples for the weak learner. These samples are drawn i.i.d. from the reweighted distributions constructed by our booster. The `Samp` procedure is given oracle access to $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, so that is can sample from D. The `Samp` procedure takes as input a function (the current hypothesis) G, the size of the sample $m_{\mathtt{WkL}}$ required by the weak learner, and the threshold s for $|G(x)|$ that defines which examples are to be withheld from the weak learner. `Samp` repeatedly draws examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, and keeps them with probability $\mu_{G,s}(x,y)$, the value of which is computed using G and s. After `Samp` has drawn a sample of size $m_{\mathtt{WkL}}$, it returns the sample, and this is what is given to the weak learner as input.

The subroutine `Est-Density` is used to estimate the current density of the measure μ_t, which is necessary to test the termination condition of our algorithm. Ideally, the algorithm terminates once $d(\mu_t) < \kappa$, and so `Est-Density` is called at the end of each round of boosting to estimate this density. `Est-Density` is given oracle access to $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ and takes as input G and s, so that it can empirically estimate $d(\mu)$ using a sample drawn from D. `Est-Density` also takes as input three parameters: $\delta_{\mathtt{dens}}$, ε, and η. The parameters ε and η are used to specify the desired accuracy of the density estimation, $\beta = \min\{\varepsilon/2, \eta/4\}$. The parameter $\delta_{\mathtt{dens}}$ specifies the tolerable probability of failure of the density estimation procedure (i.e., the probability that `Est-Density` returns an estimate of $d(\mu)$ with error greater than β).

The subroutine `OverConfident` determines when the error of $\mathrm{sign}(G)$ on examples withheld from the weak learner (i.e., examples in $\mathscr{X}_G^r$) has grown too large. If, at round t, the probability mass on $\mathscr{X}_t^r$ is large, and the error of $\text{sign}(G_t)$ on $\mathscr{X}_t^r$ exceeds $\eta + \varepsilon$, we must improve G_t on these examples to reach our target error of $\eta + \varepsilon$. Because we withhold examples in $\mathscr{X}_t^r$ from the weak learner at round $t+1$, we are not guaranteed that the next weak hypothesis, h_{t+1}, will provide any amount of progress in expectation on these examples, so some additional steps are needed to improve G_t. The role of `OverConfident` is to estimate whether the probability mass of $\mathscr{X}_t^r$ is significant, and if so, whether the error of $\text{sign}(G_t)$ on $\mathscr{X}_t^r$ is large enough that an additional correction step is necessary.

The subroutine `OverConfident` is given oracle access to $\text{EX}^{\text{Mas}}(f, D_x, \eta(x))$ and takes as input G, s, η, and ε. It also takes an additional parameter $\delta_{\texttt{err}}$, which specifies the tolerable probability of failure for `OverConfident` (i.e., the probability that `OverConfident` returns a false positive or false negative). `OverConfident` first estimates the probability that $|G(x)| > s$. If it estimates $\mathbf{Pr}_{(x,y)\sim D}[|G(x)| \geq s] < \varepsilon/4$, then the overall contribution of examples in $\mathscr{X}_G^r$ to the total error of $\text{sign}(G)$ is sufficiently small that the correction step is not needed. In this case, `OverConfident` returns false. If it estimates the probability to be greater than $\varepsilon/4$, it draws a new sample for estimating the conditional error of G on $\mathscr{X}_G^r$. The subroutine makes calls to $\text{EX}^{\text{Mas}}(f, D_x, \eta(x))$ and keeps only the examples (x, y) such that $x \in \mathscr{X}_G^r$. It draws a sufficiently large sample to estimate the error of G on this set to within $\varepsilon/4$ with all but probability $\delta_{\texttt{err}}/2$. If the estimated error exceeds $\eta + 3\varepsilon/4$, it returns true and the correction step takes place. Otherwise it returns false, as the conditional error on these points is tolerable.

Algorithm 13. $\texttt{Massart-Boost}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x)),\texttt{WkL}}(\lambda,\kappa,\eta,\varepsilon,\delta,\gamma,\alpha,m_{\texttt{WkL}})$
λ: Learning rate
κ: Target density for measure μ
η: Massart noise rate
ε: Target error in excess of η
δ: Target failure probability for `Massart-Boost`
α: Weak learner parameter indicating `WkL` can tolerate noise γ: Weak learner advantage guarantee
$\eta < 1/2-\alpha$
$m_{\texttt{WkL}}$: Sample size for `WkL`

```
$c \leftarrow \frac{4\eta\alpha}{1-2\alpha}$, $s \leftarrow \log\left(\frac{1-\eta}{\eta+c}\right)$          // set parameters for managing noise
$\delta_{\texttt{err}} \leftarrow \delta\eta\gamma^2/1536$, $\delta_{\texttt{dens}} \leftarrow \delta\eta\gamma^2/1024$          // set failure probabilities for subroutines
$G_0 \leftarrow 0, t \leftarrow 0$          // initialize $G$, round counter
$\widehat{d} \leftarrow 1$          // initialize density estimate
while $\widehat{d} > \kappa$ do
    $t \leftarrow t+1$
    $S \leftarrow \texttt{Samp}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G_{t-1},n,s)$          // draw a sample for the weak learner
    $h_t \leftarrow \texttt{WkL}(S)$          // obtain a weak hypothesis
    $h_t^s(x) \leftarrow \begin{cases} h_t(x) & \text{if } x \in \mathscr{X}_{t-1}^s, \\ 0 & \text{otherwise} \end{cases}$          // zero out hypothesis on $\mathscr{X}_t^r$
    $G_t \leftarrow G_{t-1} + \lambda h_t^s$          // update working hypothesis
    if $\texttt{OverConfident}^{\mathrm{EX}^{\mathrm{Mas}}}(G_t, s, \delta_{\texttt{err}}, \varepsilon)$ then
        $h_t^r(x) \leftarrow \begin{cases} -\mathrm{sign}(G_t(x)) & \text{if } x \in \mathscr{X}_t^r, \\ 0 & \text{otherwise} \end{cases}$          // if error on $\mathscr{X}_t^r$ is high, be less confident
        $G_t \leftarrow G_t + \lambda h_t^r$          // update working hypothesis
    $\widehat{d} \leftarrow \texttt{Est-Density}^{\mathrm{EX}^{\mathrm{Mas}}}(G_t, s, \delta_{\texttt{dens}}, \varepsilon)$          // estimate density of measure
$H \leftarrow \mathrm{sign}(G_t)$
return $H$
```

Routine 14. $\texttt{Samp}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G, m_{\texttt{WkL}}, s)$, rejection sampling for `Massart-Boost`

```
$S \leftarrow \emptyset$
while $|S| \leq m_{\texttt{WkL}}$ do
    $(x,y) \leftarrow \mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$
    With prob $\mu_{G,s}(x,y)$, $S \leftarrow S \,||\, (x,y)$            // draw a sample for WkL from $D_\mu$
return $S$
```

Routine 15. $\texttt{Est-Density}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G, s, \delta_{\texttt{dens}}, \varepsilon)$, estimate density of μ.
Used to decide when to terminate main loop of `Massart-Boost`

```
$\beta \leftarrow \min\{\varepsilon/2, \eta/4\}$
Draw set $S$ of $\log(1/\delta_{\texttt{dens}})/(2\beta^2)$ examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$
$\widehat{d} \leftarrow \frac{1}{|S|}\sum_{(x,y)\in S} \mu_{G,s}(x,y)$            // estimate the density of $\mu$
return $\widehat{d}$
```

Routine 16. $\texttt{OverConfident}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G, s, \delta_{\texttt{err}}, \varepsilon)$, adjust the value of G on $\mathscr{X}_G^r$

```
Draw set $S$ of $32\log(2/\delta_{\texttt{err}})/\varepsilon^2$ examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$
if $\frac{|S\cap\mathscr{X}_G^r|}{|S|} \leq \varepsilon/4$ then
    return false            // if $\mathscr{X}_G^r$ is small, return false
$S \leftarrow \emptyset$
while $|S| \leq 8\log(2/\delta_{\texttt{err}})/\varepsilon^2$ do
    $(x,y) \leftarrow \mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$
    if $|G(x)| \geq s$ then
        $S \leftarrow S \,||\, (x,y)$
$\widehat{\varepsilon} \leftarrow \frac{1}{2|S|}\sum_{(x,y)\in S} |y - \mathrm{sign}(G(x))|$            // if $\mathscr{X}_G^r$ is large, estimate error on $\mathscr{X}_G^r$
if $\widehat{\varepsilon} \geq \eta + 3\varepsilon/4$ then
    return true            // if error and $\mathscr{X}_G^r$ are large, return true
else
    return false            // if error is small, return false
```

4.6.3 Convergence of `Massart-Boost`

In this subsection, we bound the error of the final hypothesis output by Algorithm 13 and the number of rounds of boosting required to achieve this error bound. We begin by showing an invariant of our algorithm that will be useful in subsequent potential arguments.

Lemma 4.6.2 (Invariant for $|G_t(x)|$)**.** *For all rounds t of boosting and all examples $(x,y) \in (\mathscr{X},\mathscr{Y})$, at the end of round t, $|G_t(x)| < s+\lambda$.*

Proof. We first show that at the end of round t, $|G_t(x)| \leq s+\lambda$. On examples x such that $|G_{t-1}(x)| \geq s$, either $G_t(x) = G_{t-1}(x) - \lambda \operatorname{sign}(G_{t-1}(x))$ or $G_t(x) = G_{t-1}(x)$, and so $|G_t(x)| \leq |G_{t-1}(x)|$. Since $|G_t(x)| \geq |G_{t-1}(x)|$ only when $|G_{t-1}(x)| < s$, we now consider how much larger it can be. For examples such that $|G_{t-1}(x)| < s$, either $G_t(x) = G_{t-1}(x) + \lambda h_t(x) + \lambda h_t^r(x)$ (when `OverConfident` returns true) or $G_t(x) = G_{t-1}(x) + \lambda h_t(x)$. In the first case, if $\lambda h_t^r(x) \neq 0$, then $\operatorname{sign}(G_{t-1}(x) + h_t(x)) = -h_t^r(x)$, and so $|G_t(x)| \leq |G_{t-1}(x) + \lambda h_t(x)|$ for both cases. Since the hypothesis $h_t(x)$ output by the weak learner has codomain $[-1,1]$, it follows that $|G_t(x)| \leq |G_{t-1}(x) + \lambda h_t(x)| < s+\lambda$. □

We now recall the potential function introduced in Section 4.3.2. We denote by $\phi_t(x,y)$ the function

$$\phi_t(x,y) = \int_{yG_t(x)}^{\infty} M(z)dz.$$

Then our potential function is defined as

$$\Phi(t) = \mathop{\mathbb{E}}_{(x,y)\sim D}[\phi_t(x,y)] = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{yG_t(x)}^{\infty} M(z)dz.$$

We will make use of the following upper-bound on $d(\mu_t)$ in terms of $\Phi(t)$.

Lemma 4.6.3 (Potential upper-bounds density)**.** *For every round t of* `Massart-Boost`*, $d(\mu_t) \leq \Phi(t)$.*

Proof. We show that $d(\mu_t) \leq \Phi_t$ by showing $\mu_t(x,y) \leq \phi_t(x,y)$. For examples (x,y) such that $yG_t(x) > 0$, we have

$$\phi_t(x,y) = \int_{yG_t(x)}^{\infty} e^{-z} dz = e^{-yG_t(x)} \geq \mu_t(x,y).$$

For the remaining points, we simply observe that either $\mu_t(x,y) = 1$ or $\mu_t(x,y) = 0$. In either case, the potential

$$\phi_t(x,y) = \int_{yG_t(x)}^{\infty} M(z) dz \geq \int_0^{\infty} e^{-z} dz = 1$$

and so we have that $\mu_t(x,y) \leq \phi_t(x,y)$, and therefore $d(\mu_t) \leq \Phi_t$. ☐

We now prove that `Massart-Boost` makes progress against Φ at each round.

Lemma 4.3.3 (Potential Drop). *Take $\lambda = \gamma/8$, $\delta_{\texttt{WkL}} = \delta\eta\gamma^2/1536$, and assume $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha}$. Then for every round of boosting t, with all but probability $\delta\eta\gamma^2/768$,*

$$\Phi(t) - \Phi(t+1) \geq \frac{\gamma^2}{32}\left(d(\mu_t) - \frac{\eta}{2}\right)$$

Proof. We first show that for all $(x,y) \sim D$ such that $x \in \mathscr{X}_t^s$,

$$\phi_t(x,y) - \phi_{t+1}(x,y) \geq \lambda \mu_t(x,y)(yh_t(x) - 2\lambda).$$

We prove this statement for examples such that $0 \leq yG_t(x), yG_{t+1}(x) < s$, i.e., in the non-constant region of M, and observe that this suffices to prove the statement for all (x,y) such that $x \in \mathscr{X}_t^s$. To see that this is true, note that within the constant region of M, $\phi_t(x,y) - \phi_{t+1}(x,y) = \lambda \mu_t(x,y) y h_t(x)$.

For examples moved by h_t from constant to non-constant regions of M,

$$\begin{aligned}
\phi_t(x,y) - \phi_{t+1}(x,y) &= \int_{yG_t(x)}^{0} 1dz + \int_{0}^{yG_{t+1}(x)} e^{-z}dz \\
&\geq \int_{0}^{\lambda y h_t(x)} e^{-z}dz \\
&\geq \lambda M(0)(yh_t(x) - 2\lambda) && \text{(by assumption)} \\
&= \lambda \mu_t(x,y)(yh_t(x) - 2\lambda)
\end{aligned}$$

Similarly, for examples moving into the constant region from non-constant,

$$\begin{aligned}
\phi_t(x,y) - \phi_{t+1}(x,y) &= -\int_{yG_{t+1}(x)}^{0} 1dz - \int_{0}^{yG_t(x)} e^{-z}dz \\
&= \int_{yG_t(x)}^{0} e^{-z}dz + \int_{0}^{yG_{t+1}(x)} 1dz \\
&\geq \int_{yG_t(x)}^{yG_{t+1}} e^{-z}dz \\
&\geq \lambda \mu_t(x,y)(yh_t(x) - 2\lambda) && \text{(by assumption/proved below)}
\end{aligned}$$

and so it only remains to prove the claim for examples such that $0 \leq yG_t(x), yG_{t+1}(x) < s$. By the definition of ϕ_t, we have

$$\begin{aligned}
\phi_t(x,y) - \phi_{t+1}(x,y) &= \int_{yG_t(x)}^{yG_{t+1}(x)} M(z)dz \\
&= \int_{yG_t(x)}^{yG_{t+1}(x)} e^{-z}dz \\
&= e^{-v}(yG_{t+1}(x) - yG_t(x)) && \text{(for some } v \in [yG_t(x), yG_{t+1}(x)]) \\
&\geq e^{-yG_{t+1}(x)} \lambda y h_t(x) \\
&= e^{-yG_t(x)} e^{-\lambda y h_t(x)} \lambda y h_t(x) \\
&\geq \mu_t(x,y)\lambda y h_t(x) - 2\mu_t(x,y)\lambda^2 && ((xe^{-x} \geq x - 2x^2 \text{ for } x \in [-1,1]) \\
&= \lambda \mu_t(x,y)(y h_t(x) - 2\lambda)
\end{aligned}$$

and so the contribution to the potential drop from $(x,y) \in \mathscr{X}_t^s$ is as claimed.

We now consider the contribution to the potential drop from examples (x,y) where $x \in \mathscr{X}_t^r$, by analyzing two complementary cases.

1. $\texttt{OverConfident}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G_t, s, \delta_{\texttt{err}}, \varepsilon)$ returns false

2. $\texttt{OverConfident}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G_t, s, \delta_{\texttt{err}}, \varepsilon)$ returns true

In the first case, $h_t^r = 0$, and so $yG_{t+1}(x) = yG_t(x)$ holds for all these examples. Therefore the contribution to the potential drop is

$$\mathop{\mathbb{E}}_{(x,y)\sim D}\left[\phi_t(x,y) - \phi_{t+1}(x,y) \middle| x \in \mathscr{X}_t^r\right] = 0.$$

In the second case, $h_t^r(x) = -\mathrm{sign}(G_t(x))$, and so $yG_{t+1}(x) = yG_t(x) - \mathrm{sign}(G_t(x))$ for these examples. $\texttt{OverConfident}^{\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))}(G_t, s, \delta_{\texttt{err}}, \varepsilon)$ only returns true if it has estimated the error on examples such that $x \in \mathscr{X}_t^r$ exceeds $\eta + 3\varepsilon/4$. This routine estimates the error from a sample of size $8\log(2/\delta_{\texttt{err}})/\varepsilon^2$, and so it holds by the Chernoff-Hoeffding inequality that with all but probability $\delta_{\texttt{err}}/2$, that

$$\mathbf{Pr}_{(x,y)\sim D}\left[yG_t(x) \leq -s \middle| x \in \mathscr{X}_t^r\right] \geq \eta + \varepsilon/2.$$

This implies a contribution to the potential drop of

$$\begin{aligned}
\mathop{\mathbb{E}}_{(x,y)\sim D}\left[\phi_t(x,y) - \phi_{t+1}(x,y) \middle| x \in \mathscr{X}_t^r\right] &= \mathbf{Pr}_{(x,y)\sim D}[yG_t(x) \leq -s | x \in \mathscr{X}_t^r] \int_{yG_t(x)}^{yG_t(x)+\lambda} 1 dz \\
&\quad + \mathbf{Pr}_{(x,y)\sim D}[yG_t(x) \geq s | x \in \mathscr{X}_t^r] \int_{yG_t(x)}^{yG_t(x)-\lambda} e^{-z} dz \\
&\geq (\eta + \varepsilon/2)\lambda + (1 - \eta - \varepsilon/2) \int_{yG_t(x)}^{yG_t(x)-\lambda} e^{-z} dz \\
&\geq (\eta + \varepsilon/2)\lambda + (1 - \eta - \varepsilon/2)e^{-s}(1 - e^{-\lambda}) \\
&\qquad\qquad (\text{from } yG_t(x) \leq s + \lambda) \\
&\geq (\eta + \varepsilon/2)\lambda - (1 - \eta - \varepsilon/2)e^{-s}(\lambda - \lambda^2) \\
&\qquad\qquad (\text{from } e^{-\lambda} \leq 1 - \lambda + \lambda^2) \\
&= (\eta + \varepsilon/2)\lambda - (1 - \eta - \varepsilon/2)(\frac{\eta + c}{1 - \eta})(\lambda - \lambda^2) \\
&\qquad\qquad (\text{by definition of } s) \\
&\geq \frac{\varepsilon\lambda}{2}(1 + \eta - \eta\lambda) - c\lambda(1 - \lambda) - \eta\lambda^2, \quad (\text{from } \eta < \tfrac{\eta+c}{1-\eta})
\end{aligned}$$

and so as long as $c \leq \varepsilon/2 \leq \frac{\varepsilon(1+\eta-\eta\lambda)}{2(1-\lambda)}$, we have

$$\mathop{\mathbb{E}}_{(x,y)\sim D}[\phi_t(x,y) - \phi_{t+1}(x,y) | x \in \mathscr{X}_t^r] \geq -\eta\lambda^2.$$

Recall that we have assumed $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha} = 2c$ and so the stated bound holds.

We now lower-bound the drop in the potential function. With probability at least $1 - \delta_{\texttt{err}}$, `Est-Density` does not overestimate the error of the current hypothesis on points x for which $x \in \mathscr{X}_t^r$ by more than $\varepsilon/4$, and so we have

$$
\begin{aligned}
\Phi(t) - \Phi(t+1) &= \mathop{\mathbb{E}}_{(x,y)\sim D}[\phi_t(x,y) - \phi_{t+1}(x,y)] \\
&\geq \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^s] \mathop{\mathbb{E}}_{(x,y)\sim D}[\lambda \mu_t(x,y)(yh_t(x) - 2\lambda)\big| x \in \mathscr{X}_t^s] \\
&\qquad - \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r]\eta\lambda^2 \\
&= \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^s] \mathop{\mathbb{E}}_{(x,y)\sim D}[\lambda \mu_t(x,y)(yh_t(x) - 2\lambda)\big| x \in \mathscr{X}_t^s] \\
&\qquad - \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r]\eta\lambda^2 \\
&\qquad + \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \mathop{\mathbb{E}}_{(x,y)\sim D}[\lambda \mu_t(x,y)(yh_t(x) - 2\lambda)\big| x \in \mathscr{X}_t^r] \\
&\qquad\qquad\qquad\qquad\qquad \text{(since } \mu_t(x,y) = 0 \text{ for } x \in \mathscr{X}_t^r\text{)} \\
&\geq \mathop{\mathbb{E}}_{(x,y)\sim D}[\lambda \mu_t(x,y)(yh_t(x) - 2\lambda)] - \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r](\eta\lambda^2) \\
&\geq \mathop{\mathbb{E}}_{(x,y)\sim D}[\lambda \mu_t(x,y)(yh_t(x) - 2\lambda)] - \eta\lambda^2.
\end{aligned}
$$

From our weak learner guarantee and Lemma 4.5.1, we know that with all but probability $\delta_{\texttt{WkL}}$, h_t has advantage $\gamma/2$ against D_{μ_t}. Therefore with all but probability $\delta_{\texttt{WkL}} + \delta_{\texttt{err}}$,

$$\Phi(t) - \Phi(t+1) \geq \frac{\lambda\gamma}{2} d(\mu_t) - 2\lambda^2 d(\mu_t) - \eta\lambda^2.$$

Then taking $\delta_{\texttt{WkL}} = \delta_{\texttt{err}} = \frac{\delta\eta\gamma^2}{1536}$ and $\lambda = \gamma/8$, we have

$$\Phi(t) - \Phi(t+1) \geq \frac{\gamma^2}{32}\left(d(\mu_t) - \frac{\eta}{2}\right)$$

with all but probability $\delta\eta\gamma^2/768$. $\square$

Now we use our guaranteed drop in potential to show bounds on termination, as well as the density of the measure μ_t at the end of the final round t.

Lemma 4.6.4 (Termination)**.** *Let* `WkL` *be an* (α,γ)*-weak learner requiring a sample of size* $m_{\texttt{WkL}}$ *and let* $\delta_{\texttt{WkL}} = \delta\eta\gamma^2/1536$*. Let* $\lambda = \gamma/8$ *and* $\kappa \geq \eta$*. Then with all but probability* $\delta/3$*,* $\texttt{Massart-Boost}^{\texttt{WkL}}$ *terminates within* $T \leq 128/(\eta\gamma^2)$ *rounds, and conditioned on termination,* $d(\mu_T) \leq \kappa + \varepsilon/2$ *with all but probability* $\delta/8$*.*

Proof. `Massart-Boost` terminates once `Est-Density` estimates $\widehat{d}(\mu_t) \leq \kappa$. Since `Est-Density` draws a sample of size $2\log(1/\delta_{\texttt{dens}})/\beta^2$ for $\beta = \min\{\varepsilon/2, \eta/4\}$, the Chernoff-Hoeffding inequality bounds the probability that `Est-Density` overestimates $d(\mu_t)$ by more than β by $\delta_{\texttt{dens}}$. Therefore the probability that `Massart-Boost` fails to terminate at the end of any round for which $d(\mu_t) \leq \kappa - \beta$ is no more than $\delta_{\texttt{dens}}$. We condition on this failure not occuring for the rest of the proof.

From Lemma 4.3.3, we have that with probability at least $1 - \delta_{\texttt{WkL}} - \delta_{\texttt{err}}$,

$$\Phi(t) - \Phi(t+1) \geq \frac{\gamma^2}{32}\left(d(\mu_t) - \frac{\eta}{2}\right).$$

We have taken $\kappa \geq \eta$, and $\beta \leq \eta/4$, so except with probability $\delta_{\texttt{WkL}} + \delta_{\texttt{err}}$, the potential drops by at least $\frac{\gamma^2}{32}(\kappa - \beta - \frac{\eta}{2}) > \frac{\eta\gamma^2}{128}$ in each round. The potential function begins at

$$\Phi_0 = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_0^\infty M(z)dz = 1$$

and has minimum value 0, so taking $T = \frac{128}{\eta\gamma^2}$, it must be the case that $d(\mu) \leq \kappa - \beta$ by round T with probability at least $1 - T(\delta_{\texttt{WkL}} + \delta_{\texttt{err}})$. So with all but probability $128(\delta_{\texttt{WkL}} + \delta_{\texttt{err}})/(\eta\gamma^2)$, $d(\mu) \leq \kappa - \beta$ after T rounds, and so `Massart-Boost` must have terminated by then except with probability $\delta_{\texttt{dens}}$. This gives a total failure probability of

$$\delta_{\texttt{dens}} + \frac{128(\delta_{\texttt{WkL}} + \delta_{\texttt{err}})}{(\eta\gamma^2)} = \delta_{\texttt{dens}} + \delta/6 \leq \delta/3$$

It remains to bound the probability that `Massart-Boost` terminates at round t with $d(\mu_t) >$

$\kappa + \varepsilon/2$. Again, using the Chernoff-Hoeffding inequality and the sample size of `Est-Density`, if $d(\mu_t) > \kappa + \varepsilon/2$, `Massart-Boost` terminates with probability no more than $\delta_{\texttt{dens}}$. Union bounding over all rounds gives a failure probability $128\delta_{\texttt{dens}}/(\eta\gamma^2) = \delta/8$. □

4.6.4 Error Bounds

In this subsection we prove upper-bounds for the error of the final hypothesis $H = \text{sign}(G)$, both with respect to the distribution D and to the target function f on the marginal distribution D_x.

Lemma 4.6.5 (Label error)**.** *When the algorithm terminates at round t, with all but probability $\delta/4$ over the randomness of* `Massart-Boost`*'s oracles and subroutines,*

$$\text{err}^D_{0\text{-}1}(H) \leq \kappa + \varepsilon$$

Proof. We begin by bounding the error on examples $(x,y) \in \mathscr{X}_t^s$. For all $(x,y) \in \mathscr{X}_t^s$, $H(x) \neq y$ if and only if $yG_t(x) \leq 0$, and therefore $\mu_t(x,y) = 1$. For all other examples, the measure $\mu_t(x,y) \geq 0$. From Lemma 4.6.4, we have that with all but probability $\delta/8$, $d(\mu_t) \leq \kappa + \varepsilon/2$ upon termination. Conditioning on this event and considering the minimum contribution to the density by all examples misclassified by H, we have

$$\begin{aligned}
\kappa + \varepsilon/2 &\geq \mathop{\mathbb{E}}_{(x,y)\sim D} \mu(x,y) \\
&= \sum_{\substack{(x,y):\\ H(x)=y}} D(x,y)\mu(x,y) + \sum_{\substack{(x,y):\\ H(x)\neq y}} D(x,y)\mu(x,y) \\
&\geq \sum_{\substack{(x,y):\\ H(x)\neq y}} D(x,y) \\
&= \mathbf{Pr}_{(x,y)\sim D}[H(x) \neq y \big| x \in \mathscr{X}_t^s].
\end{aligned}$$

We now bound the error of H on examples $(x,y) \in \mathscr{X}_t^r$. When the algorithm terminates at round t, with all but probability $\delta_{\texttt{err}}$, at least one of the following holds.

1. $\mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \leq \varepsilon/2$
2. $\mathbf{Pr}_{(x,y)\sim D}[H(x)) \neq y \big| x \in \mathscr{X}_t^r] \leq \eta + \varepsilon$.

We first consider the case where, in the last round of boosting, `OverConfident` returns false. In this case, either the `OverConfident` routine estimated $\mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \leq \varepsilon/4$ or it estimated that $\mathbf{Pr}_{(x,y)\sim D}[H(x)) \neq y \big| x \in \mathscr{X}_t^r] \leq \eta + 3\varepsilon/4$. The routine uses a sample of size $8\log(2/\delta_{\texttt{err}})/\varepsilon^2$ to estimate the probability that $x \in \mathscr{X}_t^r$, and so the probability of underestimating this quantity by more than $\varepsilon/4$ is no more than $\delta_{\texttt{err}}/2$, by the Chernoff-Hoeffding inequality. Similarly, the routine uses a sample of size $8\log(2/\delta_{\texttt{err}})/\varepsilon^2$ to estimate the error on examples such that $x \in \mathscr{X}_t^r$, and so underestimates this error by more than $\varepsilon/4$ with probability no greater than $\delta_{\texttt{err}}/2$. So if `OverConfident` returns false, at least one of the lemma's conditions hold with probability at least $1 - \delta_{\texttt{err}}$.

If `OverConfident` returns true, then $|G_t(x)| = |G_{t-1}(x) + \lambda h_t^s(x)| - \lambda$. From Lemma 4.6.2, we know $|G_{t-1}(x)| < s + \lambda$ for all x, and $h_t^s(x) = 0$ for all x such that $|G_{t-1}(x)| \geq s$. It follows that $x \in \mathscr{X}_t^s$ for all x, and so $\mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \leq \varepsilon/2$.

It remains to bound the total error of H. The error bound for $x \in \mathscr{X}_t^s$ shows

$$\mathrm{err}_{0\text{-}1}^{D}(H) \leq \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^s](\kappa + \varepsilon/2) + \mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \cdot \mathbf{Pr}_{(x,y)\sim D}[H(x) \neq y \big| x \in \mathscr{X}_t^r.]$$

We have also just shown tells us that with all but probability $\delta_{\texttt{err}}$, either

$$\mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \leq \varepsilon/2$$

or

$$\mathbf{Pr}_{(x,y)\sim D}[H(x) \neq y \big| x \in \mathscr{X}_t^r] \leq \eta + \varepsilon.$$

We took $\kappa \geq \eta$, so in either case,

$$\text{err}_{0\text{-}1}^{D}(H) \leq \kappa + \varepsilon$$

and so the claimed error with respect to the labels holds except with probability $\delta_{\texttt{err}} + \delta/8 \leq \delta/4$. □

Lemma 4.6.6 (Target function error)**.** *When the algorithm terminates, with all but probability $\delta/4$,*

$$\text{err}_{0\text{-}1}^{D_x,f}(H) \leq \frac{\kappa + \varepsilon}{1 - \eta}$$

Proof. Lemma 4.6.5 shows that when the algorithm terminates, with all but probability $\delta/4$,

$$\text{err}_{0\text{-}1}^{D}(H) \leq \kappa + \varepsilon,$$

so we consider the worst-case difference between misclassification error and target function error.

$$\begin{aligned}
\kappa + \varepsilon &\geq \text{err}_{0\text{-}1}^{D}(H) \\
&= \mathbf{Pr}_{x \sim D_x}[H(x) \neq f(x)] \cdot \mathbf{Pr}_{(x,y) \sim D}[y = f(x) \big| H(x) \neq f(x)] \\
&\qquad + \mathbf{Pr}_{x \sim D_x}[H(x) = f(x)] \cdot \mathbf{Pr}_{(x,y) \sim D}[y \neq f(x) \big| H(x) = f(x)] \\
&\geq \mathbf{Pr}_{x \sim D_x}[H(x) \neq f(x)] \cdot \mathbf{Pr}_{(x,y) \sim D}[y = f(x) \big| H(x) \neq f(x)] \\
&\geq \mathbf{Pr}_{x \sim D_x}[H(x) \neq f(x)](1 - \eta) \\
&= \text{err}_{0\text{-}1}^{D_x,f}(H)(1 - \eta)
\end{aligned}$$

and so $\text{err}_{0\text{-}1}^{D_x,f} \leq \frac{\kappa+\varepsilon}{1-\eta}$ with all but probability $\delta/4$. □

4.6.5 Sample Complexity Analysis

In this subsection, we give sample complexity bounds for `Massart-Boost`'s subroutines during a single round of boosting. In all of the following lemmas, we assume that `Massart-Boost` is being run with an (α, γ)-Massart noise weak learner requiring a sample of size $m_{\texttt{WkL}}$. As elsewhere, we use ε to denote the target error of the final hypothesis in excess of η, and use κ to denote the density of μ below which `Massart-Boost` terminates. Let $\delta_{\texttt{dens}}$ denote the probability that `Est-Density` fails to estimate the density of μ to within error $\beta = \min\{\varepsilon/2, \eta/4\}$ and let $\delta_{\texttt{err}}$ denote the probability that `OverConfident` fails to estimate the error of G_t on examples (x, y) such that $|G_t(x, y)| \geq s$.

Lemma 4.6.7 (Sample complexity of `Samp`)**.** *Let* $\delta_{\texttt{samp}} = \delta\eta\gamma^2/(1536\log(2/\delta_{\texttt{WkL}}))$ *and* $\delta_{\texttt{WkL}} = \delta\eta\gamma^2/1536$*. With all but probability* $\delta_{\texttt{samp}}$*, the* `Samp` *routine draws no more than*

$$m \in O\left(\frac{\log(1/\delta_{\texttt{Samp}})}{\kappa^2} + \frac{m_{\texttt{WkL}}}{\kappa}\right)$$

examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$*.*

Proof. Because `Massart-Boost` terminates once the density of the measure μ is estimated to be less than κ, and

$$\log(1/\delta_{\texttt{dens}})/(2\beta^2) \geq \max\{2\log(1/\delta_{\texttt{dens}})/\varepsilon^2, 8\log(1/\delta_{\texttt{dens}})/\eta^2\}$$

many samples are used to estimate $d(\mu)$, it holds with all but probability $\delta_{\texttt{dens}}$ that $d(\mu) \geq \kappa - \min\{\varepsilon/2, \eta/4\}$.

`Samp` terminates once it has kept $m_{\texttt{WkL}}$ examples, and so from Lemma 4.5.4 we can conclude that

$$\begin{aligned} m &= \frac{\log(1/\delta_{\texttt{Samp}})}{(\kappa - \min\{\varepsilon/2, \eta/4\})^2} + \frac{2m_{\texttt{WkL}}}{\kappa - \min\{\varepsilon/2, \eta/4\}} \\ &\in O\left(\frac{\log(1/\delta_{\texttt{Samp}})}{\kappa^2} + \frac{m_{\texttt{WkL}}}{\kappa}\right) \end{aligned}$$

examples suffice except with probability $\delta_{\texttt{Samp}}$. □

Lemma 4.6.8 (Sample complexity of testing weak hypotheses)**.** *Let $\delta_{\texttt{WkL}} = \delta\eta\gamma^2/1536$. With all but probability $\delta\eta\gamma^2/512$, at most*

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma))}{\kappa^2} + \frac{m_{\texttt{WkL}}\log(1/(\delta\eta\gamma))}{\kappa} + \frac{\log(1/(\delta\eta\gamma))}{\gamma^2\kappa}\right)$$

examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ are drawn to identify a good enough weak hypothesis.

Proof. We have just shown in Lemma 4.6.7 that, with all but probability $\delta_{\texttt{samp}}$,

$$m \in O\left(\frac{\log(1/\delta_{\texttt{Samp}})}{\kappa^2} + \frac{2m_{\texttt{WkL}}}{\kappa}\right)$$

examples are required to draw a sample for WkL. Recall from Definition 4.2.5 that we assume WkL has failure probability 1/3, and from Lemma 4.5.1, that we invoke WkL on $2\log(2/\delta_{\texttt{WkL}})$ different samples to ensure we have at least one hypothesis with advantage γ, except with probability $\delta_{\texttt{WkL}}/2$. To estimate which hypothesis is best, we draw $2\log(2/\delta_{\texttt{WkL}})/\gamma^2$ examples from D_μ, against which we test each hypothesis. To draw these additional $2\log(2/\delta_{\texttt{WkL}})/\gamma^2$ examples from D_μ, with all but probability $\delta_{\texttt{samp}}$, we make at most

$$m \in O\left(\frac{\log(1/\delta_{\texttt{samp}})}{\kappa^2} + \frac{\log(1/\delta_{\texttt{WkL}})}{\kappa\gamma^2}\right)$$

calls to $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$.

We took $\delta_{\texttt{WkL}} = \delta\eta\gamma^2/1536$ and $\delta_{\texttt{samp}} = \delta\eta\gamma^2/(1536\log(2/\delta_{\texttt{WkL}}))$, so to repeatedly run the weak learner and identify a good enough hypothesis, we require

$$\begin{aligned} m &\in O\left(\frac{\log(1/\delta_{\texttt{WkL}})\log(1/\delta_{\texttt{Samp}})}{\kappa^2} + \frac{m_{\texttt{WkL}}\log(1/\delta_{\texttt{WkL}})}{\kappa} + \frac{\log(1/\delta_{\texttt{WkL}})}{\kappa\gamma^2}\right) \\ &\in O\left(\frac{\log(1/(\delta\eta\gamma))}{\kappa^2} + \frac{m_{\texttt{WkL}}\log(1/(\delta\eta\gamma))}{\kappa} + \frac{\log(1/(\delta\eta\gamma))}{\gamma^2\kappa}\right) \end{aligned}$$

examples, except with probability

$$\delta_{\texttt{Samp}} + 2\log(2/\delta_{\texttt{WkL}})\delta_{\texttt{samp}} \leq 3\log(2/\delta_{\texttt{WkL}})\delta_{\texttt{samp}} = \frac{\delta\eta\gamma^2}{512}.$$

□

Lemma 4.6.9 (Sample complexity of `OverConfident`). *With all but probability $\delta\eta\gamma^2/768$, the routine* `OverConfident` *draws no more than*

$$m \in O\left(\frac{\log(1/\delta\eta\gamma)}{\varepsilon^3}\right)$$

examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$.

Proof. `OverConfident` draws samples to estimate two population statistics: the probability that $x \in \mathscr{X}_t^r$ and, if that estimate exceeds $\varepsilon/4$, the error of G_t on examples such that $x \in \mathscr{X}_t^r$.

To estimate $\mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r]$ to within error $\varepsilon/8$ with all but probability $\delta_{\texttt{err}}/2$, it draws $32\log(2/\delta_{\texttt{err}})/\varepsilon^2$ examples. Then to estimate $\mathbb{E}_{(x,y)\sim D}[|y - \mathrm{sign}(G_t(x))| \,|\, x \in \mathscr{X}_t^r]$ to within error $\varepsilon/4$ with failure probability $\delta_{\texttt{err}}/2$, it uses a sample of size $8\log(2/\delta_{\texttt{err}})/\varepsilon^2$, but requires that all these examples satisfy $x \in \mathscr{X}_t^r$. As we know $\mathbf{Pr}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \geq \varepsilon/8$ with all but probability $\delta_{\texttt{err}}/2$, another use of the Chernoff-Hoeffding inequality allows us to upper-bound by $2\delta_{\texttt{err}}$ the probability that `OverConfident` draws more than

$$
\begin{aligned}
m &= \frac{64\log(1/\delta_{\texttt{err}})}{\varepsilon^2} + \frac{128\log(2/\delta_{\texttt{err}})}{\varepsilon^3} \\
&\in O\left(\frac{\log(1/\delta_{\texttt{err}})}{\varepsilon^3}\right) \\
&\in O\left(\frac{\log(1/\delta\eta\gamma)}{\varepsilon^3}\right)
\end{aligned}
$$

examples to estimate the error.

Therefore with all but probability $2\delta_{\texttt{err}} = \delta\eta\gamma^2/768$, `OverConfident` terminates having drawn no more than

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma)}{\varepsilon^3}\right)$$

examples. □

Lemma 4.6.10 (Sample complexity of one round)**.** *With all but probability $5\delta\eta\gamma^2/1536$, one round of boosting with* `WkL` *draws no more than*

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma)}{\min\{\varepsilon,\eta\}^2} + \frac{\log(1/(\delta\eta\gamma)}{\varepsilon^3} + \frac{\log(1/(\delta\eta\gamma))}{\kappa^2} + \frac{m_{\texttt{WkL}}\log(1/(\delta\eta\gamma))}{\kappa} + \frac{\log(1/(\delta\eta\gamma))}{\gamma^2\kappa}\right)$$

examples from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$.

Proof. In a single round of boosting, at most one call is made to `Est-Density` and `OverConfident` routines, and one weak hypothesis is chosen; no calls to the example oracle $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ are otherwise made. The `Est-Density` procedure draws exactly

$$\frac{\log(1/\delta_{\texttt{dens}})}{2\min\{\varepsilon/2, \eta/4\}^2} \in O\left(\frac{\log(1/(\delta\eta\gamma)}{\min\{\varepsilon,\eta\}^2}\right)$$

examples. Lemma 4.6.9 shows that, with all but probability $\delta\eta\gamma^2/768$, the `OverConfident` routine draws no more than

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma)}{\varepsilon^3}\right)$$

examples. Lemma 4.6.8 shows that, with all but probability $\delta\eta\gamma^2/512$, at most

$$O\left(\frac{\log(1/(\delta\eta\gamma))}{\kappa^2} + \frac{m_{\mathtt{WkL}}\log(1/(\delta\eta\gamma))}{\kappa} + \frac{\log(1/(\delta\eta\gamma))}{\gamma^2\kappa}\right)$$

examples are drawn to choose a weak hypothesis. So with all but probability $\delta\eta\gamma^2(\frac{1}{768}+\frac{1}{512}) = 5\delta\eta\gamma^2/1536$, a single round draws no more than

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma)}{\min\{\varepsilon,\eta\}^2} + \frac{\log(1/(\delta\eta\gamma)}{\varepsilon^3} + \frac{\log(1/(\delta\eta\gamma))}{\kappa^2} + \frac{m_{\mathtt{WkL}}\log(1/(\delta\eta\gamma))}{\kappa} + \frac{\log(1/(\delta\eta\gamma))}{\gamma^2\kappa}\right)$$

examples. □

4.6.6 Boosting Theorem

We can now put together the Lemmas of Section 4.6.3, Section 4.6.4, and Section 4.6.5 to prove our main result.

Theorem 4.6.1 (Boosting Theorem)**.** *Let* `WkL` *be an* (α,γ)*-weak learner requiring a sample of size* $m_{\mathtt{WkL}}$*. Then for any* $\delta \in (0,1/2]$*, any Massart distribution* D *with noise rate* $\eta < 1/2$*, and any* $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha}$*, taking* $\lambda = \gamma/8$ *and* $\kappa = \eta$*,* $\mathtt{Massart\text{-}Boost}^{\mathtt{WkL}}(\lambda,\kappa,\eta,\varepsilon,\delta,\alpha,\gamma,m_{\mathtt{WkL}})$ *will, with probability* $1-\delta$*,*

- *run for* $T \in O\left(1/(\eta\gamma^2)\right)$ *rounds*
- *output a hypothesis* H *such that* $\mathrm{err}_{0\text{-}1}^{D}(H) \leq \eta+\varepsilon$ *and* $\mathrm{err}_{0\text{-}1}^{D_x,f}(H) \leq \frac{\eta+\varepsilon}{1-\eta}$
- *make no more than* $m \in O\left(\frac{\log(1/(\delta\eta\gamma))}{\eta\gamma^2}\left(\frac{1}{\varepsilon^3}+\frac{1}{\eta^2}+\frac{m_{\mathtt{WkL}}}{\eta}+\frac{1}{\eta\gamma^2}\right)\right)$ *calls to* $\mathrm{EX}^{\mathrm{Mas}}(f,D_x,\eta(x))$
- *run in time* $O\left(\frac{\log(1/(\delta\eta\gamma))}{\eta^2\gamma^4}\left(\frac{1}{\varepsilon^3}+\frac{1}{\eta^2}+\frac{m_{\mathtt{WkL}}}{\eta}+\frac{1}{\eta\gamma^2}\right)\right)$*, neglecting the runtime of the weak learner.*

Proof. Lemma 4.6.4 shows that `Massart-Boost` terminates within $T \in O\left(1/(\eta\gamma^2)\right)$ rounds, except with probability $\delta/3$. From Lemmas 4.6.5 and 4.6.6, we have that with all but probability $\delta/4$, $\mathrm{err}_{0\text{-}1}^{D}(H) \leq \kappa + \varepsilon$ and $\mathrm{err}_{0\text{-}1}^{D_x,f}(H) \leq \frac{\kappa+\varepsilon}{1-\eta}$, so taking $\kappa = \eta$ gives

$$\mathrm{err}_{0\text{-}1}^{D}(H) \leq \eta + \varepsilon$$

and

$$\mathrm{err}_{0\text{-}1}^{D_x,f}(H) \leq \frac{\kappa+\varepsilon}{1-\eta}$$

To bound sample complexity, recall Lemma 4.6.10 implies that with all but probability $5\delta\eta\gamma^2/1536$, one round of boosting with `WkL` draws no more than

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma))}{\min\{\varepsilon,\eta\}^2} + \frac{\log(1/(\delta\eta\gamma)}{\varepsilon^3} + \frac{\log(1/(\delta\eta\gamma))}{\kappa^2} + \frac{m_{\mathtt{WkL}}\log(1/(\delta\eta\gamma))}{\kappa} + \frac{\log(1/(\delta\eta\gamma))}{\gamma^2\kappa}\right)$$

examples. We have taken $\kappa = \eta$, so union bounding the error probabilities over all $T \leq 128/\eta\gamma^2$ rounds of boosting gives us a sample bound of

$$m \in O\left(\frac{\log(1/(\delta\eta\gamma))}{\eta\gamma^2\varepsilon^3} + \frac{\log(1/(\delta\eta\gamma))}{\eta^3\gamma^2} + \frac{m_{\mathtt{WkL}}\log(1/(\delta\eta\gamma))}{\eta^2\gamma^2} + \frac{\log(1/(\delta\eta\gamma))}{\eta^2\gamma^4}\right)$$

exceeded with probability no more than

$$\frac{128}{\eta\gamma^2} \cdot \frac{5\delta\eta\gamma^2}{1536} = \frac{5\delta}{12}.$$

To prove the bound on overall runtime, we first observe that the runtime of a single round of `Massart-Boost`, neglecting calls to the weak learner, is linear in the runtime of subroutines `OverConfident` and `Est-Density`, and quasilinear in the runtime of `Samp` (from repetition of `WkL`). The runtime of each of these subroutines is dominated by computing $G_t(x)$ for each example drawn from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$, either to decide membership of x in $\mathscr{X}_t^r$ or to compute $\mu_t(x,y)$. The cost of evaluating G_t is linear in t, and so from our round and sample complexity bounds, we have the total runtime over all $T \in \left(1/\eta\gamma^2\right)$ rounds `Massart-Boost` is

$$\begin{aligned}
&O\left(T\left(\frac{\log(1/(\delta\eta\gamma))}{\eta\gamma^2\varepsilon^3}+\frac{\log(1/(\delta\eta\gamma))}{\eta^3\gamma^2}+\frac{m_{\mathtt{WkL}}\log(1/(\delta\eta\gamma))}{\eta^2\gamma^2}+\frac{\log(1/(\delta\eta\gamma))}{\eta^2\gamma^4}\right)\right)\\
&\in O\left(\frac{\log(1/(\delta\eta\gamma))}{\eta^2\gamma^4\varepsilon^3}+\frac{\log(1/(\delta\eta\gamma))}{\eta^4\gamma^4}+\frac{m_{\mathtt{WkL}}\log(1/(\delta\eta\gamma))}{\eta^3\gamma^4}+\frac{\log(1/(\delta\eta\gamma))}{\eta^3\gamma^6}\right)
\end{aligned}$$

Finally, we observe that the total probability of failure to achieve all of the claimed bounds is no more than $\frac{\delta}{3}+\frac{\delta}{4}+\frac{5\delta}{12}=\delta$, completing the proof. □

4.6.7 Final Hypothesis *H*

This subsection contains some explanation of the structure of the final hypothesis H output by our algorithm. We show that these hypotheses can be both efficiently represented and evaluated.

`Massart-Boost` maintains a function $G : \mathscr{X} \to \{\pm 1\}$, initialized to the zero function $G_0(x) = 0$. When `Massart-Boost` terminates at round t, it outputs the classifier $\operatorname{sign}(G_t)$. G_t can be computed from the threshold parameter s and a length-t sequence of pairs $((h_1,b_1),\ldots,(h_t,b_t)) \in (\mathscr{H} \times \{0,1\})^t$, where h_i is simply the weak learner hypothesis from round i, and $b_i = 1$ if `OverConfident` returned true at round i. $G_t(x)$ can then be efficiently computed by routine 17.

Routine 17. `ComputeG`$(x, s, (h_1, b_1), \ldots, (h_t, b_t))$, procedure for evaluating G, the classifier returned by `Massart-Boost`

```
$\sigma = 0$
for $i \in \{1, \ldots, t\}$ do
  if $|\sigma| < s$ then
    $\sigma \leftarrow \sigma + \lambda h_i(x)$
  else
    if $b_i = 1$ then
      $\sigma \leftarrow \sigma - \lambda$
return $\sigma$
```

Lemma 4.6.4 says we may assume $T \in \text{poly}(1/\eta, 1/\gamma)$, so long as the weak learner's hypotheses can be efficiently represented and evaluated, G_T can be as well, and of course $H = \text{sign}(G_T)$.

4.7 Appendix: Improved Round Complexity Analysis

In this section we revisit the round complexity of `Massart-Boost`. We show that a more careful use of the lower-bound on progress against our potential function (Lemma 4.3.3) proves convergence in $O\left(\frac{\log^2(1/\eta)}{\gamma^2}\right)$ rounds, saving nearly a factor η^{-1} in both time and sample complexity.

Recall that Lemma 4.3.3 shows that in each round of `Massart-Boost` we have

$$\Phi(t) - \Phi(t+1) \geq \frac{\gamma^2}{8}\left(d(\mu_t) - \frac{\eta}{2}\right)$$

for potential function

$$\Phi(t) = \mathop{\mathbb{E}}_{(x,y)\sim D}[\phi_t(x,y)] = \mathop{\mathbb{E}}_{(x,y)\sim D} \int_{yG_t(x)}^{\infty} M(z)dz.$$

For simplicity, Lemma 4.6.4 uses the fact that the algorithm terminates once it estimates $d(\mu) \leq \kappa$ to approximately lower-bound $d(\mu_t)$ by κ. Since we take $\kappa \geq \eta$, this lower-bounds the potential drop in each round by $O(\eta\gamma^2)$. However, this lower bound is loose at the beginning of the algorithm, when $d(\mu_0) = 1$. To use this observation to obtain a tighter analysis, we first lower-bound the density of the measure μ_t by the potential function $\Phi(t)$.

Lemma 4.7.1. *For every round t, with all but probability $\delta_{\texttt{err}} = \delta\eta\gamma^2/1536$, $\frac{\Phi_t}{s+\lambda+1} - 2(\eta+\varepsilon) \leq d(\mu_t)$.*

Proof. To show $\frac{\Phi_t}{s+\lambda+1} - 2(\eta+\varepsilon) \leq d(\mu_t)$, we independently consider the contribution to the density from examples $(x,y) \in \mathscr{X}_t^s$ and $(x,y) \in \mathscr{X}_t^r$ as follows,

$$\begin{aligned} d(\mu_t) &= \mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y)] \\ &= \mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y) \big| x \in \mathscr{X}_t^s] \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[x \in \mathscr{X}_t^s] \\ &\qquad + \mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y) \big| x \in \mathscr{X}_t^r] \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[x \in \mathscr{X}_t^r]. \end{aligned}$$

If $(x,y) \in \mathscr{X}_t^s$, one of two cases holds:

1. $-s < yG_t(x) \leq 0$, so $\mu_t(x,y) = 1$ and $\phi_t(x,y) = -yG_t(x) + 1 \leq s+1$
2. $0 < yG_t(x) < s$, so $\mu_t(x,y) = \exp(-yG_t(x))$ and $\phi_t(x,y) = \exp(-yG_t(x))$

both of which imply $\mu_t(x,y) \geq \phi_t(x,y)/(s+1)$, and so

$$\mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y) \big| x \in \mathscr{X}_t^s] \geq \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+1} \Big| x \in \mathscr{X}_t^s\right] \geq \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1} \Big| x \in \mathscr{X}_t^s\right] - 2(\eta+\varepsilon).$$

If $(x,y) \in \mathscr{X}_t^r$, then we again have two cases to consider:

1. $yG_t(x) \leq -s$, so $\mu_t(x,y) = 0$ and $\phi_t(x,y) = -yG_t(x) + 1 \leq s+\lambda+1$

2. $yG_t(x) \geq s$, so $\mu_t(x,y) = 0$ and $\phi_t(x,y) = \exp(-yG_t(x)) \leq (\eta + c)/(1-\eta)$.

We observe that examples (x,y) falling into case 2 satisfy

$$\mu_t(x,y) \geq \phi_t(x,y) - (\eta + c)/(1-\eta) \geq \phi_t(x,y) - \frac{\eta + c}{1-\eta},$$

and in case 1, $\mu_t(x,y) \geq \phi_t(x,y)/(s+\lambda+1) - 1$, so to prove our lower-bound on $d(\mu_t)$, we must upper-bound $\mathbf{Pr}_{(x,y)\sim D}[yG_t(x) \leq -s]$. By the definition of Algorithm 13, with all but probability $\delta_{\texttt{err}}$ over the coins of `OverConfident`, at the end of each round t either $\mathbf{Pr}_{x\sim D_x}[x \in \mathscr{X}_t^r] \leq \varepsilon/2$ or $\mathbf{Pr}_{(x,y)\sim D}[yG_t(x) \leq -s \mid x \in \mathscr{X}_t^r] \leq \eta + \varepsilon$.

If $\mathbf{Pr}_{x\sim D_x}[x \in \mathscr{X}_t^r] \leq \varepsilon/2$, then this gives us

$$\begin{aligned}
d(\mu_t) &\geq \left(1 - \frac{\varepsilon}{2}\right) \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1} \middle| x \in \mathscr{X}^s\right] + \frac{\varepsilon}{2} \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1} - 1 \middle| yG_t(x) \leq -s\right] \\
&\geq \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1}\right] - \frac{\varepsilon}{2} \\
&\geq \frac{\Phi(t)}{s+\lambda+1} - 2(\eta + \varepsilon),
\end{aligned}$$

and so the stated bound holds.

If $\mathbf{Pr}_{(x,y)\sim D}[yG_t(x) \leq -s \mid x \in \mathscr{X}_t^r] \leq \eta + \varepsilon$, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y)\big|x \in \mathscr{X}_t^r] \\
&\quad= \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1} - 1\big|yG_t(x) \leq -s\right] \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[yG_t(x) \leq -s\big|x \in \mathscr{X}_t^r] \\
&\qquad\quad+ \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\phi_t(x,y) - \frac{\eta+c}{1-\eta}\big|yG_t(x) \geq s\right] \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[yG_t(x) \geq s\big|x \in \mathscr{X}_t^r] \\
&\quad\geq (\eta+\varepsilon) \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1} - 1\big|yG_t(x) \leq -s\right] \\
&\qquad\quad+ (1-\eta-\varepsilon) \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\phi_t(x,y) - \frac{\eta+c}{1-\eta}\big|yG_t(x) \geq s\right] \\
&\quad\geq \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1}\big|x \in \mathscr{X}_t^r\right] - \eta - \varepsilon - (1-\eta-\varepsilon)(\frac{\eta+c}{1-\eta}) \\
&\quad\geq \mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1}\big|x \in \mathscr{X}_t^r\right] - 2(\eta+\varepsilon),
\end{aligned}$$

in which case it again holds that

$$\begin{aligned}
d(\mu_t) &= \mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y)\big|x \in \mathscr{X}_t^s] \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[x \in \mathscr{X}_t^s] \\
&\qquad+ \mathop{\mathbb{E}}_{(x,y)\sim D}[\mu_t(x,y)\big|x \in \mathscr{X}_t^r] \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \\
&\geq \left(\mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1}\big|x \in \mathscr{X}_t^s\right] - 2(\eta+\varepsilon)\right) \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[x \in \mathscr{X}_t^s] \\
&\qquad+ \left(\mathop{\mathbb{E}}_{(x,y)\sim D}\left[\frac{\phi_t(x,y)}{s+\lambda+1}\big|x \in \mathscr{X}_t^r\right] - 2(\eta+\varepsilon)\right) \cdot \mathop{\mathbf{Pr}}_{(x,y)\sim D}[x \in \mathscr{X}_t^r] \\
&= \frac{\Phi(t)}{s+\lambda+1} - 2(\eta+\varepsilon).
\end{aligned}$$

□

Now that we have a lower-bound on $d(\mu_t)$ in terms of $\Phi(t)$, we can show faster convergence and prove the following theorem.

Theorem 4.7.1 ((Improved) Boosting Theorem)**.** *Let* `WkL` *be an* (α, γ)*-weak learner requiring a sample of size* $m_{\texttt{WkL}}$*. Then for any* $\delta \in (0, 1/2]$*, any Massart distribution* D *with noise rate* $\eta < 1/2$*, and any* $\varepsilon \geq \frac{8\eta\alpha}{1-2\alpha}$*, taking* $\lambda = \gamma/8$ *and* $\kappa = \eta$*,* $\texttt{Massart-Boost}^{\texttt{WkL}}(\lambda, \kappa, \eta, \varepsilon, \delta, \gamma, \alpha, m_{\texttt{WkL}})$ *will, with probability* $1 - \delta$*,*

- *run for* $T \in O\left(\log^2(1/\eta)/\gamma^2\right)$ *rounds*
- *output a hypothesis* H *such that* $\mathrm{err}_{0\text{-}1}^{D}(H) \leq \eta + \varepsilon$ *and* $\mathrm{err}_{0\text{-}1}^{D_x,f}(H) \leq \frac{\eta+\varepsilon}{1-\eta}$
- *make no more than* $m \in O\left(\frac{\log^2(1/\eta)\log(1/(\delta\eta\gamma))}{\gamma^2}\left(\frac{1}{\varepsilon^3} + \frac{1}{\eta^2} + \frac{m_{\texttt{WkL}}}{\eta} + \frac{1}{\eta\gamma^2}\right)\right)$ *calls to example oracle* $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$*,*
- *run in time* $O\left(\frac{\log^4(1/\eta)\log(1/(\delta\eta\gamma))}{\gamma^4}\left(\frac{1}{\varepsilon^3} + \frac{1}{\eta^2} + \frac{m_{\texttt{WkL}}}{\eta} + \frac{1}{\eta\gamma^2}\right)\right)$*, neglecting the runtime of the weak learner.*

Proof. It follows from Lemma 4.6.3, Lemma 4.3.3, and Lemma 4.7.1 that

$$\begin{aligned}
\Phi(t+1) &\leq \Phi(t) - \frac{\gamma^2}{32}\left(d(\mu_t) - \frac{\eta}{2}\right) \\
&\leq \Phi(t) - \frac{\gamma^2}{32}\left(\frac{\Phi(t)}{s+\lambda+1} - 2\eta - \frac{\eta}{2}\right) \\
&\leq \Phi(t)\left(1 - \frac{\gamma^2}{64(s+1)}\right) + \frac{\eta\gamma^2}{8} && (\text{ from } s > \lambda).
\end{aligned}$$

Unrolling the recursion, we have that

$$\begin{aligned}
\Phi(t) &\leq \left(1 - \frac{\gamma^2}{64(s+1)}\right)^t + \frac{t\eta\gamma^2}{8} \\
&\leq e^{-\gamma^2 t/(64(s+1))} + \frac{t\eta\gamma^2}{8},
\end{aligned}$$

and so taking $t = 64\log(1/\eta)(s+1)/\gamma^2$ and Lemma 4.6.3 gives

$$\begin{aligned} d(\mu_t) &\leq \Phi(t) \\ &\leq \eta + 8\eta\log(1/\eta)(s+1) \\ &\leq \eta + 8\eta\log(1/\eta)(\log(1/\eta)+1) \\ &\leq \eta + 24\eta\log^2(1/\eta) \end{aligned}$$

where the last inequality follows from $2\log(1/\eta) > 1$ for all $\eta < 1/2$. As we have already shown a potential drop of at least $\frac{\gamma^2\eta}{32}$ at each step for which $d(\mu) \geq \eta$, running for an additional $768\log^2(1/\eta)/\gamma^2$ rounds suffices to guarantee $d(\mu) \leq \eta = \kappa$. This gives a total round complexity of

$$T \in O\left(\frac{\log^2(1/\eta)}{\gamma^2}\right).$$

The stated error bounds are the same as those proved in Theorem 4.6.1, and the tighter sample complexity and runtime follow immediately from the improved round complexity. □

4.8 Appendix: Lower Bound on Error for Massart Boosting

In this section, we show that no "black-box" generic boosting algorithm for Massart noise can have significantly better error than that of our algorithm, $\eta + \Theta(\alpha\eta)$. While the error term essentially matches the error lower bound of η for RCN boosters from [KS03], it is unclear from their result whether generalizing to Massart noise should imply a lower bound of OPT or a lower bound of η, since RCN is the special case of Massart noise where $\eta = \text{OPT}$. We show that the lower bound generalizes to the worst-case noise η, so long as OPT is not negligible in the input size. Therefore, no Massart-noise tolerant boosting algorithm can actually take advantage of a distribution with small expected noise to achieve accuracy better than its worst-case noise.

We consider the case where the target function $f \in \mathscr{C}$ is highly biased towards -1 labels (w.l.o.g.) and there is a small fraction of examples $(x,-1)$ where it cannot be distinguished whether $f(x)=1$ and $\eta(x)=0$, or $f(x)=-1$ and $\eta(x)>0$. As described in Section 4.1.3, if the booster does not reweight the distributions on which it queries the weak learner to emphasize examples labeled 1, an adversarial weak learner can return the constant function -1 and have high correlation. At the same time, if it does reweight its distribution to emphasize positively labeled examples, it risks violating the Massart condition by assigning to some $x \in \mathscr{X}$ a probability of appearing with its noisy label $y=-f(x)$ that is greater than $1/2-\alpha$.

Our adversarial, "rude" weak learner `rWkL` exploits this tension by providing information attainable without knowledge of f. `rWkL` returns a hypothesis h that does the following: on each heavy-hitter x of given distribution D', $h(x)$ is the majority vote label from examples; on all non-heavy-hitters, $h(x)=-1$. Assuming pseudorandomness of f, no efficient algorithm can use `rWkL` to learn f. So, the focus of the proof is showing that `rWkL` is a valid Massart noise weak learner.

By drawing polynomially many more examples than the black-box boosting algorithm, `rWkL` can *replicably* learn the heavy-hitters of its given distribution D'. Furthermore, assuming D' is Massart with noise rate $1/2-\alpha$, the labels of these heavy hitters $x \in \mathscr{X}$ must be biased towards the true label $f(x)$, guaranteeing advantage on heavy-hitters. To show `rWkL` also handles non-heavy-hitters correctly, we first show that boosting with `rWkL` can be efficiently simulated without knowing f, and then we appeal to the pseudorandomness of f. If reweighted distribution D' is Massart, then our adversarial weak learner only fails to return a hypothesis with γ advantage if D' is supported on many non-heavy-hitters x whose true label $f(x)=1$. We can conclude by observing that finding many such non-heavy-hitters implies a violation of the pseudorandomness assumption.

Theorem 4.8.1. *If one-way functions exist, then no black-box Massart noise-tolerant boosting algorithm achieves label error $\eta+o(\alpha\eta)$, even when* OPT $\ll \eta$.

We formalize the notion of black-box boosting and related definitions in Section 4.8.1. We describe the hard learning problem for the lower bound in Section 4.8.2. We describe our adversarial weak learner and note its useful properties in Section 4.8.3. In Section 4.8.4, we state and prove our lower bound.

4.8.1 Lower Bound Preliminaries

First, we define black-box boosting. In particular, we formalize the notion of a sampling procedure $\texttt{SP}$, the subroutine a boosting algorithm uses to construct weak learner queries from labeled examples. Recall the definition of an efficient Massart noise weak learner from Section 4.2:

Definition 4.2.5 (Massart Noise Weak Learner)**.** *Let $\mathscr{C}$ be a concept class of functions $f : \mathscr{X} \to \{\pm 1\}$. Let $\alpha \in [0, 1/2)$. Let $\gamma : \mathbb{R} \to \mathbb{R}$ be a function of α. A Massart noise (α, γ)-weak learner $\texttt{WkL}$ for $\mathscr{C}$ is an algorithm such that, for any distribution D_x over $\mathscr{X}$, function $f \in \mathscr{C}$, and noise function $\eta(x)$ with noise bound $\eta < 1/2 - \alpha$, $\texttt{WkL}$ outputs a hypothesis $h : \mathscr{X} \to \{\pm 1\}$ such that $\mathbf{Pr}_S[\mathrm{adv}^D(h) \geq \gamma] \geq 2/3$, where the sample S is drawn from Massart distribution $D = \texttt{Mas}\{f, D_x, \eta(x)\}$.*

Definition 4.5.2 (Efficient Massart Noise Weak Learner)**.** *Let $\texttt{WkL}$ be an (α, γ)-Massart noise weak learner. Let n be the maximum bit complexity of a single example $(x, y) \in \mathscr{X} \times \{\pm 1\}$, and let $m_{\texttt{WkL}}$ denote the number of examples comprising sample S. $\texttt{WkL}(S)$ is* efficient *if*

1. *$\texttt{WkL}$ uses $m_{\texttt{WkL}}(n, \eta, \gamma) = \mathrm{poly}(n, 1/(1 - 2\eta), 1/\gamma)$ examples.*

2. *$\texttt{WkL}$ outputs a hypothesis h in time $\mathrm{poly}(n, 1/(1 - 2\eta), 1/\gamma)$.*

3. *Hypothesis $h(x)$ has bit complexity $\mathrm{poly}(n, 1/(1 - 2\eta), 1/\gamma)$.*

4. *For all $x \in \mathscr{X}$, the hypothesis $h(x)$ can be evaluated in time $\mathrm{poly}(n, 1/(1 - 2\eta), 1/\gamma)$.*

We let $CT_{\texttt{WkL}}$ denote the time $\texttt{WkL}$ takes to output a hypothesis, BC_h denote the maximum bit complexity of a returned hypothesis h, and R_h denote the maximum time to evaluate a returned hypothesis h on any $x \in \mathscr{X}$. Recall that we define the runtime $R_{\texttt{WkL}}$ of $\texttt{WkL}$ as an upper bound on $CT_{\texttt{WkL}} + BC_h + R_h$.

For a boosting algorithm to construct new distributions to query the weak learner, the boosting algorithm must be able to convert examples from D into examples from a new distribution. We refer to this part of the boosting algorithm as a *sampling procedure* $\mathtt{SP}$.

Definition 4.8.2 (Sampling Procedure)**.** *A sampling procedure* $\mathtt{SP}^{\mathscr{O}}$ *is a probabilistic oracle algorithm that uses (potentially many) examples from* $\mathscr{O}$ *to return an example* $(x,y) \in \mathscr{X} \times \{\pm 1\}$.

We prove a lower bound against the following formulation of a black-box Massart boosting algorithm. In this setting, the boosting algorithm interacts with a example generator $\mathtt{EG}$, which generates examples for the weak learner. The boosting algorithm provides to $\mathtt{EG}$ an efficient sampling procedure $\mathtt{SP}$, as well as oracle access to its example oracle EX. The sampling procedure $\mathtt{SP}$ will induce a new distribution over $\mathscr{X} \times \{\pm 1\}$. We denote by $D^{\mathtt{SP}}$ the distribution induced by $\mathtt{SP}$ when supplied with EX as its example oracle. The weak learner $\mathtt{WkL}$ uses $m_{\mathtt{WkL}}$ examples drawn i.i.d. by $\mathtt{EG}$ to compute a hypothesis h, returned to the boosting algorithm. Note that the weak learner is required to return a hypothesis with advantage γ only if $D^{\mathtt{SP}}$ is a Massart noise distribution with noise bound $1/2 - \alpha$. For simplicity, we assume that the boosting algorithm and $\mathtt{EG}$ know the format of h and $\mathtt{SP}$, and that executing these subroutines can be done efficiently in their respective bit complexities.

Definition 4.8.3 (Black-box Massart Boosting Algorithm)**.** *Let* $\mathscr{C}$ *be a concept class over* $\mathscr{X}$*, and let* $f \in \mathscr{C}$ *be an unknown function. Let* n *denote the maximum bit complexity of an* $x \in \mathscr{X}$*. Let* D_x *be a fixed but unknown distribution over* $\mathscr{X}$*. Let* $\mathrm{EX} = \mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ *be a noisy example oracle for Massart noise distribution* $D = \mathtt{Mas}\{f, D_x, \eta(x)\}$*. Let* $\mathtt{EG}$ *be an example generator with query access to* EX*. Let* $\mathtt{WkL}$ *be an efficient* (α, γ)*-Massart noise weak learner with runtime* $R_{\mathtt{WkL}}$*, hypothesis bit complexity* BC_h*, and hypothesis evaluation time* R_h*. Let* $m_{\mathtt{WkL}}$ *denote the number of examples* $\mathtt{WkL}$ *requires. A black-box Massart boosting algorithm* $\mathtt{BlackBoxBoost}$*, with round bound* T *and sample complexity* m*, is a probabilistic polynomial-time algorithm with misclassification error* η^* *if* $\mathtt{BlackBoxBoost}$ *satisfies the following conditions:*

1. *Sample complexity m:* `BlackBoxBoost` *draws* $m = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$ *examples from sample oracle* EX.

2. *Round bound T:* `BlackBoxBoost` *queries* `WkL` *at most* $T = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$ *times.*

3. *Weak Learner Queries:* `BlackBoxBoost` *queries* `WkL` *by providing input* `SP` *to* `EG`*, where* $\texttt{SP}^{\text{EX}}$ *is an efficient sampling procedure satisfying the following conditions:*

   - `SP` *runs in time* $\text{poly}(n, m_{\texttt{WkL}}, 1/(1-2\eta), 1/\gamma, R_h)$.
   - `SP` *draws at most* $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$ *examples from* EX.
   - `SP` *is represented with bit complexity* $\text{poly}(n, m_{\texttt{WkL}}, 1/(1-2\eta), 1/\gamma, R_h)$.
   - `SP` *may use previous weak learner hypotheses as subroutines in* `SP`.

   `EG(SP)` *runs* `SP` $m_{\texttt{WkL}}$*-many times to generate a sample S containing* $m_{\texttt{WkL}}$ *examples.* `EG` *gives S to* `WkL`*, which returns a hypothesis h to* `BlackBoxBoost`.

4. *Correctness: If* `WkL` *returns a hypothesis with advantage γ over $D^{\texttt{SP}}$ in each round that $D^{\texttt{SP}}$ is a Massart distribution, then* `BlackBoxBoost` *returns a classifier $H : \mathscr{X} \to \{\pm 1\}$ with misclassification error* $\text{err}_{0\text{-}1}^{D}(H) < \eta^*$ *with constant probability.*

5. *Runtime:* `BlackBoxBoost` *runs in time* $\text{poly}(n, m_{\texttt{WkL}}, 1/(1-2\eta), 1/\gamma, R_h)$.

For clarity, the following pseudocode illustrates this black-box boosting framework.

Algorithm 18. Black-box Boosting Framework

```
Black-box boosting algorithm BlackBoxBoost draws $m$ examples from EX.
for $t = 1$ to $t = T$ do
    BlackBoxBoost constructs sample procedure $\mathtt{SP}_t$, possibly using hypotheses $h_1, \ldots, h_{t-1}$
    BlackBoxBoost gives $\mathtt{SP}_t$ to example generator EG
    Weak learner WkL gives $m_{\mathtt{WkL}}$ to EG
    EG uses $\mathtt{SP}_t$ to draw $m_{\mathtt{WkL}}$ i.i.d. examples from $D^{\mathtt{SP}_t}$. Let $S$ denote the set of these examples.
    EG gives sample $S$ to WkL
    WkL($S$) returns hypothesis $h_t$ to BlackBoxBoost
BlackBoxBoost outputs trained classifier $H$
```

The example generator EG is primarily used to correct the type mismatch between the boosting algorithm and weak learner. The boosting algorithm constructs distributions to query the weak learner, and the weak learner is defined to run on samples.

We will show that black-box Massart boosting algorithms cannot learn functions from pseudorandom function families with non-negligible probability. The following definition appears in [KS03]. As noted in [KS03], if one-way functions exist, then p-biased pseudorandom function families exist.

Definition 4.8.4 (p-biased Pseudorandom Function Family)**.** *For* $0 < p < 1$*, a p-biased pseudorandom function family is a family of functions* $\{f_s : \{0,1\}^{|s|} \to \{\pm 1\}\}_{s \in \{0,1\}^*}$ *which can be efficiently evaluated and satisfy the following p-biased pseudorandomness property:*

- *Efficient evaluation: There is a deterministic algorithm which, given an n-bit seed s and an n-bit input x, runs in time* $\text{poly}(n)$ *and outputs* $f_s(x)$*.*
- *p-biased pseudorandomness: Let* $\mathscr{F}_{n,p}$ *be the distribution over functions from* $\{0,1\}^n$ *to* $\{\pm 1\}$

such that function F has weight $p^{|F^{-1}(1)|}(1-p)^{|F^{-1}(-1)|}$. For all probabilistic polynomial time algorithms $\mathscr{A}$, the distinguishing advantage of $\mathscr{A}$ is a negligible function in n,

$$\left|\mathbf{Pr}_{F\sim\mathscr{F}_{n,p}}[\mathscr{A}^F(1^n)\Rightarrow 1]-\mathbf{Pr}_{s\sim\{0,1\}^n}[\mathscr{A}^{f_s}(1^n)\Rightarrow 1]\right| < \mathrm{negl}(n)$$

4.8.2 Adversarial Massart Distribution

Next, we describe the hard Massart noise learning problem used to prove our lower bound (Theorem 4.4.1). The following definitions apply to the remainder of this section.

Let $\eta\in[0,1/2), \alpha\in(0,1/2-\eta), \gamma(\alpha)=\alpha/20$. Define $\eta'=\eta(1+\alpha/5)$. Let $\{f_s:\{0,1\}^{|s|}\to\{\pm 1\}\}_{s\in\{0,1\}^*}$ be a η'-biased pseudorandom random function family with minority value 1.

Let n denote the security parameter, chosen to be at least a large polynomial in $1/(1-2\eta)$ and $1/\gamma$. Let $\mathscr{X}=\{0,1\}^n$, and let D_x be the uniform distribution over $\mathscr{X}$. For $s\in\{0,1\}^n$, let $\mathscr{C}_s$ be the concept class containing only the function $f_s:\{0,1\}^n\to\{\pm 1\}$.

The noise function $\eta(x)$ is chosen as follows. On the minority elements $x\in f_s^{-1}(1)$, let $\eta(x)=0$. On the majority elements $x\in f_s^{-1}(-1)$, let $\eta(x)=\eta$ for a random $\rho/(1-\eta')$-fraction of these x's, where $1/\mathrm{poly}(n)<\rho<\alpha/1000$. For the remaining elements, let $\eta(x)=0$. Finally, let Massart noise distribution $D=\texttt{Mas}\{f_s,D_x,\eta(x)\}$ with example oracle $\mathrm{EX}=\mathrm{EX}^{\mathrm{Mas}}(f_s,D_x,\eta(x))$. Note that the noise bound is η and $\mathrm{OPT}=\rho\eta$.

Throughout this section, we assume n is a polynomial in $1/(1-2\eta)$ and $1/\gamma$, so that we can assume the probability of EX returning the same data point $x\in\mathscr{X}$ more than once during the $\mathrm{poly}(n,1/(1-2\eta),\gamma)$ rounds of boosting is a negligible function in n.

4.8.3 Adversarial Weak Learner and Example Generator

In this section, we describe our adversarial weak learner `rWkL`, provide pseudocode, and prove that it has some nice properties. We also describe an example generator `rEG` that does not directly call EX.

Adversarial Weak Learner

We now define our "rude" weak learner $\texttt{rWkL}_{m,T}(S)$, which attempts to be maximally unhelpful by returning hypotheses h that rely entirely on majority vote labels. The weak learner $\texttt{rWkL}$ never provides the booster with any information about f_s that the booster could not have computed itself, and therefore the pseudorandomness of f_s will guarantee that the booster cannot boost $\texttt{rWkL}$ to obtain a hypothesis with error noticeably less than η'. The main technical challenge of proving our lower bound will come from showing that it is in fact possible for $\texttt{rWkL}$ to achieve noticeable advantage γ against all Massart distributions supplied to it by the booster, without revealing any information about f_s that cannot be efficiently simulated.

Recall that boosting algorithm $\texttt{BlackBoxBoost}$ invokes the weak learner by constructing $\texttt{SP}$, an efficient sampling procedure, which induces a distribution $D^{\texttt{SP}}$. The weak learner $\texttt{rWkL}$ attempts to return a hypothesis $h : \mathscr{X} \to \{\pm 1\}$ satisfying the following two conditions:

- For all $x \in \mathscr{X}$ that have large probability mass in $D^{\texttt{SP}}$ ($\approx \frac{\gamma}{10m}$ or larger), $h(x)$ is the most likely label for x under $D^{\texttt{SP}}$, i.e., $\text{sign}(\mathbb{E}_{(x^*,y)\sim D^{\texttt{SP}}}[y \mid x = x^*])$. We will refer to such x's as "heavy-hitters".

- For other x with smaller probability mass in $D^{\texttt{SP}}$, $h(x)$ is the most likely label for all non-heavy-hitters under $D^{\texttt{SP}}$, i.e., $\text{sign}(\mathbb{E}_{(x^*,y)\sim D^{\texttt{SP}}}[y \mid x^* \notin \mathscr{X}^{\text{H}}])$. The weak learner $\texttt{rWkL}$ is given access to m, the number of examples drawn by the boosting algorithm, so that $\texttt{rWkL}$ may accurately predict which examples x are heavy-hitters.

The weak learner identifies heavy-hitters using a two-step process. First, $\texttt{rWkL}$ uses a subset of its sample S to identify candidate heavy-hitters. It initially adds all x-values from this subset to the set of candidate heavy-hitters, $\mathscr{X}^{\text{H}}$. Next, $\texttt{rWkL}$ checks each $x \in \mathscr{X}^{\text{H}}$ to see if it is indeed a heavy-hitter of $D^{\texttt{SP}}$. Fresh examples from its samples S are used to empirically estimate this probability $\widehat{p}_x \stackrel{\text{def}}{=} \mathbf{Pr}_{(x',y')\sim S^{\text{SP}}}[x = x']$. The weak learner then randomly picks a value $v \in [\frac{\gamma}{20m}, \frac{\gamma}{10m}]$, and removes from $\mathscr{X}^{\text{H}}$ all x's for which $\widehat{p}_x < v$. This step ensures that, with high probability, $\mathscr{X}^{\text{H}}$ contains *exactly* v-heavy-hitters of D^{SP}.

The random choice of v_h and v_y will allow us to argue that, for fixed $v = (v_h, v_y)$, the hypothesis output by `rWkL` is not too sensitive to the specific sample drawn by `rWkL`. That is, if `rWkL` was repeatedly executed with the same choice of v, but different samples drawn from the same distribution, `rWkL` would output the same hypothesis with high probability. This stability property is fully justified in Subsection 4.8.3, but, informally, it will allow us to argue that the booster could simulate the example oracle EX itself when generating samples for `rWkL`, without making additional queries to its example oracle, and that with high probability the hypotheses output by `rWkL` would be the same in this case as those output when the sampling procedure queries EX. Analyzing the behavior of the boosting algorithm when the sampling procedure does not draw examples from EX (and therefore the labels of examples do not depend on f_s) simplifies the argument that `rWkL` can satisfy the definition of a Massart noise-tolerant weak learner without leaking information to the booster about f_s.

We now present pseudocode for our adversarial (α, γ)-weak learner.

Algorithm 19. $\texttt{rWkL}_{m,T}(S)$, adversarial weak learner
Precondition: S contains $m_{\texttt{WkL}}$ examples drawn i.i.d. from $D^{\texttt{SP}}$

```
if $m < n$ then
    $m = n$
                                                  // Step 1: Draw candidate heavy-hitters
$\mathscr{X}^{\mathrm{H}} \leftarrow x$-values from $O(m^2/\gamma)$ examples from $S$
$v_h \leftarrow_r [\frac{\gamma}{20m}, \frac{\gamma}{10m}]$ uniformly at random
for all $x \in \mathscr{X}^{\mathrm{H}}$ do          // Step 2: Remove non-$v$-heavy-hitters
    Estimate $\widehat{p}_x \overset{\text{def}}{=} \mathbf{Pr}[D^{\texttt{SP}}$ returns $x]$ using $O(m^{11}T^2/\gamma^4)$ fresh examples from $S$
    if $\widehat{p}_x < v_h$ then
        remove $x$ from $\mathscr{X}^{\mathrm{H}}$
$v_y \leftarrow_r [\frac{1}{2}, \frac{1}{2} + \frac{\gamma}{10m}]$
for all $x \in \mathscr{X}^{\mathrm{H}}$ do          // Step 3: Assign majority labels
    $S_x \leftarrow m^9 T^2/\gamma^4$ fresh examples from $S$
    $\widehat{p}_1 \leftarrow$ fraction of $S_x$ with label 1
    if $\widehat{p}_1 \geq v_y$ then
        $y_x = 1$
    else
        $y_x = -1$
$h(x) = \begin{cases} y_x & x \in \mathscr{X}^{\mathrm{H}} \\ -1 & \text{otherwise} \end{cases}$          // Step 4: Output hypothesis $h$
return $h \overset{\text{def}}{=} \{\mathscr{X}^{\mathrm{H}}, \{y_x\}\}$
```

This weak learner has polynomial sample complexity (Lemma 4.8.5), runs in polynomial time (Lemma 4.8.6), and does not use any hardcoded information about f_s, so `WkL` is efficiently simulatable (Lemma 4.8.7).

Example Generation

In this section, we define the two example generation procedures we will use in our lower bound argument: `hEG` and `rEG`.

Recall that an example generator `EG` is tasked with interfacing between the boosting algorithm, which creates reweighted distributions $D^{\mathtt{SP}}$, and the weak learner, which runs on samples S whose elements are drawn from $D^{\mathtt{SP}}$. To accomplish this, the example generator needs information from the weak learner and the boosting algorithm. The weak learner tells the example generator $m_{\mathtt{WkL}}$, the sample size it needs, and the boosting algorithm provides oracle access to its example oracle EX, as well as the sampling procedure `SP`. The example generator therefore invokes `SP` $m_{\mathtt{WkL}}$-many times, returning sample S.

Algorithm 20. $\mathtt{hEG}^{\mathrm{EX}}_{m_{\mathtt{WkL}}}(\mathtt{SP})$, honest example generator
Precondition: `SP` is a sampling procedure that returns an example (x,y)

```
$S = \emptyset$
**for** $i = 1$ to $i = m_{\mathtt{WkL}}$ **do**
    $(x,y) \leftarrow \mathtt{SP}^{\mathrm{EX}}$
    $S \leftarrow S \| (x,y)$
**return** $S$
```

Our second example generation procedure `rEG` behaves identically, except it never calls its oracle EX. Rather, `rEG` simulates calls to EX using EXSim. The routine EXSim draws x values from the same marginal distribution over $\mathscr{X}$ that EX does, $\mathscr{U}(\mathscr{X})$. It then generates the label y by taking $y = -1$ with probability $1 - \eta' - \rho + \rho\eta$, and $y = -1$ otherwise, in effect sampling from the same marginal distribution over ± 1 that EX does, but independent of the value x it has already drawn, and therefore independent of f_s.

Routine 21. EXSim, example oracle simulator

```
$x \leftarrow_r U_n$
$y = \begin{cases} -1 & \text{w. p. } 1-\eta'-\rho+\rho\eta \\ 1 & \text{o.w.} \end{cases}$        // i.e. $\mathbf{Pr}[y=1] = \mathbf{Pr}_{(x,y)\sim D}[y=1]$
return $(x,y)$
```

Algorithm 22. $\texttt{rEG}^{\text{EX}}_{m_{\texttt{WkL}}}(\texttt{SP})$, "rude" example generator
Precondition: $\texttt{SP}$ is a sampling procedure that returns an example (x,y)

```
$S = \emptyset$
for $i = 1$ to $i = m_{\texttt{WkL}}$ do
    $(x,y) \leftarrow \texttt{SP}^{\text{EXSim}}$
    $S \leftarrow S \| (x,y)$
return $S$
```

By pseudorandomness, we will show that with high probability over $v = (v_h, v_y)$, and over choice of S, S', where S is generated by $\texttt{hEG}$ and S' is generated by $\texttt{rEG}$, we have $\texttt{rWkL}(S;v) = \texttt{rWkL}(S';v)$ (Section 4.8.3).

Efficiency of $\texttt{rWkL}$ and $\texttt{rEG}$

In this section, we show that weak learner $\texttt{rWkL}$ and example generator $\texttt{rEG}$ are efficient and simulatable in polynomial time.

1. Efficiency of $\texttt{rWkL}$: polynomial sample complexity (Lemma 4.8.5) and polynomial runtime (Lemma 4.8.6).

2. Boosting with $\texttt{rWkL}$ and $\texttt{rEG}$ can be efficienctly simulated (Lemma 4.8.7).

Recall that $\texttt{SP}$ is a probabilistic algorithm that returns a labeled example. Let $m_{\texttt{SP}}$ denote the sample complexity of $\texttt{SP}$. Let $R_{\texttt{SP}}$ denote the runtime of $\texttt{SP}$ (including the time to query its oracle).

Lemma 4.8.5 (Sample Complexity of $\texttt{rWkL}$)**.** $m_{\texttt{rWkL}} = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$.

Proof. By Definition 4.8.3, $m_{\texttt{SP}} = \text{poly}(n, m, 1/(1-2\eta), 1/\gamma)$, $m = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$, and $T = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$. Step 1 requires $O(m^2T/\gamma)$ examples. Step 2 requires $O(m^{13}T^2/\gamma^5)$ examples. Step 3 requires $O(m^{10}T^2/\gamma^5)$ examples. Therefore Step 2 dominates the sample complexity of the weak learner, and $m_{\texttt{rWkL}} = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$ as claimed. $\square$

Lemma 4.8.6 (Runtime of $\texttt{rWkL}$)**.** $\texttt{rWkL}$ *runs in time* $R_{\texttt{rWkL}} = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$.

1. $\texttt{rWkL}$ *outputs hypothesis* h *in time* $CT_{\texttt{rWkL}} = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$.
2. *The maximum bit complexity of* h *is* $BC_h = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$.
3. *Hypothesis* h *can be evaluated in time* $R_h = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$.

Proof. By Definition 4.8.3, $m = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$, and $T = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$. Recall the runtime of a weak learner was defined as a bound on the sum of the three quantities listed in the lemma statement.

The hypotheses $h \overset{\text{def}}{=} \{\mathscr{X}^{\text{H}}, \{y_x\}\}$ output by $\texttt{rWkL}$ provides individual labels y_x for a maximum of $O(m^2/\gamma)$ elements in $\mathscr{X}^{\text{H}}$. Thus, $\{\mathscr{X}^{\text{H}}, \{y_x\}\}$ has bit complexity at most $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$. The instructions for executing this hypothesis can also be written using $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$ bits. For all $x \in \mathscr{X}$, $h(x)$ can be evaluated in time linear in the bit complexity of h. An algorithm can check if $x \in \mathscr{X}^{\text{H}}$ by scanning the representation of h for x, outputting y_x if found or -1 if not. Each step of $\texttt{rWkL}$ runs in time linear in the sample complexity of $\texttt{rWkL}$. By Lemma 4.8.5, $m_{\texttt{WkL}} = \text{poly}(n, 1/(1-2\eta, 1/\gamma)$. Thus, $\texttt{rWkL}$ outputs h in time $\text{poly}(n, 1/(1-2\eta, 1/\gamma)$. $\square$

Next, we argue that black-box boosting with $\texttt{rWkL}$ is efficiently simulatable. The following Lemma permits us to apply use the boosting algorithm in a distinguisher for pseudorandomness.

Lemma 4.8.7 (Boosting with `rWkL` and `rEG` can be efficiently simulated)**.** *Given query access to a function oracle for f_s, a probabilistic algorithm $\mathscr{A}$ can simulate* `BlackBoxBoost` *boosting the weak learner* `rWkL`*, using* `rEG` *to generate samples for* `rWkL`*, in time* $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$.

Proof. To simulate the initial m examples drawn by the booster, $\mathscr{A}$ simulates EX as follows. It draws a data point $x \in X$ uniformly at random from $\mathscr{X}$, and queries its function oracle on this point. If the label returned by the function oracle is 1, $\mathscr{A}$ returns $(x, 1)$. If the label is a -1, it will return $(x, 1)$ with probability $\rho\eta$, and $(x, -1)$ otherwise. Because $\mathscr{A}$ only has negligible probability of drawing the same x-value twice, and because the noise function $\eta(x)$ is both random and non-zero only on a ρ-sized fraction of negatively-labeled examples, the m examples drawn by this procedure are computationally indistinguishable from m examples drawn from EX, and so $\mathscr{A}$ successfully simulates the initial sample for `BlackBoxBoost`. The algorithm $\mathscr{A}$ can then run the algorithm `BlackBoxBoost`, which by Definition 4.8.3, runs in time $R_b = \text{poly}(n, m_{\texttt{rWkL}}, 1/(1-2\eta), 1/\gamma, R_h)$.

To simulate samples generated by `rEG`, $\mathscr{A}$ can simply run Algorithm 22, using Routine 21 for the oracle to the sampling procedure `SP`. We have just shown in Lemma 4.8.5 and Lemma 4.8.6 that $m_{\texttt{rWkL}}$ and R_h are both $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$, and because the weak learner uses no special hard-coded information about f_s, it can also be efficiently simulated by $\mathscr{A}$. These steps are repeated for $T = \text{poly}(n, 1/(1-2\eta), 1/\gamma)$ rounds of boosting, each of which is efficiently simulatable in time $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$. Any additional post-processing must also be efficiently simulatable, since `BlackBoxBoost` is assumed to run in time $\text{poly}(n, m_{\texttt{WkL}}, 1/(1-2\eta), 1/\gamma, R_h)$, and we have just shown that both $m_{\texttt{WkL}}$ and R_h are $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$. Therefore the entire interaction can be simulated by a probabilistic algorithm $\mathscr{A}$ with a function oracle for f_s, in time $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$.

□

`rWkL` is a Massart Noise-Tolerant Weak Learner

In this section, we analyze the advantage guarantee of `rWkL`. We begin by proving that the hypotheses output by `rWkL` satisfies replicability. We then use this property, along with pseudorandomness of f_s, to argue that with high probability over choice of sample S generated by `hEG`, S' generated by `rEG`, and randomness $v = (v_h, v_y)$, that $\texttt{rWkL}(S;v) = \texttt{rWkL}(S',v)$[2]. We then show that the hypothesis generated by `rWkL`, when run on a sample generated by `rEG`, will have good advantage against D^{SP}. Therefore the hypothesis generated by `rWkL` during a real run of the boosting algorithm, where the sample is generated by `hEG`, must also have good advantage against D^{SP}.

Replicability of Weak Hypotheses.

Recall that `rWkL` and `rEG` (or `hEG`) utilize randomness in two ways: i) to draw the input sample S, and ii) to pick thresholds v_h and v_y. Let h^v be the hypothesis that is most often returned when `rWkL` is run with thresholds $v = (v_h, v_y)$. In this section, we show that weak learner `rWkL` has the following property called *replicability*: for a fixed distribution D, with high probability over $S \sim D$ and randomly chosen v, the hypothesis output by $\texttt{rWkL}(S)$ is exactly h^v.[3] For any fixed v, we will refer to this hypothesis as the *canonical v-hypothesis* of `rWkL`.

First, we will show that `rWkL` run with `hEG` is replicable. In the next section, we apply pseudorandomness to show that with high probability, `rWkL` run with `rEG` outputs the same canonical v-hypothesis as it does when run with `hEG`.[4]

Recall that `rWkL` (Algorithm 19) is designed to return a hypothesis h that assigns majority vote labels to v_h-heavy-hitters of D^{SP}, where v_h is randomly chosen in the interval $[\frac{\gamma}{20m}, \frac{\gamma}{10m}]$.

Definition 4.8.8 (Heavy-Hitter). *Let D be a distribution over $\mathscr{X} \times \{\pm 1\}$. We call $x \in \mathscr{X}$ a* v-heavy-

[2]One can think of this property as a guarantee of replicability across shifted but close distributions.

[3]For this dissertation, this section was modified to more clearly connect this section to future work on replicability.

[4]Replicability is defined in [ILPS22] as stability of an algorithm between two runs, where the two samples sets are drawn from the *same* distribution. This result is a specific instance of a replicable algorithm being stable across *different* distributions. Here, the proof of "cross-distribution replicability" relies on the pseudorandomness assumption used throughout the lower bound argument.

hitter *of D if* $\mathbf{Pr}_{(x^*,y)\sim D}[x = x^*] > v$.

Recall that `rWkL` returns a hypothesis $h \overset{\text{def}}{=} \{\mathscr{X}^{\mathrm{H}}, \{y_x\}, b\}$. First, we show the consistency of $\mathscr{X}^{\mathrm{H}}$.

Lemma 4.8.9 (Consistency of $\mathscr{X}^{\mathrm{H}}$; $\mathscr{X}^{\mathrm{H}}$ is the set of v-heavy-hitters)**.** *Let D be any distribution over $\mathscr{X} \times \{\pm 1\}$, and let sample S be a set of $m_{\mathtt{rWkL}}$ examples drawn i.i.d. from D. Then with probability $1 - O(\frac{1}{mT})$ over the choice of S and $v_h \in [\frac{\gamma}{20m}, \frac{\gamma}{10m}]$ (Step 2 of Algorithm 19), the set $\mathscr{X}^{\mathrm{H}}$ computed by* `rWkL`(S) *is exactly the set of v_h-heavy-hitters of D.*

Proof. Recall that `rWkL` constructs a candidate list of heavy-hitters $\mathscr{X}^{\mathrm{H}}$ in Step 1 of Algorithm 19, and prunes that list in Step 2.

In Step 1, $\mathtt{rWkL}_{m,T}(S)$ uses $O(m^2/\gamma)$ examples to produce the initial set $\mathscr{X}^{\mathrm{H}}$. Let x be a v-heavy-hitter. The probability that $x \notin \mathscr{X}^{\mathrm{H}}$ by the end of Step 1 is at most

$$(1 - \gamma/(20m))^{m^2/\gamma} < \exp(-m/20).$$

Union bounding over the (at most) $20m/\gamma$ v_h-heavy hitters, the probability that the set $\mathscr{X}^{\mathrm{H}}$ does not initially contain all v-heavy hitters is negligible in m.

In Step 2, `rWkL` estimates $\widehat{p}_x$ for each $x \in \mathscr{X}^{\mathrm{H}}$ using $O(m^{11}T^2/\gamma^5)$ examples. The probability that a sample of this size contains fewer than $O(m^9T^2/\gamma^4)$ instances of x, given that x is a heavy-hitter, is negligible in m, by a Chernoff-Hoeffding bound. Given this many instances of x, the probability that the estimate $\widehat{p}_x$ has error greater than $O(\gamma^2/(m^4T))$ is again a negligible function in m by a Chernoff-Hoeffding bound. Recall `rWkL` chooses v_h uniformly at random from the interval $[\frac{\gamma}{20m}, \frac{\gamma}{10m}]$. The probability that v_h is chosen to be within distance $O(\gamma^2/(m^4T))$ of the probability of a specific $\gamma/(20m)$-heavy-hitter of D^{SP} is therefore no more than $O(\gamma/(m^3T))$. Union bounding over the at most $20m/\gamma$ heavy hitters, we have the following. Let S_0, S_1 be samples of size $m_{\mathtt{rWkL}}$ drawn from D^{SP}. Denote by $\mathscr{X}_0^{\mathrm{H}}(v_h)$ and $\mathscr{X}_1^{\mathrm{H}}(v_h)$ the sets of v_h-heavy-hitters estimated by `rWkL` provided $\mathscr{X}_0^{\mathrm{H}}$ and $\mathscr{X}_1^{\mathrm{H}}$ respectively. Then

$$\Pr_{\substack{S_0,S_1 \\ v_h\sim[\frac{\gamma}{20m},\frac{\gamma}{10m}]}}[\mathscr{X}_0^{\mathrm{H}}(v_h) \neq \mathscr{X}_1^{\mathrm{H}}(v_h)] \in O(1/(m^2T)).$$

It remains to show that, with high probability, all non-v-heavy-hitters are not included in $\mathscr{X}^{\mathrm{H}}$ after Step 2. There are at most $O(m^2/\gamma)$ candidate heavy hitters drawn in step 1. With all but negligible probability in m, `rWkL` estimates $\widehat{p}_x$ for all candidate heavy hitters to within error $O(\gamma^2/(m^4T))$. Then, as above, the probability that v_h is chosen to be within distance $O(\gamma^2/(m^4T))$ of the probability of a non-v_h-heavy-hitter of D^{SP} is no more than $O(\gamma/(m^3T))$, and union bounding over the $O(m^2/\gamma)$ candidates gives probability $O(1/(mT))$. Therefore with probability $1-O(\frac{1}{mT})$, at the end of Step 2, $\mathscr{X}^{\mathrm{H}}$ contains exactly the v_h-heavy-hitters of D^{SP}. □

Next, we show the consistency of $\{y_x\}$, the labels given by `rWkL`(S) to $x \in \mathscr{X}^{\mathrm{H}}$.

Lemma 4.8.10 (Replicability of h on heavy-hitters)**.** *Let D be a distribution over $\mathscr{X} \times \{\pm 1\}$, and let S_0 and S_1 be samples of $m_{\mathtt{rWkL}}$ examples drawn i.i.d. from D. Denote by h_0^v and h_1^v the output of* `rWkL`$(S_0; v_h = v)$ *and* `rWkL`$(S_1; v_h = v)$ *respectively. Let $\mathscr{X}_0^{\mathrm{H}}(v)$ and $\mathscr{X}_1^{\mathrm{H}}(v)$ denote the respective sets of v-heavy-hitters computed by* `rWkL`$(S_0; v_h = v)$ *and* `rWkL`$(S_1; v_h = v)$*. Then we have*

$$\Pr_{\substack{S_0,S_1 \\ v\sim[\frac{\gamma}{20m},\frac{\gamma}{10m}]}}[\mathscr{X}_0^{\mathrm{H}}(v) \neq \mathscr{X}_1^{\mathrm{H}}(v) \text{ or } \exists x \in \mathscr{X}_0^{\mathrm{H}}(v) \text{ s.t. } h_0(x) \neq h_1(x)] \in O(1/(mT)).$$

Proof. By Lemma 4.8.9, the probability that both $\mathscr{X}_0^{\mathrm{H}}(v)$ and $\mathscr{X}_1^{\mathrm{H}}(v)$ are exactly the set of v-heavy-hitters of D is at least $1-O(1/mT)$, over the choice of v, S_0, and S_1.

For each heavy-hitter x, `rWkL` uses $O(m^{11}T^2/\gamma^4)$ examples to estimate the probability that x has label 1 in D (Step 3 of Algorithm 19). Given that $x \in \mathscr{X}^{\mathrm{H}}$, the probability that this sample contains fewer than $O(m^7T^2/\gamma^4)$ instances of x is negligible in m. By a Chernoff-Hoeffding bound, this estimate has error at most $O(\gamma^2/(m^3T))$ with all but negligible probability in m. By an argument similar to the one of Lemma 4.8.9, the probability that v_y falls within $O(\gamma^2/(m^3T))$ of the true probability that x is labeled 1 in D is $O(\gamma/(m^2T))$. Union bounding over the (at most) $20m/\gamma$ heavy-hitters proves the claim

$$\Pr_{\substack{S_0,S_1 \\ v\sim[\frac{\gamma}{20m},\frac{\gamma}{10m}]}}[\mathscr{X}_0^{\mathrm{H}}(v) \neq \mathscr{X}_1^{\mathrm{H}}(v) \text{ or } \exists x \in \mathscr{X}_0^{\mathrm{H}}(v) \text{ s.t. } h_0(x) \neq h_1(x)] \in O(1/(mT)).$$

$\square$

Observing that `rWkL` outputs the constant function -1 on all non-heavy-hitters, we have the following corollary.

Corollary 4.8.11 (Replicability of h). *Let D be a distribution over $\mathscr{X}$, and let samples S_0, S_1 be two sets of $m_{\texttt{rWkL}}$ examples drawn i.i.d. from D. Let h_0^v and h_1^v denote the hypotheses output by* `rWkL`$(S_0; v)$ *and* `rWkL`$(S_1; v)$ *respectively. Then we have,*

$$\Pr_{S_0,S_1,v}[h_0^v \neq h_1^v] \in O(1/(mT)).$$

We now use the replicability of h and the pseudorandomness of $\{f_s\}$ to show that boosting `rWkL` run with `rEG` must also output the canonical v-hypothesis with high probability, unless $\Pr_{x\sim D_x^{\mathtt{SP}}}[x \notin \mathscr{X}^{\mathrm{H}}] < \gamma$.

Lemma 4.8.12 (`rWkL` does not distinguish between `hEG` and `rEG`). *Assume $\{f_s\}$ is a pseudorandom function family. Let $\{\mathtt{SP}_t\}_{t=1}^T$ be a sequence of sampling procedures constructed by the black-box boosting algorithm when boosting* `rWkL` *for T rounds. Let $D^{\mathtt{SP}_t}$ denote the distribution induced by $\mathtt{SP}_t$ and the honest example generator* `hEG`, *and let $D_r^{\mathtt{SP}_t}$ denote the distribution induced by $\mathtt{SP}_t$ and the random example generator* `rEG`. *Let S denote a sample of $m_{\texttt{rWkL}}$ examples drawn i.i.d. from $D^{\mathtt{SP}_t}$, and let S_r denote a sample of $m_{\texttt{rWkL}}$ examples drawn i.i.d. from $D_r^{\mathtt{SP}_t}$. Let h^v and h_r^v denote the* *hypotheses output by* $\texttt{rWkL}(S;v)$ *and* $\texttt{rWkL}(S_r;v)$ *respectively. Then for all* $t \in [T]$*,*

$$\Pr_{\substack{S,S_r\\v}}[h_r^v \neq h^v] \in O(1/(mT)).$$

Proof. By Corollary 4.8.11, we have that $\texttt{rWkL}(S)$ returns the canonical v-hypothesis h^v for D^{SP_t} with high probability over choice of v and S. Then if the claim does not hold, then it must be the case that there exists a round $t \in [T]$ such that, with probability $\omega(1/(mT))$ over choice of v, S, and S_r, we have $\texttt{rWkL}(S;v) \neq \texttt{rWkL}(S_r;v)$. Assuming this, we can construct the following distinguisher $\mathscr{A}$ against the pseudorandomness of $\{f_s\}$.

The distinguisher $\mathscr{A}$ executes the following procedure. It first chooses a round $t \in [T]$ uniformly at random, and simulates the interaction between the booster and rWkL until round t. At round t, $\mathscr{A}$ draws a sample S_0 of $m_{\texttt{rWkL}}$ examples from D^{SP_t} by simulating hEG. It then draws a sample S_1 of $m_{\texttt{rWkL}}$ examples by simulating rEG. It simulates rWkL on both of these samples using the same choice of randomness v for both simulations, and checks whether $\texttt{rWkL}(S_0;v) = \texttt{rWkL}(S_1;v)$. If not, it returns 1, and otherwise returns 0.

In the case that $\mathscr{A}$ is give oracle access to a random function F, both S_0 and S_1 are drawn from the same distribution, and so by Corollary 4.8.11, $\texttt{rWkL}(S_0;v) = \texttt{rWkL}(S_1;v)$ with probability $1 - O(1/(mT))$ over the choice of v, and therefore $\mathscr{A}$ outputs 1 with probability $O(1/(mT))$.

In the case that $\mathscr{A}$ is supplied a pseudorandom function f_s, by assumption there exists a round $t \in [T]$ at which $\Pr_{\substack{S_0,S_1\\v}}[\texttt{rWkL}(S_0;v) \neq \texttt{rWkL}(S_1;v)] \in \omega(1/(mT))$. Therefore in this case, $\mathscr{A}$ outputs 1 with probability noticeably (in n) larger than in the random case, and so $\mathscr{A}$ is a distinguisher against the pseudorandomness of f_s. This is a contradiction, and therefore the claim holds.

□

Informally, Lemma 4.8.12 will allow us to construct distinguishing adversaries against the pseudorandomness of f_s that make only m queries of their function oracle. In the following lemmas, we will prove that `rWkL` satisfies the definition of a Massart noise-tolerant weak learner when invoked on distributions constructed by the booster. That is, when `rWkL` is given a sample from a Massart distribution generated by the boosting algorithm, it returns a weak hypothesis with advantage γ with probability at least $2/3$. We will rely on appeals to the pseudorandomness of f_s in these proofs, by showing that failure of `rWkL` to return hypothesis with good advantage allows for the construction of distinguishers against the pseudorandomness of f_s. These distinguishers will simulate the boosting procedure, but it will be useful for our proofs to claim that the distinguishers can generate samples for `rWkL` without making additional queries to their function oracles to generate labels for these samples. Lemma 4.8.12 allows us to design distinguishers that use `rEG` to generate samples for `rWkL`, rather than generating samples using `hEG`. Recall that `rEG` makes no calls to the example oracle EX, and simply generates labels randomly for examples drawn from the underlying marginal distribution D_x. Therefore we will assume that our distinguishers only query their function oracles for the purposes of simulating the first m examples drawn by the booster.

Advantage of `rWkL`.

We will prove the following lemma by separately considering the advantage of weak hypotheses on heavy hitters of D^{SP} and non-heavy hitters.

Lemma 4.4.2 (Advantage of `rWkL`)**.** *Let D^{SP_t} denote the distribution induced by the sampling procedure SP_t and `hEG` at round $t \in [T]$ of boosting. Similarly, let $D_r^{\mathsf{SP}_t}$ denote the distribution induced by SP_t and `rEG`. Let S_t denote a sample drawn i.i.d. from $D_r^{\mathsf{SP}_t}$. Then for all* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$ *rounds of boosting `rWkL` with `rEG`, if D^{SP_t} is Massart, then with probability $1 - O(1/(mT))$ over its internal randomness, `rWkL`(S_t) outputs a hypothesis h_t with advantage at least γ against D^{SP_t}, except with negligible probability in m over the choice of SP_t.*

Recall that n is chosen to be a polynomial in $1/(1-2\eta)$ and $1/\gamma$, and D_x is the uniform distribution over $\{0,1\}^n$. By birthday-paradox-style arguments, with all but negligible probability in n, no $x \in \mathscr{X}$ is output more than once by EX throughout boosting. Henceforth, we assume no

$x \in \mathscr{X}$ is output more than once by EX.

Lemma 4.8.13 (`rWkL` advantage against heavy-hitters of $D^{\mathtt{SP}}$)**.** *Let $D^{\mathtt{SP}_t}$ be the distribution induced by the sampling procedure $\mathtt{SP}_t$ at round t. Similarly, let $D_r^{\mathtt{SP}_t}$ denote the distribution induced by $\mathtt{SP}_t$ and `rEG`. Let S_t denote a sample drawn i.i.d. from $D_r^{\mathtt{SP}_t}$, and let h_t be the hypothesis output by $\mathtt{rWkL}(S_t)$. Then for all* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$ *rounds of boosting* `rWkL` *with* `rEG`*, either*

1. $\mathbf{Pr}\left[\frac{1}{2}\mathbb{E}_{(x,y)\sim D^{\mathtt{SP}_t}}[h_t(x)y \mid x \in \mathscr{X}^{\mathrm{H}}] \geq \alpha\right] \in 1 - O(1/(mT))$
2. *or $D^{\mathtt{SP}_t}$ is not Massart.*

Proof. From replicability of h_t (Lemma 4.8.12), we have that with probability $1-O(1/(mT))$, `rWkL` outputs the same hypothesis that it would have had it been given a sample from $D^{\mathtt{SP}_t}$. For the remainder of the proof then, we will analyze the behavior of `rWkL` given such a sample from $D^{\mathtt{SP}_t}$, and show that it must have good advantage against the heavy-hitters of $D^{\mathtt{SP}_t}$.

Suppose the second case does not hold, and therefore $D^{\mathtt{SP}_t}$ is Massart. To compute y_x for each $x \in \mathscr{X}^{\mathrm{H}}$, `rWkL` uses m^9T^2/γ^4 examples from $D_r^{\mathtt{SP}_t}$. Because $x \in \mathscr{X}^{\mathrm{H}}$, $D_x^{\mathtt{SP}_t}(x) \geq \gamma/(40m)$ with high probability, and taking $\gamma = \alpha/20$, we have that at least $4m/\alpha^2$ instances of x occur in S_x (Step 3 of Algorithm 19) with all but negligible probability in m. The majority label of these $4m/\alpha^2$ examples is then taken to be the prediction of h_t on x, which will agree with $f(x)$ with all but negligible probability in m, because we have assumed $D^{\mathtt{SP}_t}$ is Massart, and so

$$\mathbf{Pr}_{(x,y)\sim D^{\mathtt{SP}_t}}[y = f(x) \mid x \in \mathscr{X}^{\mathrm{H}}] \geq 1/2 + \alpha.$$

It then follows that

$$\mathbf{Pr}\left[\frac{1}{2}\mathop{\mathbb{E}}_{(x,y)\sim D^{\mathtt{SP}_t}}[h_t(x)y \mid x \in \mathscr{X}^{\mathrm{H}}] < \alpha\right] \leq \mathrm{negl}(m)$$

or $D^{\mathtt{SP}_t}$ is not Massart, when h_t is the hypothesis output by `rWkL` given a sample S from $D^{\mathtt{SP}_t}$.

Applying Lemma 4.8.12 allows us to conclude that

$$\mathbf{Pr}\left[\frac{1}{2}\mathop{\mathbb{E}}_{(x,y)\sim D^{\mathrm{SP}_t}}[h_t(x)y \mid x \in \mathscr{X}^{\mathrm{H}}] < \alpha\right] \leq O(1/(mT))$$

or D^{SP_t} is not Massart, when $h_t \leftarrow \mathtt{rWkL}(S_t)$. □

Lemma 4.8.14 ($\mathtt{rWkL}$ advantage against non-heavy hitters of D^{SP})**.** *Let D^{SP_t} be the distribution induced by the sampling procedure $\mathtt{SP}_t$ at round t. Similarly, let $D_r^{\mathrm{SP}_t}$ denote the distribution induced by $\mathtt{SP}_t$ and $\mathtt{rEG}$. Let S_t denote a sample drawn i.i.d. from $D_r^{\mathrm{SP}_t}$, and let h_t be the hypothesis output by $\mathtt{rWkL}(S_t)$. Then for all* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$ *rounds of boosting* $\mathtt{rWkL}$ *with* $\mathtt{rEG}$*, with all but negligible probability in m over choice of* $\mathtt{SP}_t$*, either*

1. $\mathbf{Pr}_{S_t,v}\left[\frac{1}{2}\mathbb{E}_{(x,y)\sim D^{\mathrm{SP}_t}}[h_t(x)y \mid x \notin \mathscr{X}^{\mathrm{H}}] < \gamma\right] < 1/\mathrm{poly}(n)$
2. $\mathbf{Pr}_{(x,y)\sim D^{\mathrm{SP}_t}}[x \notin \mathscr{X}^{\mathrm{H}}] < \gamma$
3. *or D^{SP_t} is not Massart*

Proof. Suppose that the first two conditions fail, implying that there exists a round t of boosting for which the advantage of h_t on the non-heavy hitters of D^{SP_t} is less than γ, and that this will noticeably impact the overall advantage. Because h_t takes a constant value -1 on all non-heavy hitters, it must then be the case that

$$\mathbf{Pr}_{(x,y)\sim D^{\mathrm{SP}_t}}[y = 1 \mid x \notin \mathscr{X}^{\mathrm{H}}] > 1/2 - \gamma.$$

Since we are considering the advantage only on examples such that $x \notin \mathscr{X}^{\mathrm{H}}$, then $D^{\mathrm{SP}_t}(x,y) < \gamma/(10m)$ for all these examples. Furthermore, since we have assumed $\sum_{(x,y):x\notin\mathscr{X}^{\mathrm{H}}} D^{\mathrm{SP}_t}(x,y) \geq \gamma$, there must be at least $5m$ non-heavy-hitter examples x such that $D^{\mathrm{SP}_t}(x,1) > D^{\mathrm{SP}_t}(x,-1)$ in order for D^{SP_t} to satisfy $\mathbf{Pr}_{(x,y)\sim D^{\mathrm{SP}_t}}[y = 1 \mid x \notin \mathscr{X}^{\mathrm{H}}] > 1/2 - \gamma$. Then for D^{SP_t} to be Massart, it must hold that $f(x) = 1$ for every example x such that $D^{\mathrm{SP}_t}(x,1) > D^{\mathrm{SP}_t}(x,-1)$. However, if this is true with non-negligible probability in n, then we can construct the following distinguisher against f_s, which we denote by $\mathscr{A}$.

The distinguisher $\mathscr{A}$ simulates the boosting procedure run with `rWkL` and `rEG`, as described in Lemma 4.8.7 up until round t, chosen uniformly at random from $[1,T]$. Once the boosting procedure reaches round t, $\mathscr{A}$ simulates the t'th round of boosting and then queries its function oracle on all examples from the sample of the weak learner at that round that satisfy $D^{\mathsf{SP}_t}(x,1) > D_x^{\mathsf{SP}_t}(x)(1/2-\alpha)$. If $f(x)=1$ for all these examples, $\mathscr{A}$ outputs 1, and otherwise outputs 0.

To lower bound the advantage of our distinguisher, we will first show that there must be a significant number of examples drawn by `rWkL` in round t that satisfy $D^{\mathsf{SP}_t}(x,1) > D_x^{\mathsf{SP}_t}(x)(1/2-\alpha)$. We begin by lower bounding the probability that this condition holds for a single non-heavy hitter example.

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{x\sim D_x^{\mathsf{SP}_t}}[D^{\mathsf{SP}_t}(x,1) > D_x^{\mathsf{SP}_t}(x)(1/2-\alpha) \mid x \notin \mathscr{X}^{\mathrm{H}}] \\
&\qquad = 1 - \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathsf{SP}_t}}[D^{\mathsf{SP}_t}(x,1) \leq D_x^{\mathsf{SP}_t}(x)(1/2-\alpha) \mid x \notin \mathscr{X}^{\mathrm{H}}] \\
&\qquad = 1 - \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathsf{SP}_t}}[D^{\mathsf{SP}_t}(x,-1) \geq D_x^{\mathsf{SP}_t}(x)(1/2+\alpha) \mid x \notin \mathscr{X}^{\mathrm{H}}] \\
&\qquad \geq 1 - \frac{1+2\gamma}{1+2\alpha} \\
&\qquad = \frac{2(\alpha-\gamma)}{1+2\alpha} \\
&\qquad \geq \alpha-\gamma \\
&\qquad > \alpha/2,
\end{aligned}$$

where the last line follows from taking $\gamma = \alpha/20$, and the third line follows from observing that

$$\begin{aligned}
\Pr_{x\sim D_x^{\mathrm{SP}_t}}[D^{\mathrm{SP}_t}(x,-1) \geq D_x^{\mathrm{SP}_t}(x)(1/2+\alpha) \mid x \notin \mathscr{X}^{\mathrm{H}}] &= \sum_{\substack{x\notin\mathscr{X}^{\mathrm{H}} \\ D^{\mathrm{SP}_t}(x,-1)\geq D_x^{\mathrm{SP}_t}(x)(1/2+\alpha)}} D^{\mathrm{SP}_t}(x)\cdot\frac{1}{\Pr_{x\sim D_x^{\mathrm{SP}_t}}[x\notin\mathscr{X}^{\mathrm{H}}]} \\
&\leq \sum_{\substack{x\notin\mathscr{X}^{\mathrm{H}} \\ D^{\mathrm{SP}_t}(x,-1)\geq D_x^{\mathrm{SP}_t}(x)(1/2+\alpha)}} \frac{D^{\mathrm{SP}_t}(x,-1)}{1/2+\alpha}\cdot\frac{1}{\Pr_{x\sim D_x^{\mathrm{SP}_t}}[x\notin\mathscr{X}^{\mathrm{H}}]} \\
&\leq \frac{\Pr_{(x,y)\sim D^{\mathrm{SP}_t}}[y=-1 \mid x\notin\mathscr{X}^{\mathrm{H}}]}{1/2+\alpha}.
\end{aligned}$$

We have assumed that $\Pr_{(x,y)\sim D^{\mathrm{SP}_t}}[y=-1 \mid x\notin\mathscr{X}^{\mathrm{H}}] < 1/2+\gamma$, and so

$$\Pr_{x\sim D_x^{\mathrm{SP}_t}}[D^{\mathrm{SP}_t}(x,-1) \geq D_x^{\mathrm{SP}_t}(x)(1/2+\alpha) \mid x \notin \mathscr{X}^{\mathrm{H}}] \leq \frac{1+2\gamma}{1+2\alpha}.$$

Then because `rWkL` has a sample of size $O(m^{13}T^2/\gamma^5)$, we have that the probability that fewer than n of them satisfy $D^{\mathrm{SP}_t}(x,1) > D_x^{\mathrm{SP}_t}(x)(1/2-\alpha)$ must be negligible in m by the Chernoff-Hoeffding inequality.

Next, we bound the distinguishing advantage of $\mathscr{A}$, beginning with $\Pr_{F\sim\mathscr{F}_{n,\eta'}}[\mathscr{A}^F \Rightarrow 1]$. In the case that f is a random function, the boosting procedure can correctly identify an x such that $f(x)=1$ with probability no greater than $\frac{\eta'}{\eta'+(1-\eta')\rho\eta}$. This follows immediately from taking the largest of the following conditional probabilities:

$$\begin{aligned}
&\Pr_{(x,y)\sim D^{\mathrm{SP}_t}}[f(x)=1 \mid y=1] = \frac{\eta'}{\eta'+(1-\eta')\rho\eta} \\
&\Pr_{(x,y)\sim D^{\mathrm{SP}_t}}[f(x)=1 \mid y=-1] = 0 \\
&\Pr_{(x,y)\sim D^{\mathrm{SP}_t}}[f(x)=1 \mid y \text{ unknown }] = \eta'.
\end{aligned}$$

So if f is a truly random function then the boosting procedure has probability no more than $(\frac{\eta'}{\eta'+(1-\eta')\rho\eta})^n$ of correctly identifying at least n non-heavy hitter preimages of 1 under f. We have just shown that with all but negligible probability, `rWkL` draws at least n examples satisfying $D^{\mathsf{SP}_t}(x,1) > D_x^{\mathsf{SP}_t}(x)(1/2-\alpha)$, and $\mathscr{A}$ returns 1 only if all of these examples are preimages of 1 under f. Therefore

$$\mathop{\mathbf{Pr}}_{F\sim\mathscr{F}_{n,\eta'}}[\mathscr{A}^F \Rightarrow 1] \leq \left(\frac{\eta'}{\eta'+(1-\eta')\rho\eta}\right)^n + \mathrm{negl}(n),$$

where the additive $\mathrm{negl}(n)$ term comes from the probability that fewer than n qualifying examples were drawn by `rWkL` in that round.

We now consider the case that $\mathscr{A}$ is provided f_s as its oracle. Towards contradiction we have assumed that there exists some round t at which, with probability p that is non-negligible in m, the booster produces a Massart distribution D^{SP_t} for which $\mathbf{Pr}_{(x,y)\sim D^{\mathsf{SP}_t}}[y=1 \mid x \notin \mathscr{X}^{\mathrm{H}}] > 1/2-\gamma$. Therefore with probability $1/T$ the distinguisher $\mathscr{A}$ will halt its simulation at this round, and so with probability p/T will produce such a distribution. Then with all but negligible probability, it will draw n examples such that $D^{\mathsf{SP}_t}(x,1) \geq D_x^{\mathsf{SP}_t}(x)(1/2-\alpha)$. Since the distribution is Massart, all of these examples must satisfy $f(x)=1$, and so we have

$$\mathop{\mathbf{Pr}}_{s\sim\{0,1\}^n}[\mathscr{A}^{f_s} \Rightarrow 1] = p/T - \mathrm{negl}(m),$$

which is non-negligible in m, and therefore n. Therefore $\mathscr{A}$ has distinguishing advantage

$$\mathop{\mathbf{Pr}}_{s\sim\{0,1\}^n}[\mathscr{A}^{f_s} \Rightarrow 1] - \mathop{\mathbf{Pr}}_{F\sim\mathscr{F}_{n,\eta'}}[\mathscr{A}^F \Rightarrow 1] > \mathrm{negl}(n),$$

which contradicts pseudorandomness of $\mathscr{F}_{n,\eta'}$. Therefore it must be the case that the boosting procedure only has negligible probability (in m) of generating a Massart distribution at any round that has at least γ probability mass assigned to non-heavy hitters, and for which the constant function -1 does not have advantage at least γ against non-heavy-hitters of D^{SP_t}.

□

We can now combine Lemma 4.8.13 and Lemma 4.8.14 to show that `rWkL`, given a sample generated by `rEG`, will output a hypothesis with good advantage against $D^{\mathtt{SP}_t}$.

Lemma 4.4.2 (Advantage of `rWkL`)**.** *Let $D^{\mathtt{SP}_t}$ denote the distribution induced by the sampling procedure $\mathtt{SP}_t$ and* `hEG` *at round $t \in [T]$ of boosting. Similarly, let $D_r^{\mathtt{SP}_t}$ denote the distribution induced by $\mathtt{SP}_t$ and* `rEG`*. Let S_t denote a sample drawn i.i.d. from $D_r^{\mathtt{SP}_t}$. Then for all* $\mathrm{poly}(n, 1/(1-2\eta), 1/\gamma)$ *rounds of boosting* `rWkL` *with* `rEG`*, if $D^{\mathtt{SP}_t}$ is Massart, then with probability* $1 - O(1/(mT))$ *over its internal randomness,* `rWkL`(S_t) *outputs a hypothesis h_t with advantage at least γ against $D^{\mathtt{SP}_t}$, except with negligible probability in m over the choice of* $\mathtt{SP}_t$*.*

Proof. The advantage of h_t against $D^{\mathtt{SP}_t}$ is $\frac{1}{2}\,\mathbb{E}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x)]$ where

$$\begin{aligned}
\mathop{\mathbb{E}}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x)] &= \mathop{\mathbb{E}}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x) \mid x \in \mathscr{X}^{\mathrm{H}}] \cdot \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathtt{SP}_t}}[x \in \mathscr{X}^{\mathrm{H}}] \\
&\quad + \mathop{\mathbb{E}}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x) \mid x \notin \mathscr{X}^{\mathrm{H}}] \cdot \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathtt{SP}_t}}[x \notin \mathscr{X}^{\mathrm{H}}] \\
&\geq \alpha \cdot (1 - \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathtt{SP}_t}}[x \notin \mathscr{X}^{\mathrm{H}}]) + \mathop{\mathbb{E}}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x) \mid x \notin \mathscr{X}^{\mathrm{H}}] \cdot \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathtt{SP}_t}}[x \notin \mathscr{X}^{\mathrm{H}}] \\
&= \alpha - \mathop{\mathbf{Pr}}_{x\sim D_x^{\mathtt{SP}_t}}[x \notin \mathscr{X}^{\mathrm{H}}] \cdot (\alpha - \mathop{\mathbb{E}}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x) \mid x \notin \mathscr{X}^{\mathrm{H}}]),
\end{aligned}$$

with all but probability $O(1/(mT))$, following from Lemma 4.8.13. From Lemma 4.8.14, we have that if $D^{\mathtt{SP}_t}$ is Massart, then with all but negligible probability, either $\mathbb{E}_{(x,y)\sim D^{\mathtt{SP}_t}}[yh(x) \mid x \notin \mathscr{X}^{\mathrm{H}}] \geq \gamma$ or $\mathbf{Pr}_{x\sim D_x^{\mathtt{SP}_t}}[x \notin \mathscr{X}^{\mathrm{H}}] < \gamma$. Therefore h has advantage at least γ against $D^{\mathtt{SP}_t}$ with probability at least $1 - O(1/(mT))$, and the claim holds. □

4.8.4 Lower Bound for Black-Box Massart Boosting

Finally, we prove that no black-box boosting algorithm can boost `rWkL` to misclassification error better than $\eta(1+o(\alpha))$ with noticeable probability. At a high level, the proof idea is that any black-box booster interacting with `rWkL` can be efficiently simulated, and so if a boosting algorithm was able to achieve misclassification error noticeably better than $\eta(1+o(\alpha))$ for $\{f_s\}$, then there must be a distinguisher against the pseudorandomness of this function family, and so such error cannot be achievable via black-box boosting algorithms so long as pseudorandom functions exist.

Theorem 4.4.1 (Error Lower Bound Theorem). *Let $\eta \in [0,1/2), \alpha \in (0,1/2-\eta)$. Let $\{f_s\}$ be an η'-biased pseudorandom function family with security parameter n, where $\eta' = \eta(1+\alpha/5)$. Let η, α be at least inversely polynomially in n bounded away from $1/2$. Then, for random s, no efficient black-box boosting algorithm* `BlackBoxBoost` *with example bound m running for T rounds, given query access to $(\alpha, \gamma(\alpha) \overset{\text{def}}{=} \alpha/20)$-weak learner* $\texttt{rWkL}_{m,T}$ *and* $\text{poly}(n, 1/(1-2\eta), 1/\gamma)$ *examples from example oracle* $\text{EX}(U_n, f_s, \eta(x))$*, can output a hypothesis with label error at most $\eta(1+o(\alpha))$. In particular, for all polynomials q, for all polynomial time black-box Massart boosting algorithms* `BlackBoxBoost` *with query access to* `rWkL` *and example oracle* EX*, for n sufficiently large,* $\mathbf{Pr}_{s\in U_n}\left[\text{err}_{\text{0-1}}^{U_n,f_s}(H) \le \eta'\right] < \frac{1}{q(n)}$*, where H is the trained classifier output by* `BlackBoxBoost`.

Proof of Theorem 4.4.1. Let $\eta' = \eta(1+c\alpha)$. Suppose that `BlackBoxBoost` achieves label error better than $\eta' - \varepsilon$, for some noticeable ε, and with noticeable probability δ. Then we can construct a distinguisher $\mathscr{A}$ for f_s as follows.

The distinguisher $\mathscr{A}$ simulates the interaction between the booster and `rWkL`, where the samples for `rWkL` are drawn by `rEG` (as described in Lemma 4.8.7). Once the booster outputs its final hypothesis H, $\mathscr{A}$ draws a set S of n/ε^2 elements from the uniform distribution over $\mathscr{X}$, restricted to examples on which it has not already queried its oracle. Because `rWkL` is being run on samples drawn by `rEG`, $\mathscr{A}$ will only have simulated EX, and therefore queried its oracle, for the m

examples used by the booster itself, and therefore n/ε^2 elements can be drawn efficiently and the restricted distribution has only negligible statistical distance from D_x. The distinguisher $\mathscr{A}$ then queries both H and its oracle on all elements of S, returning 1 if its oracle and H disagree on fewer than an $\eta' - \varepsilon/2$ fraction of the elements, and 0 otherwise.

To show that $\mathscr{A}$ has non-negligible advantage distinguishing f_s from a truly random function, we first consider the probability that $\mathscr{A}$ outputs 1 when given oracle access to a truly random function, drawn from $\mathscr{F}_{n,\eta'}$. Because $\mathscr{A}$ is checking $H(x) \neq f(x)$ only on examples it has not previously queried, once H is fixed, we have $\mathbf{Pr}_{x \sim \mathscr{U}(\mathscr{X})}[H(x) \neq f(x) \mid x \text{ not previously queried}] \geq \eta'$. Therefore

$$\begin{aligned}\mathop{\mathbf{Pr}}_{F \sim \mathscr{F}_{n,\eta'}}[\mathscr{A}^F \Rightarrow 1] &= \mathop{\mathbf{Pr}}_{\substack{F \sim \mathscr{F}_{n,\eta'} \\ S \sim \mathscr{U}(\mathscr{X})}}[\mathop{\mathbf{Pr}}_{x \sim S}[H(x) \neq F(x)] \leq \eta' - \varepsilon/2] \\ &\leq \operatorname{negl}(n),\end{aligned}$$

where the last line follows from a Chernoff-Hoeffding bound and the fact that $\mathscr{A}$ has drawn n/ε^2 elements from $\mathscr{X}$ to check.

We now consider the probability that $\mathscr{A}$ returns 1 when given oracle access to pseudorandom f_s. We have assumed that our booster has noticeable probability δ of outputting a hypothesis H with error less than $\eta' - \varepsilon$, and from Lemma 4.8.12, we have that

$$\begin{aligned}\mathop{\mathbf{Pr}}_{s \sim \{0,1\}^n}[\mathscr{A}^{f_s} \Rightarrow 1] &= \mathop{\mathbf{Pr}}_{\substack{s \sim \{0,1\}^n \\ S \sim \mathscr{U}(\mathscr{X})}}[\mathop{\mathbf{Pr}}_{x \sim S}[H(x) \neq f_s(x)] \leq \eta' - \varepsilon/2] \\ &\geq \delta(1 - \operatorname{negl}(n)).\end{aligned}$$

Since we have assumed δ is noticeable, and we have just shown that $\mathscr{A}$ has distinguishing advantage

$$\Pr_{s\sim\{0,1\}^n}[\mathscr{A}^{f_s} \Rightarrow 1] - \Pr_{F\sim\mathscr{F}_{n,\eta'}}[\mathscr{A}^F \Rightarrow 1] > \delta/2,$$

the distinguisher $\mathscr{A}$ contradicts the pseudorandomness of f_s, and therefore `rWkL` cannot be efficiently boosted to construct a hypothesis with error noticeably better than η' with any noticeable probability.

□

4.9 Appendix: Application: Massart Learning of Unions of High-Dimensional Rectangles

In this section, we exhibit a Massart weak learner for learning unions of rectangles and apply our boosting algorithm.

Definition 4.9.1 (Rectangle)**.** *A rectangle $B \in \mathbb{R}^d$ is an intersection of inequalities of the form $x \cdot v < t$, where $v \in \{\pm e_j : j \in [d]\}$ and $t \in \mathbb{R}$. We may write a rectangle as a set B of pairs (v,t), that has size at most $2d$.*

We are interested in learning concepts $f \in C$ that are indicator functions of unions of k rectangles $B_1, \ldots, B_k$. That is, the class $\mathscr{C}$ consists of functions:

$$f(x) = \begin{cases} +1 & \text{if } x \in \cup_{i\in[k]} \cap_{(v,t)\in B_i} [x \cdot v < t] \\ -1 & \text{otherwise} \end{cases}$$

We refer to the negation of $\cup_{i\in[k]} B_i$ as the "negative region". Our weak learner aims to find if possible a rectangle entirely contained in the negative region to get some advantage over a random guess. To this end, we establish a structural result which shows that unless an overwhelming part of the mass is positive, there always exists a rectangle with non-trivial mass that is contained in the negative region. Moreover this rectangle has a lot of structure as it consists of at most k inequalities.

Lemma 4.9.2 (Structural result)**.** *If the negative region has probability more than ε, there exists a rectangle contained in the negative region that has mass at least $\varepsilon/(2d)^k$. This rectangle can be written as an intersection of at most k inequalities.*

Proof. The negative region can be written as a union of $(2d)^k$ rectangles B' with at most k inequalities

$$\cup_{B' \in B_1 \times B_2 \times \cdots \times B_k} \cap_{(v,t) \in B'} [x.v \geq t]$$

by choosing which inequality is not satisfied in every rectangle.

Since the union of the rectangles covers is exactly the negative region and has mass at least ε, at least one rectangle B' has probability more than $\varepsilon/(2d)^k$. □

4.9.1 Weak Learner

Our weak learner exploits the structural result of Lemma 4.9.2 to obtain an advantage over a random guess. If the probability mass is overwhelmingly positive, then the hypothesis $h(x) = +1$ must correlate well with the observed labels. On the contrary, if there is sufficient negative mass, there must exist a rectangle where predicting $h(x) = -1$ correlates with the labels of the examples within that rectangle. This idea is presented in pseudo-code in $\texttt{WkL}_{\texttt{box}}$ and formalized in Lemma 4.9.3 which gives the guarantees of our weak learner.

Algorithm 23. $\texttt{WkL}_{\texttt{box}}{}^{\mathrm{EX}(f,D_x,\eta(x))}(d,k,\alpha)$, Massart noise weak learner for unions of rectangles

```
$S \leftarrow \frac{k\,O(d)^k}{\alpha^2}$ examples from EX
$S^- \leftarrow$ number of these examples labeled $-1$
if $\frac{|S^-|}{|S|} < \frac{\alpha}{2}$ then
    return $h = +1$                                  // the constant 1 hypothesis
else
    for all Rectangles $B$ = choice of $k$ examples and $k$ dimensions do
        $S_B \leftarrow \{(x,y) \in S | x \in B\}$
        $S_B^+ \leftarrow \{(x,y) \in S | x \in B, y = +1\}$
        $B_{best} \leftarrow B$ that minimizes $|S_B^+|/|S_B|$ and has $|S_B|/|S| > \frac{\alpha}{8(2d)^k}$.
    Let $z \in \{\pm 1\}$ be the best most popular label in $S \setminus S_{B_{best}}$
    Hypothesis $h(x) = \begin{cases} -1 & x \in B_{best} \\ z & \text{otherwise} \end{cases}$
    return $h$
```

Lemma 4.9.3. *The algorithm* $\texttt{WkL}_{\texttt{box}}$ *is a* $(\alpha, \frac{\alpha^2}{O(d)^k})$*-Weak Learner for unions of k rectangles in d dimensions. It requires* $k\frac{O(d)^k}{\alpha^2}$ *samples and runs in time* $\frac{k^k O(d)^{k^2+1}}{\alpha^{2k}}$.

Proof. The algorithm starts by drawing drawing a set S of $N = k\frac{O(d)^k}{\alpha^2}$ examples from EX. Since the VC-dimension of rectangles defined by k inequalities is $O(k)$ this guarantees that, with probability at least $2/3$, for any rectangle B, the empirical probabilities computed over the sample S are close to actual ones:

1. $|\mathbf{Pr}[x \in B] - \mathbf{Pr}_S[x \in B]| \leq \alpha/O(d)^k$

2. $|\mathbf{Pr}[y = +1 \text{ and } x \in B'] - \mathbf{Pr}_S[y = +1 \text{ and } x \in B']| \leq \alpha/O(d)^k$

3. $|\mathbf{Pr}[y=-1]-\mathbf{Pr}_S[y=-1]| \leq \alpha/O(d)^k \leq \frac{\alpha}{4}$

Therefore, in the case that $|S^-|/|S| < \frac{\alpha}{2}$, we have that $\mathbf{Pr}[y=-1] < \frac{3}{4}\alpha$. Thus, the hypothesis $h=+1$ gets error at most $\frac{3}{4}\alpha + (\frac{1}{2}-\alpha) \leq \frac{1}{2}-\frac{\alpha}{4}$.

Otherwise, there is at least $\frac{\alpha}{4}$ probability in the negative region. By Lemma 4.9.2, there is a rectangle B^* defined by k inequalities that is contained entirely in the negative region and has probability at least $\frac{\alpha}{4(2d)^k}$. For this rectangle B^* it holds that $\mathbf{Pr}[x \in B^*] \geq \frac{\alpha}{4(2d)^k}$ and $\mathbf{Pr}[y = +1|x \in B] \leq \frac{1}{2}-\alpha$. This means that within the sample S it holds that $\mathbf{Pr}_S[x \in B^*] > \frac{\alpha}{8(2d)^k}$ and $\mathbf{Pr}_S[y=+1|x \in B^*] \leq \frac{1}{2}-\frac{\alpha}{2}$. Thus, B_{best} will also satisfy $\mathbf{Pr}_S[y=+1|x \in B_{best}] \leq \frac{1}{2}-\frac{\alpha}{2}$. By the closeness guarantee of the empirical distribution, we get that $\mathbf{Pr}[y=+1|x \in B_{best}] \leq \frac{1}{2}-\frac{\alpha}{4}$ and $\mathbf{Pr}[x \in B_{best}] > \frac{\alpha}{9(2d)^k}$.

We now bound the error of the hypothesis

$$h(x) = \begin{cases} -1 & x \in B_{best} \\ z & \text{otherwise} \end{cases}$$

Within the region B_{best}, it achieves error at most $\frac{1}{2}-\frac{\alpha}{4}$, while outside of B_{best}, the error is at most. $\frac{1}{2}+\frac{\alpha}{O(d)^k}$. Thus, the total error is at most $\frac{1}{2}-\frac{\alpha^2}{O(d)^k}$ given that $\mathbf{Pr}[x \in B_{best}] > \frac{\alpha}{9(2d)^k}$.

The main computational step of the algorithm is searching over all rectangles with k inequalities. It suffices to only consider rectangles with samples as end points, thus the total runtime of the weak-learner is $O(dN)^k = \frac{k^k O(d)^{k^2+1}}{\alpha^{2k}}$ as for every inequality there are $2d$ choices for the direction v and N choices for the threshold t.

□

4.9.2 Putting Everything Together

Lemma 4.9.3 shows that $\texttt{WkL}_{\texttt{box}}$ is an $(\alpha, \frac{\alpha^2}{O(d)^k})$-Weak Learner for unions of k high-dimensional rectangles in d dimensions. Combined with Theorem 4.3.5 we get that:

Theorem 4.9.4. *There exists an algorithm that learns unions of k rectangles in d dimensions with Massart noise bounded by η, achieving misclassification error $\eta + \varepsilon$ for $\varepsilon > 0$. The total number of samples is $\frac{kd^{O(k)}}{\eta^2\varepsilon^8}$ and the total running time is $\frac{1}{\eta^3}\left(\frac{kd^k}{\varepsilon}\right)^{k+O(1)}$.*

Proof. Follows by a direct application of the weak learner to Theorem 4.3.5 for $\alpha = \varepsilon/8$ and $\gamma = \frac{\varepsilon^2}{O(d)^k}$. □

4.10 Glossary of Symbols

Problem Statement

$\mathscr{X}$ A large finite domain

D_x A distribution over $\mathscr{X}$

$\mathscr{C}$ A class of concepts from $\mathscr{X}$ to $\{\pm 1\}$

f The unknown function in $\mathscr{C}$ to be learned

$\eta(x)$ The Massart noise function

η The Massart noise parameter, an upper bound on the Massart noise function

$D = \texttt{Mas}\{f, D_x, \eta(x)\}$ A Massart distribution over $\mathscr{X} \times \{\pm 1\}$

$\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ The noisy example oracle

Weak Learners

`WkL` The (α, γ)-weak learner to be boosted

h A hypothesis returned by the weak learner

α The Massart noise tolerance of the weak learner $(1/2 - \alpha)$

γ The advantage of the weak learner

S A sample, i.e., a collection of labeled examples, $S \in (X \times \{\pm 1\})^m$

$m_{\texttt{WkL}}$ The sample complexity of the weak learner

$\delta_{\texttt{WkL}}$ The failure rate of the weak learner

Boosting Algorithm

$\eta + \varepsilon$ The target error rate of the boosting algorithm (PAC-learning parameter)

δ The target failure rate of the boosting algorithm (PAC-learning parameter)

$G : \mathscr{X} \to \mathbb{R}$ Determines the final classifier. Updated in each round of boosting

λ The learning rate of the boosting algorithm, chosen $\Theta(\gamma)$.

T The number of rounds of boosting (and weak learner queries)

$t \in [T]$ A single round of boosting, commonly used as a subscript

`Samp`, `Est-Density`, `OverConfident` The subroutines of boosting algorithm `Massart-Boost`

Boosting Algorithm – Reweighting Distributions

$\mu : X \times \{\pm 1\} \to [0,1]$ A “measure” function used to determine rejection sampling probabilities

D_μ The distribution induced by rejection sampling from $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$ according to μ

$d(\mu)$ The density of μ, $d(\mu) = \mathbb{E}_{(x,y)\sim D}[\mu(x,y)]$

κ The density below which the algorithm terminates. Needs to be larger than η for our potential argument. We want $\kappa \approx \eta$, since the algorithm cannot get error better than $\kappa + \varepsilon$.

$\mathscr{X}^r$ The set of "risky" $x \in \mathscr{X}$, i.e. $\{x \in \mathscr{X} \mid |G(x)| \geq s\}$. The weak learner is never given examples from $\mathscr{X}^r$ (i.e. $\mu(x,y) = 0$ if $x \in \mathscr{X}^r$), to ensure that the induced distribution D_μ is a Massart distribution with noise rate $(1/2 - \alpha)$.

$\mathscr{X}^s$ The set of "safe" $x \in \mathscr{X}$, i.e. $\{x \in \mathscr{X} \mid |G(x)| < s\}$.

$s = \log\left(\frac{1-\eta}{\eta+c}\right)$ The cutoff between risky and safe regions of $G(x)$

$c = \frac{4\eta\alpha}{1-2\alpha}$ A constant used to limit the noise rate of reweighted distributions

Boosting Algorithm – Analysis

Φ The global potential function $\mathbb{E}_{(x,y)\sim D}[\phi(x,y)]$

$\phi(x,y)$ The potential function of an example (x,y)

$M : \mathscr{X} \to [0,1]$ The "base" measure function, used to define both $\phi(x,y)$ and $\mu(x,y)$

Lower Bound

`rWkL` The adversarial, unboostable, "rude" weak learner

`BlackBoxBoost` A black-box boosting algorithm. For each weak learner query, `BlackBoxBoost` generates a sampling procedure `SP` and passes it to the example generator

`SP` A sampling procedure, i.e. an efficient routine to generate a sample S using $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$

$D^{\mathtt{SP}}$ The distribution induced by `SP` and $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$

`EG` An example generator. Generates a sample $S \sim_{\text{i.i.d.}} D^{\mathtt{SP}}$ using `SP`, and passes S to the weak learner. Resolves a type mismatch between boosting algorithms (which generate a distribution to query the weak learner) and weak learners (which take as input a sample drawn from a distribution) `hEG` An "honest" example generator. Runs `SP` without any alterations.

`rEG` A "rude" example generator. Runs `SP`, but simulates sampling from $\texttt{Mas}\{f, D_x, \eta(x)\}$ without using $\mathrm{EX}^{\mathrm{Mas}}(f, D_x, \eta(x))$. Used in conjunction with `rWkL` in the lower bound construction

η' The error lower bound parameter, $\eta' = \eta(1 + \Theta(\alpha))$

ρ The fraction of examples that are noisy in the lower bound construction.

OPT The average noise rate $\mathbb{E}_{(x,y)\sim D}[\eta(x)]$. In the lower bound construction, $\mathrm{OPT} = \rho\eta$

$\mathscr{X}^{\mathtt{H}}$ The set of heavy-hitters of distribution D

h^{ν} The canonical ν-hypothesis of `rWkL`.

Acknowledgements

Supported by NSF Award CCF-1652862 (CAREER), a Sloan Research Fellowship, and a DARPA Learning with Less Labels (LwLL) grant. Supported by the Simons Foundation and NSF grant CCF-1909634. Supported by NSF Award CCF-1553288 (CAREER) and a Sloan Research Fellowship. Supported by NSF grant CCF-2008006.

Chapter 4, in full, is a reprint of the material as it appeared in Proceedings of the 34th Annual Conference on Learning Theory 2022. Diakonikolas, Ilias; Impagliazzo, Russell; Kane, Daniel M.; Lei, Rex; Sorrell, Jessica; Tzamos, Christos. "Boosting in the Presence of Massart Noise". For this dissertation, the term "reproducibility" was modified to "replicability" where appropriate, and minor formatting edits were made to improve readability. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization

The notion of replicable algorithms was introduced in [ILPS22] to describe randomized algorithms that are stable under the resampling of their inputs. More precisely, a replicable algorithm gives the same output with high probability when its randomness is fixed and it is run on a new i.i.d. sample drawn from the same distribution. Using replicable algorithms for data analysis can facilitate the verification of published results by ensuring that the results of an analysis will be the same with high probability, even when that analysis is performed on a new data set.

In this work, we establish new connections and separations between replicability and standard notions of algorithmic stability. In particular, we give sample-efficient algorithmic reductions between perfect generalization, approximate differential privacy, and replicability for a broad class of statistical problems. Conversely, we show any such equivalence must break down computationally: there exist statistical problems that are easy under differential privacy, but that cannot be solved replicably without breaking public-key cryptography. Furthermore, these results are tight: our reductions are statistically optimal, and we show that any computational separation between DP and replicability must imply the existence of one-way functions.

Our statistical reductions give a new algorithmic framework for translating between notions of stability, which we instantiate to answer several open questions in replicability and privacy. This includes giving sample-efficient replicable algorithms for various PAC learning, distribution estimation, and distribution testing problems, algorithmic amplification of δ in approximate DP, conversions from item-level to user-level privacy, and the existence of private agnostic-to-realizable learning reductions under structured distributions.

5.1 Introduction

Replicability is the principle that the findings of an empirical study should remain the same when it is repeated on new data. Despite being a pillar of the scientific method, replicability is extremely difficult to ensure in today's complex data generation and analysis processes. Questionable research practices including misapplication of statistics, selective reporting of only the findings that appear most statistically significant, and the formulation of research hypotheses after the results are already known have been identified as causes of an ongoing "crisis of replicability" across the empirical sciences. Toward formulating solutions in the context of machine learning and algorithmic data analysis, Impagliazzo, Lei, Pitassi, and Sorrell [ILPS22] recently put forth a new definition of replicability for statistical learning algorithms.[1]

Definition 5.1.1. *A randomized algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ is ρ-replicable if for every distribution D over $\mathscr{X}$, we have*

$$\mathbf{Pr}[A(S_1;r) = A(S_2;r)] \geq 1 - \rho,$$

where $S_1, S_2 \in \mathscr{X}^n$ are independent sequences of i.i.d. samples from D, and r represents the coin tosses of the algorithm A.

That is, an algorithm (capturing an end-to-end data analysis process) is replicable if with high probability over the choice of two independent samples from the same distribution, it produces exactly the same output. If one research team shares both their replicable analysis process (A) and the random choices made along the way (r), then another research team can independently verify their conclusions by performing the same analysis on a fresh dataset.

Replicability is an extremely strong *stability* constraint to place on an algorithm. Informally, an algorithm is stable if its output is insensitive to small changes to its input. Nevertheless, replicability is achievable for many fundamental data analysis tasks, including statistical query

[1] [ILPS22] stated this definition under the name "reproducibility." See Section 5.2.6 for a discussion of why we refer to it as "replicability" instead.

learning, heavy hitter identification, approximate median finding, and large-margin halfspace learning [ILPS22, GKM21].

Replicability is not the first definition of algorithmic stability aimed at ensuring the utility and safety of modern data analysis. Others have played central roles in relatively mature areas such as differential privacy and adaptive data analysis. Some of the aforementioned replicable algorithms were, in fact, motivated or inspired by differentially private counterparts. Is there a systematic explanation for this? *What can we learn about the capabilities and limitations of replicable algorithms by relating replicability to other notions of algorithmic stability?*

Let us briefly recall the types of algorithmic stability that arise in these other areas:

Differential privacy.

A randomized algorithm is differentially private [DMNS16] if changing a single input record results in a small change in the distribution of the algorithm's output. When each input record corresponds to one individual's datum, differential privacy guarantees that nothing specific to any individual can be learned from the output of the algorithm. (See Section 5.2.4.) Differential privacy comes with a rich algorithmic toolkit and understanding of the feasibility of fundamental statistical tasks in query estimation, classification, regression, distribution estimation, hypothesis testing, and more.

Generalization in adaptive data analysis.

Generalization is the ability of a learning algorithm to reflect properties of a population, rather than just properties of a specific sample drawn from that population. Techniques for provably ensuring generalization form a hallmark of theoretical machine learning. However, generalization is particularly difficult to guarantee in settings where multiple analyses are performed adaptively on the same sample. Traditional notions of generalization do not hold up to downstream misinterpretation of results. For example, a classifier that encodes detailed information about its training sample in its lower order bits may generalize well, but can be used to construct a different classifier that behaves very differently on the sample than it does on the population. Interactive processes such as exploratory data analysis or feature selection followed by classification/regression can ruin the independence between the training sample and the method used to analyze it, invalidating standard generalization arguments.

Adaptivity in data analysis has been identified as one contributing factor to the replication crisis, and imposing stability conditions on learning algorithms offers solutions to this part of the problem. A variety of such stability conditions have been studied [DFH$^+$15a, DFH$^+$15b, BNS$^+$21, RZ16, CLN$^+$16, BF16, RRT$^+$16, BMN$^+$18, LS19, SZ20], each offering distinct advantages in terms of the breadth of their applicability and the quantitative parameters achievable. Two specific notions play a central role in this work. The first is *perfect generalization* [CLN$^+$16, BF16], which ensures that whatever can be inferred from the output of a learning algorithm when run on a sample S could have been learned just from the underlying population itself:

Definition 5.1.2. *An algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ is $(\beta, \varepsilon, \delta)$-perfectly generalizing if, for every distribution D over $\mathscr{X}$, there exists a distribution Sim_D such that, with probability at least $1-\beta$ over S consisting of n i.i.d. samples from D, and every set of outcomes $\mathscr{O} \subseteq \mathscr{Y}$,*

$$e^{-\varepsilon}(\mathbf{Pr}_{Sim_D}[\mathscr{O}] - \delta) \leq \mathbf{Pr}[A(S) \in \mathscr{O}] \leq e^{\varepsilon}\mathbf{Pr}_{Sim_D}[\mathscr{O}] + \delta. \tag{5.1}$$

The second is *max-information* [DFH$^+$15a] which constrains the amount of information revealed to an analyst about the training sample:

Definition 5.1.3. *An algorithm $A : \mathscr{X}^n \to \mathscr{Y}$ has (ε, δ)-max-information with respect to product distributions if for every set of outcomes $\mathscr{O} \subseteq (\mathscr{Y} \times \mathscr{X}^n)$ we have*

$$\mathbf{Pr}[(A(S), S) \in \mathscr{O}] \leq e^{\varepsilon}\mathbf{Pr}[(A(S), S') \in \mathscr{O}] + \delta,$$

where S and S' are independent samples of size n drawn i.i.d. from an arbitrary distribution D over

$\mathscr{X}$.

As with differential privacy, both perfect generalization and max-information are robust to post-processing.

Each stability definition described above is tailored to model a distinct desideratum. At first glance, they may all appear technically incomparable. For instance, differential privacy is stricter than the other definitions in that it holds in the worst case over all input datasets without any assumptions on the data-generating procedure. On the other hand, it is weaker in that it only requires insensitivity to changing one input record, rather than to resampling the entire input dataset as in max-information, perfect generalization, or replicability. Meanwhile, differential privacy, max-information, and perfect generalization quantify the sensitivity of the algorithm's output in a weaker way than replicability; the former three notions only require that the distributions on outputs are similar, whereas replicability demands that precisely the same output realization is obtained with high probability.

Nevertheless, the (surprising!) technical connections between these definitions have enabled substantial progress on the fundamental questions in their respective areas. For example, it was exactly the adaptive generalization guarantees of differential privacy that kickstarted the framework of adaptive data analysis from [DFH$^+$15b]; the definition of max-information was subsequently introduced [DFH$^+$15a] to unify existing analyses based on differential privacy and description length bounds. As another illustration, variants of replicability were introduced in [BLM20, GGKM21, GKM21] for purely technical reasons, as it was observed that such algorithms could be immediately used to construct differentially private ones. This connection was essential in proving the characterization of private PAC learnability in terms of the Littlestone dimension from online learning [ALMM19, BLM20]. In fact, this characterization shows, that, in principle a private PAC learner using n samples can be converted to a replicable PAC learner using a number of samples that is an exponential tower of height n, but it is non-constructive and does not suggest what such a learner looks like in general.

5.1.1 Our Main Results

Equivalences

Our main result is a complete characterization of the relationships between these quantities. We prove that all four central stability notions — replicability, differential privacy, perfect generalization, and bounded max-information w.r.t. product distributions — are equivalent to one another via constructive conversions that incur at most a near-quadratic overhead in sample complexity.

Our equivalences apply to an abstract and broad class of *statistical tasks* that capture learning from i.i.d. samples from a population. An instance of such a task is obtained by considering a distribution D from a pre-specified family of distributions. Given i.i.d. samples from D, the goal of a learning algorithm is to produce an outcome that is "good" for D with high probability. This formulation of a statistical task captures problems such as PAC learning, where a sample from D is a pair $(x, f(x)) \in \mathscr{X} \times \{0,1\}$ where x is drawn from an arbitrary marginal distribution over $\mathscr{X}$, and f is an arbitrary function from a fixed concept class H. A "good" outcome for such a distribution D is a hypothesis $h : \mathscr{X} \to \{0,1\}$ that well-approximates f on D. Many other objectives such as regression, distribution parameter estimation, distribution learning, hypothesis testing, and confidence interval construction can be naturally framed as statistical tasks. (See Section 5.6.4 for other examples.)

Figure 5.1 illustrates the known relationships between the various stability notions that hold with respect to any statistical task.

From these equivalences we obtain the following consequences, resolving several open questions.

$$\begin{tikzcd}
\boxed{(0.1)\text{-replicability}} \arrow[rrr, dashed, "n \mapsto n \cdot \frac{\log(1/\delta)}{\varepsilon}\text{, Thm. 5.3.1 [GKM21]}"] \arrow[d, dashed, bend left, "n \mapsto n \cdot \frac{\operatorname{poly}\log(1/\varepsilon, 1/\delta)}{\varepsilon^2}\ \text{Thm. 5.3.19}"] & & & \boxed{(\varepsilon,\delta)\text{-differential privacy}} \arrow[d, dashed, "n \mapsto n^2\ \text{Cor. 5.3.13 [RRST16]}"'] \\
\boxed{\begin{array}{c}(\delta,\varepsilon,\delta)\text{-one-way}\\ \text{perfect generalization}\end{array}} \arrow[u, dotted, bend left, "n \mapsto n\ \text{Thm. 5.3.17}"] & & & \boxed{\begin{array}{c}(\varepsilon,\delta)\text{-max-information}\\ \text{w.r.t. product distributions}\end{array}} \arrow[lll, "\text{Lem. 5.3.14}"]
\end{tikzcd}$$

Figure 5.1. Algorithmic relationships among replicability, differential privacy, max information, and perfect generalization.
The solid arrow from A to B means that every algorithm satisfying A also satisfies B. A dashed arrow means that for every statistical task, a solution satisfying A can be computationally efficiently transformed into a solution satisfying B with the stated blowup in sample complexity. The thin dotted arrow means an explicit transformation exists, but is not always computationally efficient, and assumes the outcome space is finite.
This figure suppresses constant factors everywhere and polynomial factors in δ, assumes ε is below a sufficiently small constant, and assumes that δ is a sufficiently small inverse polynomial in n.

Sample-efficient replicable algorithms.

Any differentially private algorithm solving a statistical task (with a finite outcome space) can be converted into a replicable algorithm solving the same task with a near-quadratic blowup in its sample complexity. Thus, the wealth of research on private algorithm design can be brought to bear on designing replicable algorithms. We illustrate this algorithmic paradigm by describing new replicable algorithms for some PAC learning, distribution parameter estimation, and distribution testing problems in Section 5.6.4.

Equivalence between perfect generalization and differential privacy.

For simplicity, the relationships summarized in Figure 5.1 are stated in terms of a one-way variant of perfect generalization, where only the inequality on the right of (5.1) is required to hold. But the original two-way definition turns out to be statistically equivalent for tasks with a finite outcome space. This is because a one-way perfectly generalizing algorithm can be converted to a replicable algorithm using Theorem 5.3.17, and Theorem 5.3.19 actually yields the stronger conversion back to a two-way perfectly generalizing algorithm (See Theorem 5.6.3). Thus, an (ε, δ)-differentially private algorithm (with a finite outcome space) can be converted to a perfectly generalizing one solving the same statistical task with a near quadratic blow-up in sample complexity. This resolves an open question of [CLN$^+$16]. Their work also gave a conversion from perfectly generalizing algorithms to differentially private ones with no sample complexity overhead, and while their transformation preserves accuracy for (agnostic) PAC learning, it is not clear how to analyze it for general statistical tasks. Our conversion from perfect generalization to replicability and then to differential privacy holds for all statistical tasks with a finite outcome space.

Converting item-level to user-level privacy.

Consider a "user-level" learning scenario in which n individuals each hold m training examples drawn i.i.d. from the same distribution. When is (ε, δ)-differentially private learning possible if we wish to guarantee privacy with respect to changing *all* of any individual's samples at once? Ghazi, Kumar, and Manurangsi [GKM21] showed that this is possible when $n \geq O(\log(1/\delta)/\varepsilon)$ and the task admits a replicable learner. For the special case of PAC learning a concept class H, they argued that this implies a user-level private learning algorithm whenever H is privately PAC learnable with respect to changing a single *sample*. They posed the open problem of extending this result beyond PAC learning, e.g., to private regression [JKT20, Gol21]. Our conversion from any differentially private algorithm to a replicable one implies that such a transformation is possible for *any* statistical task with a finite outcome space (Section 5.6.1). Moreover, one can always take each indvidual's number of samples m to be nearly quadratic in the sample complexity of the original item-level private learner.

Amplifying differential privacy parameters.

While almost all (ε, δ)-differentially private algorithms enjoy a mild $\propto \log(1/\delta)$ dependence in their sample complexity on the parameter δ, it was not known how to achieve this universally, say by amplifying large values of δ to asymptotically smaller ones. [BLM20] showed that for private PAC learning, such amplification is possible in principle, but posed the open question of giving an explicit amplification algorithm. By converting an (ε, δ)-differentially private algorithm with weak parameters to a replicable one, and then back to a differentially private one with strong parameters, we resolve this question for the general class of statistical tasks with a finite outcome space, and with a much milder sample complexity blowup (Section 5.6.2).

Agnostic-to-realizable reductions for distribution-family learning.

[HKLM22] introduced a simple and flexible framework for converting realizable PAC learners to agnostic learners without relying on uniform convergence arguments. The framework applies to diverse settings such as robust learning, fair learning, partial learning, and (as observed in this work) replicable learning, with differential privacy providing a notable exception.[2] While an agnostic-to-realizable reduction for private PAC learning is known [BNS16b, ABMS20], it relies on uniform convergence and is only known to hold in the distribution-free PAC model. By converting a realizable private learner to a realizable replicable learner, then to an agnostic replicable learner, and back to an agnostic private learner, we obtain a reduction that works in the absence of uniform convergence (Section 5.6.3). In particular, this reduction applies to the *distribution-family* learning model, where one is promised that the marginal distribution on unlabeled examples comes from a pre-specified family of distributions.

[2]We note the technique we introduce to adapt [HKLM22] to the replicable setting has no clear translation to the private setting.

Separating Stability: Computational Barriers and the Complexity of Correlated Sampling

All of the transformations appearing in Figure 5.1 preserve computational efficiency, with the lone exception of the transformation from perfectly generalizing algorithms to replicable ones. This transformation makes use of the technique of *correlated sampling* from the distribution of outputs of a perfectly generalizing algorithm A when run on a fixed sample S (elaborated on more in Sections 5.1.2 and 5.2.5). This step can be explicitly implemented via rejection sampling from the output space of A, with the rejection threshold determined by the probability mass function of $A(S)$, but in general it is not computationally efficient.

We show that under cryptographic assumptions, this is inherent (Section 5.4). Specifically, we show that under standard assumptions in public-key cryptography, there exists a statistical task that admits an efficient differentially private algorithm, but does not have any efficient replicable algorithm. The task is defined in terms of a public-key encryption scheme with the following rerandomizability property: Given a ciphertext $\mathsf{Enc}(\mathbf{pk}, b)$, there is an efficient algorithm producing a uniformly random encryption of b. Fixing such a rerandomizable PKE, the statistical task is as follows. Given a dataset consisting of random encryptions of the form $\mathsf{Enc}(\mathbf{pk}, b)$ where $\mathbf{pk}$ is a fixed public key and $b \in \{0,1\}$ is a fixed bit, output any encryption of b.

One can solve this problem differentially privately, essentially by choosing a random ciphertext from the input dataset and rerandomizing it. On the other hand, there is no efficient replicable algorithm for this task. If there were, then one could use the public key to produce many encryptions of 0 and 1 and run the replicable algorithm on the results to produce canonical ciphertexts c_0 and c_1, respectively. Then, given an unknown ciphertext, one could repeatedly rerandomize it, run the replicable algorithm on the results, and compare the answer to c_0 and to c_1 to identify the underlying plaintext.

We also show that cryptographic assumptions are necessary even to separate replicability from perfect generalization. Recalling again that the bottleneck in computationally equating the two notions is in implementing correlated sampling, we show in Section 5.4.2 that if one-way functions do not exist, then correlated sampling is always tractable. In addition to addressing a natural question about the complexity of correlated sampling, this shows that function inversion enables an efficient transformation from perfectly generalizing algorithms into replicable ones. (See Section 5.2.5 for more discussion.)

Separating Stability: Statistical Barriers

Our equivalences show that the sample complexities of perfectly generalizing and replicable learning are essentially equivalent. Moreover: (1) An approximate-DP algorithm can be converted to a perfectly generalizing/replicable algorithm with near-quadratic blowup; and (2) A perfectly generalizing/replicable algorithm can be converted to an approximate-DP one using roughly the same number of samples. We prove that both of these conversions are optimal by showing:

1. **Quadratic separations between differential privacy and (perfect generalization, replicability).** We first consider the problem of estimating the parameters of a product of d Bernoulli distributions. By simply taking the empirical mean of an input dataset, this problem can be solved using $O(\log d)$ without any stability constraints. However, with differential privacy, it is known that $\tilde{\Theta}(\sqrt{d})$ samples are necessary and sufficient. By adapting the "fingerprinting" method underlying these privacy lower bounds [BUV18, DSS$^+$15, BSU19] to perfect generalization, we prove that any perfectly generalizing or replicable algorithm for this problem requires $\tilde{\Omega}(d)$ samples (Section 5.5.1).

   By reducing from a variant of this one-way marginals problem, we also show a general lower bound for replicable agnostic learning. Namely, we show that every concept class H requires $\tilde{\Omega}(VC(H)^2)$ samples. For concept classes of maximal VC dimension $VC(H) = \log|H|$, this too gives a quadratic separation between replicable learning and both private and unconstrained learning (Section 5.5.2).

2. **No separation between differential privacy and (perfect generalization, replicability).** Complementing our lower bounds, we also show that every finite class H can be replicably PAC learned (in the realizable setting) to error α with sample complexity $\tilde{O}_H(1/\alpha)$ (Section 5.5.3). Up to logarithmic factors, this matches the learning rate achievable for both unconstrained and differentially private learning. Our learner works by selecting a random threshold v, and selecting a random concept from H whose error with respect to the sample is at most v. A more involved random thresholding strategy also yields an agnostic learner with sample complexity $\tilde{O}_H(1/\alpha^2)$.

5.1.2 Overview of Proofs of Equivalences

Perfect generalization is equivalent to replicability.

Recall that an algorithm is replicable if it is likely to produce exactly the same output when run on two independent samples from any given population. Replicability appears to be a dramatic strengthening of perfect generalization, which only requires the distributions of $A(S)$ and $A(S')$ to be statistically close. Nevertheless, we prove that perfectly generalizing algorithms can always be converted to replicable ones whenever the output space $\mathscr{Y}$ is finite (Theorem 5.3.17). This can be done via a primitive called *correlated sampling* (See Section 5.2.5). A correlated sampling algorithm for a class of distributions $\mathscr{P} = \{P\}$ is a procedure $CS(P,r)$ such that 1) $CS(P,r)$ produces a sample distributed according to P when provided a uniformly random input r, and 2) Whenever $P,Q \in \mathscr{P}$ satisfy $d_{\mathrm{TV}}(P,Q) \leq \eta$, we have $\mathbf{Pr}[CS(P,r) = CS(Q,r)] \geq 1 - O(\eta)$. That is, applying correlated sampling to two similar distributions results in the same output with high probability – exactly what is needed for replicability. We actually prove a stronger theorem, showing that the larger class of *one-way* perfectly generalizing algorithms (where only the right-hand inequality in 5.1 holds) are replicable via correlated sampling.

Conversely, we show how to convert replicable algorithms to perfectly generalizing ones (Theorem 5.3.19). While a ρ-replicable algorithm is automatically also a $(\beta = O(\rho), \varepsilon = 0, \delta =$

$O(\rho)$)-perfectly generalizing one, these parameters are too weak for applications where one wants to take β, δ to be inverse polynomial in the dataset size n (e.g., to prove the lower bounds in Section 5.5). To obtain a perfectly generalizing algorithm with stronger parameters, we repeatedly run the replicable algorithm using $k = O(\log(1/\delta))$ different sequences of coin tosses $r_1, \ldots, r_k$, and using $\tilde{O}(1/\varepsilon^2)$ independent samples for each sequence of coin tosses. Using the exponential mechanism from differential privacy [MT07], we select an outcome y_i that appears approximately the most frequently amongst these repetitions in a manner that ensures $(\beta = \delta, \varepsilon, \delta)$-perfect generalization. This strategy allows us to obtain inverse polynomial β, δ parameters with only a logarithmic multiplicative overhead in the number of samples.

Bounded max-information implies perfect generalization. In Lemma 5.3.14, we show that bounded max-information implies one-way perfect generalization with similar parameters. Namely, if an algorithm A has (ε, δ)-max-information with respect to product distributions, then it is also $(\sqrt{\delta}, 2\varepsilon, \sqrt{\delta})$-one-way perfectly generalizing. The idea is to take the simulator distribution Sim_D to be the distribution of $A(S')$, where the randomness is taken over both the coin tosses of A *and* the randomness of a sample $S' \sim D$. A similar argument is implicit in [BF16, Proof of Lemma 4.5]. Then by combining Theorems 5.3.17 and 5.3.19, it follows that bounded max-information also implies perfect generalization for finite outcome spaces (Theorem 5.6.3).

Replicability implies differential privacy.

In Theorem 5.3.1 we show that replicability implies differential privacy. Given a replicable algorithm, one can run it $k = O(\log(1/\delta)/\varepsilon)$ times using the same sequence of coin tosses, but on independent samples, producing outcomes $y_1, \ldots, y_k$. Replicability ensures that most of these outcomes are the same with high probability, and so this common outcome can be selected in a standard differentially private way. This argument appears in the differential privacy literature as a conversion from "globally stable" and "pseudo-globally stable" learners to private ones [BLM20, GGKM21, GKM21]. Our presentation of Theorem 5.3.1 includes an additional amplification step that avoids union bounding over correctness, making the conversion suitable for a broader range of parameters.

Differential privacy implies bounded max-information. The final conversion in Figure 5.1 is from differentially private algorithms to algorithms with bounded max-information. This argument is implicit in [RRST16] and we show how it follows from their work here (Corollaries 5.3.12 and 5.3.13).

5.1.3 Further Discussion of Related Work

Several elements of our approach were inspired by Ghazi, Kumar, and Manurangsi's study of the relationship between user-level and item-level differentially private learning [GKM21]. They introduced a notion of "pseudo-global stability" that is essentially the same as replicability, and showed that it implies differential privacy. Correlated sampling also played a crucial role in their work by allowing individuals to use shared randomness to reach consensus on a learned hypothesis. In fact, it provided a key step in their conversion from "list globally stable" algorithms [GGKM21] (learning algorithms that output a short list of hypotheses, one of which is almost guaranteed to be canonical for the given distribution) to pseudo-globally stable ones.

Stability has a long history as a tool for ensuring generalization. Early work [RW78, DW79a, BE02, SSSSS10] showed that the stability of a learning algorithm with respect to a specific loss function could ensure strong generalization guarantees with respect to that loss. A more recent literature has focused on stability notions that are not tied to a specific loss, and which ideally are robust under post-processing and adaptive composition. This includes understanding the generalization guarantees of differential privacy [DFH$^+$15a, DFH$^+$15b, BNS$^+$21, RRST16, RRS$^+$20, ZH19, JLN$^+$20] and other constraints on the information-theoretic relationship between the input and output of a learning algorithm [RZ16, BMN$^+$18, XR17, RRT$^+$16, LS19, SZ20]. A related line of work [CLN$^+$16, BF16, NSS$^+$18] considers more "semantic" notions of stability, defining it in terms of the difficulty of inferring properties specific to the sample rather than of the underlying distribution. Perfect generalization, one of the main definitions we study in this work, was introduced by [CLN$^+$16] and is a special case of *typical stability* that was introduced in independent work of Bassily and Freund [BF16].

Independent of this work, [KKMV23] study similar relationships between notions of stability. They focus on the PAC-learning setting, where they show a statistical equivalence between differential privacy, replicability, and a notion called "TV-indistinguishability" which can be thought of as a special case of perfect generalization with $\varepsilon = 0$. To clarify the differences between our work and [KKMV23], first recall how we obtain replicability from differential privacy:

- First, we exploit existing connections between privacy and bounded max-information from [RRST16] to obtain an algorithm with bounded max-information from a differentially private one.

- We prove that bounded max-information implies perfect generalization.

- We then show that we can obtain a replicable algorithm from a perfectly generalizing one by applying correlated sampling to its distribution over outputs. The relevant output distribution is induced by fixing an input sample of the perfectly generalizing algorithm and redrawing its internal randomness.

Recall that the correlated sampling procedure may not be efficient, and that we assume the output domain of the differentially private algorithm is finite.

The work of [KKMV23] follows a different approach. First, they start from a differentially private PAC learner, rather than a differentially private algorithm for a general statistical task, and factor through TV-indistinguishability and Littlestone dimension. More specifically:

- They first observe a similar equivalence of TV-indistinguishability and replicability for general statistical tasks.

- They then show that a private PAC learner implies the existence of a TV-indistinguishable learner, leveraging results from [ALMM19] showing that private PAC learning implies finite Littlestone dimension, and results from [GKM21, GGKM21] showing that finite Littestone dimension implies list global-stability.

Our approach gives us a constructive procedure for converting a private algorithm for a general statistical task into a replicable algorithm, so long as the private algorithm has finite range. Our transformations induce a modest sample complexity increase, resulting in a replicable algorithm with sample complexity n^2, given a private learner with sample complexity n. By contrast, the results of [KKMV23], while non-constructive, apply to countably infinite domains (and therefore to some uncountably infinite ranges). However, their results go through Littlestone dimension, which may be an exponential tower in n, and so they obtain sample complexity bounds which are an exponential tower in n as well.

5.1.4 Open Problems

We highlight several directions and open problems for future work.

1. Is a transformation from (one-way) perfectly generalizing algorithms to replicable algorithms possible for infinite output spaces in general? While correlated sampling introduces no sample complexity overhead in terms of the output space, it is only known to be possible when the output space is finite or the class of distributions to be sampled from is structured. (E.g., the distributions in the class all have uniformly bounded Radon-Nikodym derivative with respect to some fixed base measure).[3] In independent work, [KKMV23] make progress towards this goal by giving a transformation from TV-indistinguishability to replicability when there are only countably many options for the TV-indistinguishable algorithm $\{A(S)\}_{S \in X^n}$. It follows

[3]Formally, such a case would fall into a restricted notion of correlated sampling over a subset of distributions, similar to the multiple coupling of [AS19].

from Lemma 5.3.8 that $(\beta, \varepsilon, \delta)$-one-way perfect generalization implies $(4\varepsilon + 2\delta + 2\beta)$-TV indistinguishability, and so the result of [KKMV23] gives the following corollary.

Corollary 5.1.4. *Fix $n \in \mathbb{N}$, $\beta, \varepsilon, \delta \in (0,1]$. Let $\mathscr{X}$ be a countable domain and $A : \mathscr{X}^n \to \mathscr{Y}$ be a $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing algorithm for a statistical task. Then there exists an algorithm $A' : \mathscr{X}^n \to \mathscr{Y}$ that is $\left(\frac{2\rho}{1+\rho}\right)$-replicable for $\rho = 4\varepsilon + 2\delta + 2\beta$, and for all $S \in \mathscr{X}^n$, $A(S) = A'(S)$.*

Whether a transformation exists for general measure spaces remains open.[4] In Section 5.6.3 we discuss the *list heavy-hitters* problem that may be a candidate for separating perfect generalization from replicability over infinite output spaces.

2. What are the minimal cryptographic assumptions under which a computational separation between replicability and differential privacy exists? Our results in Section 5.4 show that one-way functions are necessary, while public-key assumptions are sufficient.

3. [ILPS22, Lemma A.7] showed that replicable algorithms compose adaptively. That is, a sequence of k adaptively chosen ρ-replicable algorithms yields a transcript that is $O(k\rho)$-replicable. One way to interpret this result is as follows: Given a sequence of k analyses that are each (0.01)-replicable using a sample of size n, one can amplify their individual replicability parameters to $O(1/k)$ at the expense of increasing their sample complexity to $O(k^2 n)$. This yields a (0.01)-replicable algorithm for performing all k analyses at a sample cost of $O(k^2 n)$.

   Our conversions between replicability and differential privacy yield a different tradeoff, at least for simulating non-adaptive composition. Given k analyses that are each (0.01)-replicable using a sample of size n, one can convert them to $\tilde{O}(1/\sqrt{k})$-differentially private

[4]We note that in the PAC-setting one can resolve this issue via factoring through Littlestone Dimension and [ILPS22]'s heavy-hitters, but this results in tower sample complexity.

algorithms each using a sample of size $\tilde{O}(\sqrt{k}n)$. "Advanced" composition of differential privacy [DRV10] yields an $(0.01, \delta)$-differentially private algorithm using $\tilde{O}(\sqrt{k}n)$ samples, which can then be turned back into a (0.01)-replicable algorithm using $\tilde{O}(kn^2)$ samples.

What is the optimal sample cost for conducting, or at least statistically simulating, the (adaptive) composition of k replicable algorithms? Is it possible to do so at a cost of $O(kn)$ samples?

4. In Section 5.5.3, we give a direct replicable algorithm for the task of realizable PAC learning of finite classes with sample cost inverse linear in the accuracy parameter α. (As opposed to inverse quadratic, which is what applying the reduction from replicability to approximate DP gives – see Theorem 5.6.13 and the following discussion.) Are there other natural problems for which there are (perhaps more dramatic) separations between what's achievable via directly constructing a replicable algorithm for a task, and what's achievable using our reduction to approximate DP? For example, can discrete distributions over $[k]$ be replicably estimated using $O(k)$ samples (as opposed to quadratic in k, which is what is obtained through our reduction)? Can the mean of a d-variate Gaussian with unknown covariance be estimated directly using $O(d)$ samples (as opposed to quadratic in d, which is what is obtained through our reduction)? Even more ambitiously, is it possible to characterize the types of problems for which our reduction from replicability to approximate DP gives tight bounds?

5. To what extent is replicability preserved under distributional shift? In Appendix 5.9, we give a simple argument showing that a ρ-replicable algorithm is $\rho(1-\delta)^{2m}$-replicable across two close distributions. Are there tighter replicability and non-replicability bounds for specific families of distributions, problems, and algorithms under distributional shifts?

5.2 Preliminaries

We start by formally defining a statistical task.

Definition 5.2.1. *A* statistical task *with data domain $\mathscr{X}$ and output space $\mathscr{Y}$ is a set of pairs $\mathscr{T} = \{(D, G_D)\}$, where D is a distribution over $\mathscr{X}$ and $G_D \subseteq \mathscr{Y}$ is a "good" set of outputs for distribution D. A randomized algorithm $\mathscr{A}$ solves statistical task $\mathscr{T}$ using m samples and with failure probability β if for every $(D, G_D) \in \mathscr{T}$,*

$$\mathbf{Pr}_{S \sim D^m, \mathscr{A}}[\mathscr{A}(S) \in G_D] \geq 1 - \beta.$$

5.2.1 Notions of Distributional Closeness

We recall the definition of total variation distance, that will be crucial in this work.

Definition 5.2.2 (Total Variation Distance)**.** *Let P and Q be probability distributions over some domain S. Then*

$$d_{\mathrm{TV}}(P, Q) := \sup_{E \subseteq S} |\mathbf{Pr}_P[E] - \mathbf{Pr}_Q[E]|.$$

We also define the notion of (ε, δ)-indistinguishability, the notion of closeness that is used in differential privacy.

Definition 5.2.3 ((ε, δ)-indistinguishability)**.** *Let P and Q be probability distributions over some domain $\mathscr{Y}$. Then, we say that P is (ε, δ)-indistinguishable from Q (denoted as $P \approx_{\varepsilon,\delta} Q$) if for all $O \subseteq \mathscr{Y}$,*

$$e^{-\varepsilon}[\mathbf{Pr}_P[O] - \delta] \leq \mathbf{Pr}_Q[O] \leq e^{\varepsilon}\mathbf{Pr}_P[O] + \delta.$$

We will frequently talk about random variables being (ε, δ)-indistinguishable, which means that their distributions are (ε, δ)-indistinguishable.

5.2.2 Differential Privacy

We say that two datasets $S, S' \in \mathscr{X}^n$ are neighboring if they differ for the data of one individual, i.e., their Hamming distance is one. Differential privacy is formulated as a notion of indistinguishability between the results of an algorithm when run on neighboring datasets.

Definition 5.2.4 (Differential Privacy [DMNS16])**.** *A randomized algorithm $\mathscr{A} : \mathscr{X}^n \to \mathscr{Y}$ is said to be (ε, δ)-*differentially private *if for every pair of neighboring datasets $S, S' \in \mathscr{X}^n$, we have that for all subsets $O \subseteq \mathscr{Y}$,*

$$\mathbf{Pr}[\mathscr{A}(S) \in O] \leq e^{\varepsilon} \cdot \mathbf{Pr}[\mathscr{A}(S') \in O] + \delta.$$

That is, we have $\mathscr{A}(S) \approx_{\varepsilon,\delta} \mathscr{A}(S')$ for all neighboring S, S'.

One important property of differential privacy is that it is closed under post-processing by arbitrary functions.

Lemma 5.2.5 (Post-Processing [DMNS16])**.** *If $\mathscr{A} : \mathscr{X}^n \to \mathscr{Y}$ is (ε, δ)-differentially private, and $\mathscr{B} : \mathscr{Y} \to \mathscr{Z}$ is any randomized function, then the algorithm $\mathscr{B} \circ \mathscr{A}$ is (ε, δ)-differentially private.*

Differential privacy can be achieved by adding some carefully chosen noise to a function and calibrating the noise to the sensitivity of the function: a measure of how different can the results of the function be when run on adjacent datasets.

Definition 5.2.6 (ℓ_1-Sensitivity)**.** *Let $f : \mathscr{X}^n \to \mathbb{R}^d$ be a function. Its ℓ_1-sensitivity is*

$$\Delta_f = \max_{\substack{S, S' \in \mathscr{X}^n \\ S, S' neighbors}} \| f(S) - f(S') \|_1.$$

There are many techniques that can be used to design differentially private algorithms. One important technique that we will use in some of our applications is the exponential mechanism.

Lemma 5.2.7 (Exponential Mechanism [MT07]). *Let L be a set of outputs and $g : L \times \mathscr{X}^n \to \mathbb{R}$ be a function that measures the quality of each output on a dataset. Assume that for every $m \in L$, the function $g(m,.)$ has ℓ_1-sensitivity at most Δ. Then, for all $\varepsilon > 0$, there exists an $(\varepsilon,0)$-DP mechanism that, on input $S \in \mathscr{X}^n$, outputs an element $m \in L$ such that, for all $a > 0$, we have*

$$\mathbf{Pr}\left[\max_{i \in [L]} g(i,S) - g(m,S) \geq 2\Delta \frac{\ln|L| + a}{\varepsilon}\right] \leq e^{-a}.$$

Standard (ε,δ)-differential privacy automatically protects the privacy of groups of individuals.

Lemma 5.2.8 (Group Privacy [DMNS16]). *Let $k \in \mathbb{N}^+$ and let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an (ε,δ)-DP algorithm. Then for all datasets $S, S' \in \mathscr{X}^m$ such that $\|S - S'\|_0 \leq k$,*

$$\mathscr{A}(S) \approx_{k\varepsilon, \delta \frac{e^{k\varepsilon}-1}{e^{\varepsilon}-1}} \mathscr{A}(S').$$

Another property of differential privacy that we will use in many of our algorithms is privacy amplification by subsampling. This says that we can have a stronger privacy protection when we run a differentially private algorithm on a subsample of a dataset.

Lemma 5.2.9 (Secrecy of the sample, [KLN+11, BBG18]). *Let $A : \mathscr{X}^n \to \mathscr{Y}$ be an (ε,δ)-differentially private algorithm. Consider the algorithm $A' : \mathscr{X}^m \to \mathscr{Y}$ that, given a dataset of size m, randomly samples n items without replacement and runs A on the resulting subsample. Then A' is (ε',δ')-differentially private for*

$$\varepsilon' = \frac{n}{m}(e^{\varepsilon} - 1), \qquad \delta' = \frac{n}{m} \cdot \delta.$$

5.2.3 Replicability

Replicability is a strong stability property for randomized algorithms, requiring that the algorithm produce the exact same output with high probability when invoked on two i.i.d. samples from the same distribution, so long as the internal randomness is held fixed. We recall the definition of replicability given in [ILPS22].

Definition 5.2.10 ([ILPS22])**.** *Let D be a distribution over domain $\mathscr{X}$. Let $\mathscr{A}$ be a randomized algorithm that takes as input samples from D. We say that $\mathscr{A}$ is ρ-reproducible if*

$$\mathbf{Pr}_{S,S',r}[\mathscr{A}(S;r) = \mathscr{A}(S';r)] \geq 1-\rho,$$

where S, S' are sets of samples drawn i.i.d. from D and r represents the internal randomness of $\mathscr{A}$.

We will sometimes use the alternative 2-parameter definition of replicability defined in [ILPS22]. Here we assume that the auxiliary inputs described in their original definition are empty.[5]

Definition 5.2.11 ([ILPS22])**.** *Let $A(S;r)$ be an algorithm operating on a sample set $S \in \mathscr{X}^n$ and internal coins r. We say that coin tosses r are η-good for A on distribution D if there exists a "canonical output" z_r such that $\mathbf{Pr}_{S\sim D^n}[A(S;r) = z_r] \geq 1-\eta$. We say that A is (η, ν)-replicable if, for every distribution D, with probability at least $1-\nu$, the coin tosses r are η-good on distribution D.*

[ILPS22] observed that the two parameter and the original single parameter definition (Definition 5.1.1) are essentially equivalent:

Claim 5.2.12 ([ILPS22])**.** *For every $0 \leq \rho \leq v \leq 1$,*

1. *Every ρ-replicable algorithm is also $(\rho/v, v)$-replicable.*

[5]See Section 2.5 for an explanation of our renaming of this definition to "replicable."

2. *Every (ρ, ν)-replicable algorithm is also $\rho + 2\nu$-replicable.*

It was proved in [ILPS22] that we can amplify the replicability parameter at an inverse quadratic cost in the desired replicability parameter. We state a version of this theorem with slightly different constants.

Lemma 5.2.13 (Amplification of Replicability, Theorem A.3, [ILPS22])**.** *Let $0 < \eta, \nu, \beta < \frac{1}{2}$ and $m > 0$. Let $\mathscr{A}$ be an (η, ν)-replicable algorithm for distribution D with sample complexity m and failure probability β. If $\rho > 0$, and $\nu + \rho < 0.25$, there exists a ρ-replicable algorithm $\mathscr{A}'$ for D with sample complexity $m' = \tilde{O}(m(\log 1/\beta)^3/\rho^2(1/2 - \eta)^2)$ and failure probability at most $4(\beta + \rho)$.*

5.2.4 PAC-Learning

We start by defining PAC Learning, which is a canonical definition of supervised learning proposed by Valiant [Val84] and Vapnik and Chervonenkis [VC74]. We first consider the realizable setting.

Definition 5.2.14 (Realizable PAC learning, [Val84, VC74])**.** *A learning problem is defined by a hypothesis class H. For any distribution D over the input space $\mathscr{X}$, consider m independent draws $x_1, x_2, \ldots, x_m$ from distribution P. For $f \in H$, a labeled sample of size m is the set $\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_m, f(x_m))\}$. We say an algorithm A is an (α, β)-accurate PAC learner for the hypothesis class H if for all functions $f \in H$ and for all distributions D over the input space, A on being given a labeled sample of size m drawn from D and labeled by f, outputs a hypothesis h such that with probability greater than or equal to $1 - \beta$ over the randomness of the sample and the algorithm,*

$$\mathbf{Pr}_{x \in D}[h(x) \neq f(x)] \leq \alpha.$$

We also consider a variant called agnostic PAC learning, where the labels of the input dataset can be noisy.

Definition 5.2.15 (Agnostic PAC learning, [Hau92, VC74]). *A learning problem is defined by a hypothesis class H. We say an algorithm $\mathscr{A}$ is an (α,β)-accurate PAC learner for the hypothesis class H if for all distributions D over input, output pairs, $\mathscr{A}$ on being given a sample of size m drawn i.i.d. from D outputs a hypothesis h such that with probability greater than or equal to $1-\beta$ over the randomness of the sample and the algorithm,*

$$\mathrm{err}_D(h) \leq \inf_{f\in H} \mathrm{err}_D(f) + \alpha.$$

where $\mathrm{err}_D(h) = \mathbf{Pr}_{(x,y)\in D}[h(x) \neq y]$. *In this context, we will sometimes refer to PAC-learning as the* ***realizable setting****.*

We will need uniform convergence for several of our results.

Theorem 5.2.16 (Uniform Convergence, e.g., [BEHW89]). *Let H be a binary class of functions with domain $\mathscr{X}$. Let its VC dimension be d. Then, for any distribution D over $\mathscr{X}$, for all $m > 0$,*

$$\mathbf{Pr}_{x_1,\ldots,x_m\sim D}\left[\sup_{h_z\in H}\left|\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}[h_z(x_i)=1]-\mathbf{Pr}_{x\sim D}[h_z(x)=1]\right|\geq\gamma\right]\leq 4(2m)^d e^{-\gamma^2 m/8}.$$

5.2.5 Correlated Sampling

In the correlated sampling problem, two (or more) players are given probability distributions $\mathscr{P},\mathscr{Q}$ over the same finite set, and with access to shared randomness. The players (without communicating) want to sample from their respective distributions, while minimizing the probability that their outputs disagree.

More formally, let $Y=\{0,1\}$, and for a set Ω, 2^Ω denotes the set of all functions from Ω to Y, and Δ_Ω denotes the set of all sampleable distributions on Ω. For two distributions $\mathscr{P}$, $\mathscr{Q}$ over Ω, let $d_{TV}(\mathscr{P},\mathscr{Q})$ denote the total variational distance between $\mathscr{P}$ and $\mathscr{Q}$.

Definition 5.2.17. *(Correlated Sampling) A correlated sampling strategy for a finite set Ω with* *error* $\varepsilon : [0,1] \to [0,1]$ *is an algorithm* $CS : \Delta_\Omega \times \mathscr{R}'$ *and a distribution* $\mathscr{R}'$ *on random strings such that:*

- *(Marginal Correctness) For all* $\mathscr{P} \in \Delta_\Omega$ *and* $w \in \Omega$, $\mathbf{Pr}_{r' \sim \mathscr{R}'}[CS(\mathscr{P}, r') = w] = \mathscr{P}(w)$.
- *(Error Guarantee) For all* $\mathscr{P}, \mathscr{Q} \in \Delta_\Omega$, $\mathbf{Pr}_{r' \sim \mathscr{R}'}[CS(\mathscr{P}, r') \neq CS(\mathscr{Q}, r')] \leq \varepsilon(d_{TV}(\mathscr{P}, \mathscr{Q}))$

Several independent papers [KT02, Hol09, Bro97] give correlated sampling strategies over finite sets, with $\varepsilon(\delta) = \frac{2\delta}{1+\delta}$. These algorithms use consistent sampling strategies, which are also used in several other contexts such as sketching algorithms, approximation algorithms, and parallel repetition theorems. However, despite the many uses of correlated sampling, some basic questions remain open. Notably, it is not known whether or not correlated sampling is possible for infinite domains, or whether correlated sampling can be made *efficient*. (All known algorithms run in exponential-time in the worst-case.) For a nice discussion as well as new results on the optimality of these constructions see [BGH$^+$16]. In Section 5.3.3, we give another application of correlated sampling, showing how any perfectly generalizing algorithm can be transformed into a replicable one, via correlated sampling. As noted in the introduction, this is the only implication that does not preserve computational efficiency, due to the inefficiency of the correlated sampling strategy.

In Section 5.4.1, we prove that this is inherent: under standard cryptographic assumptions any such transformation is intractable, and therefore under the same cryptographic assumption, correlated sampling is intractable. Moreover, we show in Section 5.4.2 that some assumption is necessary: if one-way functions do not exist (that is, all poly-time computable functions can be efficiently inverted), we show that this implies a polynomial-time algorithm for correlated sampling.

5.2.6 Terminology: "Reproducibility" and "Replicability"

[ILPS22] introduced a mathematical definition referring to a particular stability notion of a randomized learning algorithm which they originally called "reproducibility" (reproducible algorithms). In this paper, we use the term "replicability" (replicable algorithms) to refer to the same mathematical definition.

This terminology choice is more in line with the most current Association for Computing Machinery (ACM) guidance regarding artifact review and badging [Ass20], version 1.1, updated on August 24, 2020. This update changed the ACM's definitons of the terms "reproducible" and "replicable" to be more agreeable with the terminology currently used by the National Academies of Sciences, Engineering and Medicine (see Chapter 3: Understanding Reproducibility and Replicability, page 46, in [Nat19]).

According to both the ACM's and National Academies' current definitions, "reproducibility" refers to the ability of a second experimental group to obtain similar results using the *same* input data. Meanwhile, "replicability" refers to the ability of a second experimental group to obtain similar results using input data and methods that may be different than those used by the original experimental group.

The mathematical definition introduced in [ILPS22] is a guarantee that, with high probability, two executions of the same algorithm with the same randomness and different sample sets will produce the same answer. Since this guarantee is over different sample sets, the mathematical definition does not fit the "same input data" condition in the above definitions of reproducibility. Instead, the mathematical definition is a specific type of replicability — if the second experimental group runs the same algorithm with the same random string (but on a new sample), the two groups' results are guaranteed to be identical with high probability.

5.3 Equating Stability: Differential Privacy, Perfect Generalization, and Replicability

5.3.1 Replicability implies Approximate-DP

In [GKM21], the authors show a sample-efficient reduction from differentially private PAC learning to replicable PAC learning. In this section, we show their technique generalizes to arbitrary statistical problems.[6]

Recall the definition of a statistical task in Definition 5.2.1. We will show that any statistical task with a "good" replicable learner can also be solved privately, without substantial blowup in runtime or sample complexity.

Theorem 5.3.1 (Replicability $\to$ DP). *Let $\mathscr{T}$ be a statistical problem. For all $\beta > 0$, if there is a 0.01-replicable algorithm solving $\mathscr{T}$ using n_R samples and with failure probability β, then for any $0 < \varepsilon, \delta \leq 1$ there is an (ε, δ)-DP algorithm for $\mathscr{T}$ using n_{DP} samples with failure probability $O\left(\beta \log \frac{1}{\beta}\right)$, where*

$$n_{DP}(\varepsilon, \delta, \beta) \leq n_R \cdot O\left(\frac{\log \delta^{-1} \log \beta^{-1}}{\varepsilon} + \log^2 \beta^{-1}\right)$$

This conversion relies on the following private algorithm for selecting an approximate mode.

Theorem 5.3.2 (DP Selection [KKMN09, BNS16c, BDRS18]). *There exists some $c > 0$ such that for every $\varepsilon, \delta > 0$ and $m \in \mathbb{N}$, there is an (ε, δ)-DP algorithm that on input $S \in \mathscr{X}^m$, outputs with probability 1 an element $x \in X$ that occurs in S at most $\frac{c \log \delta^{-1}}{\varepsilon}$ fewer times than the true mode of S. Moreover, the algorithm runs in poly$(m, \log(|\mathscr{X}|))$ time.*

The idea, as in [GKM21], is to use replicablity to construct a sample over the output space where some correct solution appears many times. In particular, given a replicable algorithm $\mathscr{A}$ on n

[6]The argument remains similar to [GKM21], but requires a few changes to avoid union bounding over failure probability which can be costly in settings beyond PAC learning.

samples, consider the following simple procedure adapted from [GKM21]: partition a larger data set, run $\mathscr{A}$ on each part, and privately output a commonly repeated element.

```
**Algorithm 24.** DP-to-Replicability Reduction
**Result:** Privately ouputs solution to $(\mathscr{X}, R)$
**Input:** Statistical Problem $(\mathscr{X}, R)$. Distribution $D$ over $\mathscr{X}$, Replicable algorithm $\mathscr{A}$ on $n$ samples
  **Parameters:**
    • Privacy and Correctness $\beta, \varepsilon, \delta > 0$
    • Seed Number $k_1 = O(\log \beta^{-1})$
    • Partition Number $k_2 = O\left(\frac{\log \delta^{-1}}{\varepsilon} + \log \beta^{-1}\right) \cdot k_1$
**Algorithm:**
  1. For every $j \in [k_1]$ and $i \in [\frac{k_2}{k_1}]$ sample $S_{i,j} \sim \mathscr{X}^n$
  2. Sample $k_1$ random strings $\{r_j\}$.
  3. Let $y_{i,j} = \mathscr{A}(S_{i,j}; r_j)$.
  4. Run $(\varepsilon, \delta)$-DP Selection on $\{y_{i,j}\}$ and denote the output by $y^*$.
**return** $y^*$
```

Recalling the two-parameter definition of replicability (5.2.11 and 5.2.12), since our subroutine is 0.01-replicable and β-correct, it is also $(0.1, 0.1)$-replicable and β correct. Therefore, the proof of Theorem 5.3.1 is an immediate consequence of the following proposition.

Proposition 5.3.3. *For all sufficiently small $\beta, \varepsilon, \delta > 0$, if $\mathscr{A}$ is $(0.1, 0.1)$-replicable and has failure probability β, then Algorithm 24 is (ε, δ)-private and has failure probability $O(\beta \log 1/\beta)$.*

Proof. Privacy is essentially immediate from DP Selection. This follows because the input to selection based on a neighboring input database T' differs in at most one of the $\{y_i\}$ (as we've partitioned the sample disjointly). Thus the reduction automatically inherits (ε, δ)-privacy from DP Selection. The main interest in the reduction, then, is maintaining correctness which we argue next. The proof breaks into two parts:

1. With probability $1-\beta/2$, some $y^* \in \{y_i\}$ appears at least $t_1 := 2c\left(\frac{\log\delta^{-1}}{\varepsilon} + \log\beta^{-1}\right)$ times.
2. With probability $1-\beta\log 1/\beta$, *any* element appearing at least $t_2 := c\left(\frac{\log\delta^{-1}}{\varepsilon} + \log\beta^{-1}\right)$ times is correct.

The result then follows from observing that by a union bound both conditions hold with probability at least $1-O(\beta\log 1/\beta)$, and conditioned on this fact DP-Selection always outputs an element that occurs at least $t_1 - c\frac{\log\delta^{-1}}{\varepsilon} \geq t_2$ times (which is then guaranteed to be correct).

It remains to prove the claims. For the first, note that since $\mathscr{A}$ is $(0.1, 0.1)$-replicable, there exists some .1-good random string $r^* \in \{r_j\}$ with probability at least $1-\beta/4$. By a Chernoff bound, the probability that the canonical element corresponding to r^* appears fewer than t_1 times is at most $\beta/4$, which proves the claim (for a large enough choice of k_2).

Finally, we argue any common element is correct. Since $\mathscr{A}$ is a β-correct algorithm, in expectation, the number of incorrect outputs is βk_2. Hence, by Markov's inequality, the probability that there are more than $b = O(\frac{k_2}{\log\beta^{-1}})$ incorrect outputs is at most $\beta\log 1/\beta$. For small enough choice of constant in the correctness of our replicable algorithm,[7] we can make $b < t_2$, so no element appearing at least t_2 times can be incorrect as desired.

$\square$

[7]Note this choice can be taken universally with respect to all parameters and the statistical problem itself.

5.3.2 Approximate-DP Implies One-Way Perfect Generalization

Preliminaries about Perfect Generalization

Perfect generalization (Definition 5.1.2) is a notion of stability that captures the idea (like differential privacy) that an algorithm $\mathscr{A}$ does not depend on its input samples too much.

We also consider the following "two-sample" version of this definition. This is frequently easier to work with for symmetry reasons.

Definition 5.3.4. *An algorithm $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ is said to be $(\beta, \varepsilon, \delta)$-sample perfectly generalizing, if for every distribution D over $\mathscr{X}$, with probability at least $1-\beta$ over the draw of two i.i.d. samples $S_1, S_2 \sim D^m$, $\mathscr{A}(S_1) \approx_{\varepsilon,\delta} \mathscr{A}(S_2)$.*

Cummings et al. [CLN$^+$16] prove the following lemma relating perfect generalization to sample perfect generalization.

Lemma 5.3.5 ([CLN$^+$16])**.** *If algorithm $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ is $(\beta, \varepsilon, \delta)$-perfectly generalizing, then it is also $(\beta, 2\varepsilon, 3\delta)$-sample-perfectly generalizing.*

We can also prove a partial converse to this result; this will be frequently useful since it allows us to prove sample perfect generalization and invoke this result to get perfect generalization.

Lemma 5.3.6. *Fix $\beta, \varepsilon, \delta \in (0,1]$. If algorithm $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ is $(\beta, \varepsilon, \delta)$-sample perfectly generalizing, then it is also $(\sqrt{\beta}, \varepsilon, \delta + \sqrt{\beta})$-perfectly generalizing.*

Proof. Since $\mathscr{A}$ is $(\beta, \varepsilon, \delta)$-sample PG, we have that for any distribution D over $\mathscr{X}$, with probability at least $1-\beta$ over the draw of two i.i.d. datasets $S_1, S_2 \sim D^m$, we have that $\mathscr{A}(S_1) \approx_{\varepsilon,\delta} \mathscr{A}(S_2)$. This guarantees by the reverse Markov inequality that for $d = 1-\sqrt{\beta}$,

$$\begin{aligned}
&\mathbb{E}_{S_1}[\mathbf{Pr}_{S_2}(\mathscr{A}(S_1) \approx_{\varepsilon,\delta} \mathscr{A}(S_2))] \geq 1-\beta \implies \\
&\mathbf{Pr}_{S_1}[\mathbf{Pr}_{S_2}(\mathscr{A}(S_1) \approx_{\varepsilon,\delta} \mathscr{A}(S_2)) \geq d] \geq \frac{1-\beta-d}{1-d} \implies \\
&\mathbf{Pr}_{S_1}[\mathbf{Pr}_{S_2}(\mathscr{A}(S_1) \approx_{\varepsilon,\delta} \mathscr{A}(S_2)) \geq 1-\sqrt{\beta}] \geq 1-\sqrt{\beta}.
\end{aligned}$$

Now, set Sim_D to be the distribution of $\mathscr{A}(S_2)$ where the randomness of the distribution is taken over both the randomness of the algorithm and the dataset.

For any fixed dataset S_1, let G_{S_1} be the set of datasets S_2 such that $\mathscr{A}(S_1) \approx_{\varepsilon,\delta} \mathscr{A}(S_2)$. Then, we get that with probability at least $1-\sqrt{\beta}$ over the draw of S_1, for any $O \subseteq Y$,

$$\begin{aligned}
\mathbf{Pr}_{Sim_D}[O] &= \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O] \\
&= \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O \mid S_2 \in G_{S_1}]\mathbf{Pr}[S_2 \in G_{S_1}] \\
&\quad + \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O \mid S_2 \notin G_{S_1}]\mathbf{Pr}[S_2 \notin G_{S_1}] \\
&\leq \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O \mid S_2 \in G_{S_1}] + \sqrt{\beta} \\
&\leq e^{\varepsilon}\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S_1) \in O] + \delta + \sqrt{\beta}.
\end{aligned}$$

Similarly, we can also argue that

$$\begin{aligned}
\mathbf{Pr}_{Sim_D}[O] &= \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O] \\
&= \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O \mid S_2 \in G_{S_1}]\mathbf{Pr}[S_2 \in G_{S_1}] \\
&\quad + \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2) \in O \mid S_2 \notin G_{S_1}]\mathbf{Pr}[S_2 \notin G_{S_1}] \\
&\geq \mathbf{Pr}_{S_2,\mathscr{A}}[\mathscr{A}(S_2)] \in O \mid S_2 \in G_{S_1})(1-\sqrt{\beta}) \\
&\geq e^{-\varepsilon}\left(\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S_1) \in O] - \delta\right)(1-\sqrt{\beta}) \\
&\geq e^{-\varepsilon}\left(\mathbf{Pr}_{\mathscr{A}}(\mathscr{A}(S_1) \in O] - \delta - \sqrt{\beta}\right).
\end{aligned}$$

Hence, with probability at least $1-\sqrt{\beta}$ over the draw of S_1,

$$Sim_D \approx_{\varepsilon,\delta} \mathscr{A}(S_1).$$

$\square$

We also define a notion of "one-sided" perfect generalization which only requires the probability of events under $A(S)$ not to increase too much relative to their probability under the simulator distribution Sim_D. This new definition will be crucial to show the equivalence between replicability and perfect generalization, as well as for some of our applications.

Definition 5.3.7 (One-way perfect generalization)**.** *An algorithm $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ is said to be $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing if for every distribution D over $\mathscr{X}$, there exists a distribution Sim_D such that with probability at least $1-\beta$ over the draw of an i.i.d. sample $S \sim D^m$, for every output set $O \subseteq \mathscr{Y}$ we have that*

$$\mathbf{Pr}[\mathscr{A}(S) \in O] \le e^{\varepsilon} \mathbf{Pr}_{Sim_D}[O] + \delta.$$

Next, we prove a simple lemma relating the parameters achievable with perfect generalization.

Lemma 5.3.8. *Fix $m \in \mathbb{N}$, $\beta, \varepsilon, \delta \in (0,1]$. Let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be a $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing algorithm. Then, $\mathscr{A}$ is also $(\beta, 0, 2\varepsilon + \delta)$-perfectly generalizing.*

Proof. By the definition of one-way perfect generalization, we have that for all distributions D over $\mathscr{X}$, there exists a distribution Sim_D, such that with probability $1-\beta$ over the draw of S, for all $O \subseteq \mathscr{Y}$,

$$\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O] \le e^{\varepsilon} \mathbf{Pr}_{Sim_D}[O] + \delta.$$

Using the fact that for $\varepsilon \le 1$, $e^{\varepsilon} \le 1 + 2\varepsilon$, we get that

$$\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O] \le \mathbf{Pr}_{Sim_D}[O](1 + 2\varepsilon) + \delta,$$

which gives us that

$$\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O] \le \mathbf{Pr}_{Sim_D}[O] + 2\varepsilon + \delta.$$

Now, since this works for any set O, consider applying it to O^c. Then, we get that

$$\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O^c] \leq \mathbf{Pr}_{Sim_D}[O^c] + 2\varepsilon + \delta,$$

which implies (by writing $\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O^c] = 1 - \mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O]$ and likewise for Sim_D and doing some algebraic manipulation) that

$$\mathbf{Pr}_{Sim_D}[O] \leq \mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O] + 2\varepsilon + \delta.$$

Hence, the lemma is proved. $\square$

[CLN$^+$16] also proved a strong relationship between $(\varepsilon, 0)$-differential privacy and perfect generalization. Specifically, they proved the following.

Theorem 5.3.9 ([CLN$^+$16])**.** *If algorithm $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ is $(\varepsilon, 0)$-differentially private, then for all $\beta > 0$, it is $(\beta, \varepsilon\sqrt{2m\ln(2|\mathscr{Y}|/\beta)}, 0)$-perfectly generalizing.*

They left establishing similar relationships between (ε, δ)-differential privacy and perfect generalization as an open question. We resolve this question for finite outcome spaces. Our argument is indirect and involves showing that any approximate differentially private algorithm is *one-way* perfectly generalizing. We will later use this to prove that any approximate differentially private algorithm can be *compiled* into another algorithm that is perfectly generalizing (under the original definition). We do this through the tool of max information, that we discuss next.

Preliminaries about Max Information

The notion of max information was formulated in work on the connection between differential privacy and adaptive data analysis. It quantitatively captures the degree of correlation between two random variables, by comparing the joint distribution of the random variables to the product measure. Intuitively, if the joint distribution and the product measure are "close", then the random variables are not too correlated with each other.

Definition 5.3.10 (Based on [DFH$^+$15a]). *The β-approximate max information between two correlated random variables X and Z, denoted $I_\infty^\beta(X,Z)$, is defined as the minimum (infimum) value k such that for all output sets O, we have that*

$$\mathbf{Pr}_{(a,b)\sim(X,Z)}[(a,b)\in O] \le 2^k \mathbf{Pr}_{(a,b)\sim X\otimes Z}[(a,b)\in O] + \beta \tag{5.2}$$

where $X\otimes Z$ represents the product measure of the 2 random variables.

In this paper, we will be concerned about the degree of correlation between a randomly sampled dataset, and the output of an algorithm run on that dataset. Intuitively, replicability requires that an algorithm's output does not depend too much on the specific input sample it gets, so the max information between these two random variables will be a useful quantity to analyze.

Approximate Differential Privacy to Bounded Max Information

Connections between max information and differential privacy have been previously studied. Rogers, Roth, Smith and Thakkar [RRST16] give a bound on the max information between an approximate DP algorithms' outputs and its inputs (Theorem 3.1 in their paper). In fact, they prove the following general statement that can be seen by examining their proof of Theorem 3.1.

Lemma 5.3.11 ([RRST16]). *Fix $m\in\mathbb{N}$, $\varepsilon\in(0,1/2]$ and $\delta\in[0,\varepsilon/15)$. Let $\mathscr{A}:\mathscr{X}^m\to\mathscr{Y}$ be an (ε,δ)-DP algorithm. Then, for any distribution D over $\mathscr{X}$, if $S\sim D^m$, for all $t>0$, $I_\infty^\beta(S;\mathscr{A}(S))\le m\nu+6t\varepsilon\sqrt{m}$, where $\beta=e^{-t^2/2}+cm\sqrt{\frac{\delta}{\varepsilon}}$, and $\nu=C(\varepsilon^2+\sqrt{\frac{\delta}{\varepsilon}})$ for some sufficiently large constants c,C.*

We instantiate this lemma with parameters that are suitable for our application.

Corollary 5.3.12. *Fix $m \in \mathbb{N}$, sufficiently small $\rho \in (0,1)$. Let $\varepsilon = \frac{\rho}{\sqrt{8m\log(1/\rho)}}$, $\delta \le \frac{\varepsilon\rho^6}{m^2}$. Let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an (ε,δ)-DP algorithm. Then, for any distribution D over $\mathscr{X}$, if $S \sim D^m$, $I_\infty^{\rho^3}(S;\mathscr{A}(S)) \le O(\rho)$.*

Proof. Substituting the value of ε, δ, and setting $t = \sqrt{8\log(1/\rho)}$ in the expression for β in Lemma 5.3.11, we get

$$\beta = e^{-4\log(1/\rho)} + cm\sqrt{\frac{\rho^6}{m^2}} = O(\rho^3).$$

Substituting in the expression for ν in Lemma 5.3.11 gives

$$\nu = C\left(\varepsilon^2 + \sqrt{\frac{\delta}{\varepsilon}}\right) \le C\left(\frac{\rho^2}{m} + \frac{\rho^3}{m}\right) = O\left(\frac{\rho^2}{m}\right).$$

Substituting the values of ν, t, and ε in the upper bound for max information gives

$$I_\infty^{\rho^3}(S;\mathscr{A}(S)) \le m\nu + 6t\varepsilon\sqrt{m} = \rho^2 + 6\rho = O(\rho).$$

□

In general, one can convert an (ε,δ)-differentially private algorithm with $\varepsilon = O(1)$ and $\delta = 1/\text{poly}(n)$ into a bounded max-information algorithm by first amplifying the privacy parameters:

Corollary 5.3.13. *There are constants $c, C > 0$ such that the following holds. Let $\gamma > 0$. Suppose $\mathscr{A} : \mathscr{X}^n \to \mathscr{Y}$ is an (ε,δ)-differentially private algorithm solving a statistical task up to some failure probability β such that $\varepsilon \in (0,1)$ and $\delta \le \min\{\gamma^2/c^2, \rho^2/C^2\}\rho^4/64\varepsilon^3(2C\rho + \sqrt{72\log(2/\gamma)})^2 n^4$. Then there is an algorithm $\mathscr{A}' : \mathscr{X}^m \to \mathscr{Y}$ solving the same statistical task with the same failure probability β, such that*

$$m = 4\varepsilon^2 n^2 \cdot \left(\frac{2C}{\rho} + \frac{\sqrt{72\log(2/\gamma))}}{\rho^2}\right),$$

and for every distribution D over $\mathscr{X}$, if $S \sim D^m$ we have $I_\infty^\gamma(S;\mathscr{A}(S)) \le \rho$. Moreover, the conversion from $\mathscr{A}$ to $\mathscr{A}'$ preserves computational efficiency.

Proof. The algorithm $\mathscr{A}'$ simply samples n items without replacement from S and runs $\mathscr{A}$ on the result. This perfectly preserves correctness with respect to any statistical task. By Lemma 5.2.9, we have that $\mathscr{A}'$ is (ε',δ')-differentially private for

$$\varepsilon' = \frac{2n}{m} \le \frac{1}{\sqrt{m}} \min\left\{\sqrt{\frac{\rho}{2C}}, \frac{\rho}{\sqrt{72\log(2/\gamma)}}\right\}, \qquad \delta' = \frac{n}{m}\cdot\delta.$$

Note that these parameters ensure that

$$\frac{\delta'}{\varepsilon'} = \frac{\delta}{e^\varepsilon - 1} \le \frac{\delta}{\varepsilon} \le \min\left\{\frac{\gamma^2}{4c^2m^2}, \frac{\rho^2}{4C^2m^2}\right\}.$$

Now set $t = \sqrt{2\log(2/\gamma)}$ in the statement of Lemma 5.3.11, which ensures that

$$e^{-t^2/2} + cm\sqrt{\frac{\delta'}{\varepsilon'}} \le \frac{\gamma}{2} + cm\sqrt{\frac{\gamma^2}{4c^2m^2}} \le \gamma.$$

Then the lemma implies that

$$\begin{aligned} I_\infty^\gamma(S;\mathscr{A}(S)) &\le Cm\left((\varepsilon')^2 + \sqrt{\frac{\delta'}{\varepsilon'}}\right) + 6t\varepsilon'\sqrt{m} \\ &\le Cm\left(\frac{\rho}{2Cm} + \frac{\rho}{2Cm}\right) + 6\sqrt{2\log(2/\gamma)}\cdot\frac{\rho}{\sqrt{72\log(2/\gamma)}} \\ &\le \rho. \end{aligned}$$

□

Bounded Max Information to One-Way Perfect Generalization

Next, we prove a key lemma relating bounded max-information to one-way perfect generalization. The approach we follow is similar to that used to derive relationships between pointwise (ε,δ)-indistinguishability and (ε,δ)-indistinguishability in Lemma 3.3 of [KS14].

Lemma 5.3.14. *Fix $m \in \mathbb{N}$, $k > 0$, $\beta \in (0,1)$ and $\hat{\beta} = \sqrt{\frac{\beta}{1-2^{-k}}}$. Let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an algorithm. Then, for every distribution D over $\mathscr{X}$ and $S \sim D^n$, if $I_\infty^\beta(S;\mathscr{A}(S)) \le k$, then $\mathscr{A}$ is $(\hat{\beta}, 2k, \hat{\beta})$-one-way perfectly generalizing.*

Proof. The canonical distribution Sim_D we will consider is the distribution of $\mathscr{A}(S')$, where the randomness is over both $S' \sim D^m$, and the internal randomness of $\mathscr{A}$.

We start by defining a set of 'bad' outputs for each fixed dataset S, i.e. outputs on which the probability mass of $\mathscr{A}(S)$ is substantially larger than that of the canonical distribution Sim_D. Formally, for each dataset S, let

$$B_S = \{y \in \mathscr{Y} : \mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) = y] \ge 2^{2k}\mathbf{Pr}_{S' \sim D^m, \mathscr{A}}[\mathscr{A}(S') = y]\}.$$

Next, we define a set of ordered pairs consisting of datasets and their corresponding 'bad' outputs. Formally, let

$$\Theta = \{(S,y) : S \in \text{supp}(D^m), y \in B_S\}.$$

Our goal will be to prove that with high probability over a draw of a dataset S, $\mathscr{A}(S)$ lands in the bad set B_S with small probability. This can then be used to establish one-way perfect generalization.

With this in mind, consider the expression $\mathbb{E}_S[\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in B_S]] = \mathbb{E}_S[\mathbf{Pr}_{\mathscr{A}}[(S,\mathscr{A}(S)) \in \Theta]]$. By the law of total probability, this is equal to $\mathbf{Pr}_{S \sim D^m, \mathscr{A}}[(S,\mathscr{A}(S)) \in \Theta]$. Using the definition of max-information, we get that

$$\mathbf{Pr}_{S\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S))\in\Theta]\leq 2^k\mathbf{Pr}_{S,S'\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S'))\in\Theta]+\beta. \tag{5.3}$$

Now, analyzing the term $\mathbf{Pr}_{S,S'\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S'))\in\Theta]$, we get

$$\begin{aligned}\mathbf{Pr}_{S,S'\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S'))\in\Theta] &= \sum_{T\in\mathrm{supp}(D^m)}\mathbf{Pr}[S=T]\mathbf{Pr}_{S'\sim D^m,\mathscr{A}}[\mathscr{A}(S')\in B_T]\\ &\leq \sum_{T\in\mathrm{supp}(D^m)}\mathbf{Pr}[S=T]2^{-2k}\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S)\in B_T]\\ &= 2^{-2k}\sum_{T\in\mathrm{supp}(D^m)}\mathbf{Pr}[S=T]\mathbf{Pr}_{\mathscr{A},S\sim D^m}[A(S)\in B_S\mid S=T]\\ &= 2^{-2k}\mathbf{Pr}_{S\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S))\in\Theta],\end{aligned}$$

where the first inequality is by the definition of B_T and the last equality is by the law of total probability. Substituting the above in (5.3), we get that

$$\mathbf{Pr}_{S\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S))\in\Theta]\leq 2^{-k}\mathbf{Pr}_{S\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S)\in\Theta]+\beta.$$

Rearranging, this gives us that

$$\mathbb{E}_S[\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S)\in B_S]]=\mathbf{Pr}_{S\sim D^m,\mathscr{A}}[(S,\mathscr{A}(S))\in\Theta]\leq\frac{\beta}{1-2^{-k}}=\hat{\beta}^2. \tag{5.4}$$

Finally, by Markov's inequality and the above equation, we can write the following.

$$\mathbf{Pr}_S[\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S)\in B_S]>\hat{\beta}]\leq\frac{\mathbb{E}_S[\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S)\in B_S]]}{\hat{\beta}}\leq\hat{\beta}. \tag{5.5}$$

This implies that with probability $1-\hat{\beta}$ over $S\sim D^m$, $\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S)\notin B_S]>1-\hat{\beta}$. Finally, we can write with probability $1-\hat{\beta}$ over $S \sim D^m$, for every $O \subseteq \mathscr{Y}$,

$$\begin{aligned}
\mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O] &= \mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O \mid \mathscr{A}(S) \in B_S]\mathbf{Pr}[\mathscr{A}(S) \in B_S] \\
&\quad + \mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O, \mathscr{A}(S) \notin B_S] \\
&\leq 1 \cdot \hat{\beta} + \mathbf{Pr}_{\mathscr{A}}[\mathscr{A}(S) \in O, \mathscr{A}(S) \notin B_S] \\
&\leq \hat{\beta} + 2^{2k}\mathbf{Pr}_{S' \sim D^m, \mathscr{A}}[\mathscr{A}(S') \in O, \mathscr{A}(S') \notin B_{S'}] \\
&\leq \hat{\beta} + 2^{2k}\mathbf{Pr}_{S' \sim D^m, \mathscr{A}}[\mathscr{A}(S') \in O].
\end{aligned}$$

This completes the proof. □

We note that the above proof sets the failure probability due to data sampling and that due to bad coins of the algorithm to be the same. Other tradeoffs between these can be obtained by using Markov's inequality with different parameters in Equation 5.5. We chose them to be equal to each other for simplicity of presentation and because that is the setting of interest in our applications.

Observe that combining Lemma 5.3.11 and Lemma 5.3.14 above gives the following connection between differential privacy and one-way perfect generalization.

Corollary 5.3.15. *Fix $m \in \mathbb{N}$, $\varepsilon \in (0, 1/2]$ and $\delta \in [0, \varepsilon/15)$. Let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an (ε, δ)-DP algorithm. Then, for sufficiently large constants c, C, for all $t > 0$, $\mathscr{A}$ is $(\delta', \varepsilon', \delta')$-one-way perfectly generalizing, where $\varepsilon' = Cm(\varepsilon^2 + \sqrt{\frac{\delta}{\varepsilon}}) + 6t\varepsilon\sqrt{m}$, and $\delta' = \frac{\beta}{1-2^{-O(\varepsilon')}}$, where $\beta = e^{-t^2/2} + cm\sqrt{\frac{\delta}{\varepsilon}}$.*

As an example of the kind of result this can give, we show what we'd get if we instantiated it with our parameters of interest (as in Corollary 5.3.12).

Corollary 5.3.16. *Fix $m \in \mathbb{N}$, sufficiently small $\rho \in (0, 1]$. Let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an (ε, δ)-DP algorithm, where $\varepsilon = \frac{\rho}{\sqrt{8m\log(1/\rho)}}$, $\delta \leq \frac{\varepsilon\rho^6}{m^2}$. Then, $\mathscr{A}$ is $(O(\rho), O(\rho), O(\rho))$-one-way perfectly generalizing.*

Proof. From Corollary 5.3.12, we get that $I_\infty^{\rho^3}(S; \mathscr{A}(S)) \leq O(\rho)$. Substituting $k = \rho$ and $\beta = \rho^3$

in Lemma 5.3.14, we get that $\hat{\beta} = \sqrt{\frac{c\rho^3}{1-2^{-O(\rho)}}} \leq O(\sqrt{\rho^2}) = O(\rho)$ (where the first inequality is since $\frac{1}{1-2^{-O(\rho)}} = \frac{2^{O(\rho)}}{2^{O(\rho)}-1} \leq \frac{2}{2^{O(\rho)}-1} \leq \frac{C}{\rho}$ for some constant C, since $2^{c\rho} = e^{c\rho \ln 2}$ and $e^x \geq 1+x$ for all real x). This gives us from Lemma 5.3.14 that $\mathscr{A}$ is $(O(\rho), O(\rho), O(\rho))$-one-way perfectly generalizing. □

5.3.3 Perfect Generalization Implies Replicability

In this section will show that the class of one-way perfectly generalizing algorithms, which includes the special case of (two-way) perfectly generalizing algorithms, can be transformed to replicable algorithms.

Let $CS(Q, \mathscr{Y}, r')$ represent a correlated sampling procedure over domain $\mathscr{Y}$ sampling from a distribution Q over $\mathscr{Y}$ with public randomness r'. (See Section 5.2.5 for background on correlated sampling). We now describe our transformation.

Algorithm 25. Transformation from one-way perfectly generalizing algorithm to replicable algorithm

```
    Input: dataset $S = (s_1, \ldots, s_n)$, description of one-way perfectly generalizing algorithm $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$
    Output: $i \in \mathscr{U}$
1: Let $Q_S$ represent the distribution of $\mathscr{A}(S)$.
2: Output $CS(Q_S, \mathscr{Y}, r')$ where $r'$ is the random string drawn in the correlated sampling algorithm.
```

The key idea is that correlated sampling converts total variation distance into collision probability, which is the notion that is used in the definition of replicability.

We now use this to prove the main theorem of this section.

Theorem 5.3.17. *Fix $m \in \mathbb{N}$ and $\beta, \varepsilon, \delta \in (0,1)$. Let $A : \mathscr{X}^m \to \mathscr{Y}$ be a $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing algorithm with finite output space. Then, for any distribution D over $\mathscr{X}$, if $S \sim D^m$, Algorithm 25 when run on dataset S and with access to $\mathscr{A}$ is $4(\beta + 2\varepsilon + \delta)$-replicable.*

Proof. By Lemma 5.3.8, we have that $\mathscr{A}$ is also $(\beta, 0, 2\varepsilon+\delta)$-perfectly generalizing. For any distribution D over $\mathscr{X}$, let Sim_D be the canonical distribution witnessing the perfect generalization property. Then, by the definition of $(0, 2\varepsilon+\delta)$-indistinguishability, we have that with probability at least $1-\beta$ over a draw of a random dataset $S \sim D^m$,

$$d_{TV}(\mathscr{A}(S), Sim_D) \leq 2\varepsilon + \delta.$$

From the guarantee of correlated sampling, we have that

$$\mathbf{Pr}_{r' \sim R}[CS(Q_S, \mathscr{Y}, r') \neq CS(Sim_D, \mathscr{Y}, r')] \leq 2d_{TV}(Q_S, Sim_D).$$

Using the bound on TV distance from perfect generalization, we get that with probability at least $1-2\beta$ over the draw of two datasets $S_1, S_2 \sim D^m$, we have that

$$\mathbf{Pr}_{r' \sim R}[CS(Q_{S_1}, \mathscr{Y}, r') \neq CS(Sim_D, \mathscr{Y}, r')] \leq 2(2\varepsilon+\delta)$$

and

$$\mathbf{Pr}_{r' \sim R}[CS(Q_{S_2}, \mathscr{Y}, r') \neq CS(Sim_D, \mathscr{Y}, r')] \leq 2(2\varepsilon+\delta).$$

Consider the event $CS(Q_{S_1}, \mathscr{Y}, r') \neq CS(Q_{S_2}, \mathscr{Y}, r')$. Clearly, this implies either $CS(Q_{S_1}, \mathscr{Y}, r') \neq CS(Sim_D, \mathscr{Y}, r')$ or $CS(Q_{S_2}, \mathscr{Y}, r') \neq CS(Sim_D, \mathscr{Y}, r')$. Hence, we can write that with probability $1-2\beta$ over draws of S_1 and S_2,

$$\mathbf{Pr}_{r' \sim R}[CS(Q_{S_1}, \mathscr{Y}, r') \neq CS(Q_{S_2}, \mathscr{Y}, r')] \leq 4(2\varepsilon+\delta).$$

Taking the expectation with respect to the draws of S_1 and S_2 gives us

$$\mathbf{Pr}_{r' \sim R, S_1, S_2 \sim D^m}[CS(Q_{S_1}, \mathscr{Y}, r') \neq CS(Q_{S_2}, \mathscr{Y}, r')] \leq 4(2\varepsilon+\delta+\beta)$$

which proves the result. □

Combining the above result and Corollary 5.3.16, we get a transformation from approximate differentially private algorithms to replicable algorithms.

Corollary 5.3.18. *Fix $m \in \mathbb{N}$, sufficiently small $\rho \in (0,1)$. Let $\varepsilon = \frac{\rho}{\sqrt{8m\log(1/\rho)}}$, $\delta \leq \frac{\varepsilon\rho^6}{m^2}$. Let $\mathscr{A} : \mathscr{X}^m \to \mathscr{Y}$ be an (ε, δ)-DP algorithm with finite output space. Fix a distribution D over $\mathscr{X}$, and let $S \sim D^m$. Then, Algorithm 25 run with inputs S and algorithm $\mathscr{A}$ is $O(\rho)$-replicable. Additionally, on any fixed dataset, the output distribution of Algorithm 25 is the same as that of $\mathscr{A}$.*

Proof. From Corollary 5.3.16, we have that Algorithm $\mathscr{A}$ is $(O(\rho), O(\rho), O(\rho))$-one-way perfectly generalizing. Then, applying Theorem 5.3.17 proves that the transformation in Algorithm 25 gives a $O(\rho)$-replicable algorithm. Correlated sampling does not change the marginal distribution of the algorithm applied to a dataset and hence the second part of the corollary is proved. □

5.3.4 Replicability Implies Perfect Generalization

In this section, we show how to convert a replicable algorithm to a perfectly generalizing algorithm at a poly-logarithmic cost in $1/\delta$ (where δ is the additive perfect generalization parameter).

It's straightforward to show that (δ, δ)-replicability can be used to obtain $(O(\delta), 0, O(\delta))$-perfect generalization by translating from collision probability to total variation distance. However, since we typically want δ to be very small (often inverse polynomial in the number of samples m), obtaining such small parameters starting from, say, 0.1-replicability comes at a significant cost. This is because amplifying 0.1-replicability to (δ, δ)-replicability incurs a multiplicative sample complexity overhead of $O(1/\delta^2)$, which is tight by known lower bounds for replicability [ILPS22, Theorem 7.1], and prohibitively large for many applications. For example, our lower bounds showing tasks where replicability has quadratically higher sample cost than differential privacy (see Section 5.5) follow from proving such lower bounds on perfectly generalizing algorithms with

δ polynomially small in the dataset size, and then applying our conversion from replicability to perfect generalization. If such a conversion required $1/\delta^2$ samples, then this would not give us any non-trivial lower bounds on the sample cost associated with replicably solving these problems.

However, this idea still leaves hope, because it achieves $\varepsilon = 0$. Hence, by settling for larger ε, we hope to avoid this problem.

Our approach is inspired by a natural attempt to amplify weak replicability parameters into strong parameters. Suppose we wish to turn a $(0.01, 0.01)$-replicable algorithm A into a $(0.01, \delta)$-replicable one. We know that with probability at least 0.99 over the choice of the randomness r for A, there is a canonical output z such that $A(S;r) = z$ with high probability over the sample S. Consider running A using $k = O(\log(1/\delta))$ independent sequences of coin tosses, $r_1, \ldots, r_k$, then with probability $1-\delta$, at least one of these sequences will have such a canonical output. Moreover, when such a canonical output exists, we can identify it by running $A(\cdot; r_j)$ on many independent *samples* S and choosing the plurality outcome if it appears enough times. Unfortunately, there is an obstacle here to directly designing a replicable algorithm. The problem is that there may be many good sequences of coin tosses, each with their own canonical outputs, and it is unclear how to replicably identify a single one.

By relaxing our goal to achieving perfect generalization instead of replicability, we can instead use the exponential mechanism to *sample* from the set of plurality outcomes. We define the score of the plurality output c_j for coin r_j to be the number of datasets S on which $A(S;r_j) = c_j$, and sample such a c_j with probability proportional to exponential in its score. We are able to show that the resulting algorithm is $(\delta, \varepsilon, \delta)$-perfectly generalizing with $\varepsilon > 0$, but there are several technical nuances that make our analysis not quite straightforward from the standard guarantees of the exponential mechanism. For instance, we need to deal with the fact that the sets of plurality outputs could differ when our algorithm is run on two i.i.d. datasets drawn from the same distribution. Another interesting feature of this proof is that unlike standard uses of the exponential mechanism to obtain differential privacy or perfect generalization, we need to invoke the *accuracy* of the exponential mechanism in our proof of perfect generalization.

Algorithm 26. Transformation from replicable algorithm $\mathscr{A}$ to perfectly generalizing algorithm $\mathscr{A}'$

```
**Input:** Sample access to distribution $D$, description of $(0.01, 0.01)$-replicable algorithm $\mathscr{A} : \mathscr{X}^* \to \mathscr{Y}$, sample complexity parameter $m$, perfect generalization parameters $\varepsilon, \delta, \beta$
**Output:** $y \in \mathscr{Y}$
1: Let $k = O(\log(1/\delta))$, and $t = O\left(\frac{\log^4(1/\beta)\log(1/\varepsilon)}{\varepsilon^2}\right)$.
2: Draw uniformly random coins $r_1, r_2, \ldots, r_k$ for algorithm $\mathscr{A}$.
3: Draw $k$ sets $\mathbf{S_i}$, each of $t$ samples $S_{i,j} \sim D^m$.
4: for all $j \in [k]$ do
5:     for all $i \in [t]$ do
6:         Run $\mathscr{A}$ with coins $r_j$ and sample $S_{i,j}$ to get output $z_{i,j}$.
7:     Let $c_j = \arg\max_{z \in \mathscr{Y}} \sum_{i=1}^{t} \mathbb{1}[z_{i,j} = z]$, and let $\mathrm{score}\big((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k)\big) = \sum_{i=1}^{t} \mathbb{1}[z_{i,j} = c_j]$.
8: Let $C = \{(1, c_1), \ldots, (k, c_k)\}$. Run the exponential mechanism on the set $C$ with the score function score$(.,.)$, sensitivity parameter $4\sqrt{t\log(8kt/\beta)}$, and privacy parameter $\varepsilon$ to get value $(j^*, c_{j^*})$.
9: return output $c_{j^*}$ of the previous step.
```

We prove that the above algorithm is sample perfectly generalizing. Note that this can be converted to a perfectly generalizing algorithm with asymptotically the same parameters (for both perfect generalization and accuracy) by setting the δ parameter to be δ^2 instead and invoking Lemma 5.3.6.

Theorem 5.3.19. *Fix sufficiently small $\delta, \gamma > 0$ and $0 < \varepsilon \leq 1$. Every $(0.01, 0.01)$-replicable algorithm $\mathscr{A}$ with m samples that successfully solves a statistical task with probability at least $1 - \gamma^2$ can be converted to a $(2\delta, \varepsilon, 2\delta)$-sample perfectly generalizing algorithm $\mathscr{A}'$ taking $O\left(\frac{m\log(1/\varepsilon)}{\varepsilon^2}\mathrm{poly}\log(1/\delta)\right)$ samples, that succeeds on the statistical task with probability at least*

$1 - O(\delta) - \gamma \log(1/\delta)$.

Proof. Fix a distribution D over the input set $\mathscr{X}$. Our proof is constructive; the corresponding algorithm is given in Algorithm 26 ($\mathscr{A}'$), and we feed it with the following inputs: a description of algorithm $\mathscr{A}$, sample complexity parameters m, and perfect generalization parameters $(\delta, \varepsilon, \delta)$. (We also give it sample access to distribution D). We will start by proving sample perfect generalization of Algorithm 26.

Claim 5.3.20. *Algorithm $\mathscr{A}'$ (represented in Algorithm 26) with the input parameters specified in the previous paragraph is $(2\delta, \varepsilon, 2\delta)$-sample perfectly generalizing.*

Proof. Consider two samples S and S' drawn independently from D^{mkt}. We consider Algorithm 26 run on both of these samples and argue that their output distributions are close in the sense required by sample perfect generalization.

Step 1: At least one coin sequence is good w.h.p.

We say that a choice of the random coin tosses of Algorithm $\mathscr{A}$ is "good" if it has a 0.99-canonical output and call it "bad" otherwise. Then by the two parameter definition of replicability, a random coin sequence is "bad" with probability at most 0.01. Hence, the probability that all k coins sequences drawn in Step 2 of Algorithm 26 are bad is at most $(0.01)^k \leq \delta^2$ for $k = O(\log(1/\delta))$. Let E_{coin} represent the event that there is at least one good coin. We will now condition on E_{coin} occurring; fix any set of coins $r_1, \ldots, r_k$ that has non-zero probability of occurring under this conditioning. We will first consider $\mathscr{A}$ run on the two independent datasets S and S' with the same random coins fixed above.

Step 2: Empirical output frequencies are close on two independent datasets.

We define stage j of Algorithm 26 as the process involved in generating c_j (i.e., one iteration of the outer loop in Step 4). We now use uniform convergence to argue that with high probability over the samples, the empirical frequencies of the outputs of all stages, i.e. all values of $\mathit{Score}\left((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right)$, are close to their expectation.

Start by defining H to be the function class consisting of point functions, i.e., functions of the form $h_\ell(x)=1$ if $x=\ell$, and $h_\ell(x)=0$ otherwise, for every $\ell\in\mathscr{Y}$. It is easy to prove that the VC dimension of H is equal to 1. Let Q_j be the distribution of the output of the replicable algorithm $\mathscr{A}$ when run with coin r_j on a random sample.

Then, using uniform convergence (Theorem 5.2.16), we get for every fixed j and for every $\gamma>0$, that

$$\mathbf{Pr}_{z_{1,j},\ldots,z_{t,j}\sim Q_j}\left[\sup_{h_\ell\in H}\left|\frac{1}{t}\sum_{i=1}^{t}\mathbb{1}[h_\ell(z_{i,j})=1]-\mathbf{Pr}_{z\sim Q_j}[h_\ell(z)=1]\right|\geq\gamma\right]\leq 8te^{-\gamma^2t/8}.$$

Observe that $\frac{1}{t}\sum_{i=1}^{t}\mathbb{1}[h_\ell(z_{i,j})=1]=\frac{1}{t}\sum_{i=1}^{t}\mathbb{1}[z_{i,j}=\ell]$. Similarly, $\mathbf{Pr}_{z\sim Q}[h_\ell(z)=1]=\mathbf{Pr}_{z\sim Q_j}[z=\ell]$. Hence, we get that

$$\mathbf{Pr}_{z_{1,j},\ldots,z_{t,j}\sim Q_j}\left[\sup_{\ell\in\mathscr{Y}}\left|\frac{1}{t}\sum_{i=1}^{t}\mathbb{1}[z_{i,j}=\ell]-\mathbf{Pr}_{z\sim Q_j}[z=\ell]\right|\geq\gamma\right]\leq 8te^{-\gamma^2t/8}.$$

Setting $\gamma=2\sqrt{\frac{\log(8kt/\delta)}{t}}$, we get that

$$\mathbf{Pr}_{z_{1,j},\ldots,z_{t,j}\sim Q_j}\left[\sup_{\ell\in\mathscr{Y}}\left|\sum_{i=1}^{t}\mathbb{1}[z_{i,j}=\ell]-t\mathbf{Pr}_{z\sim Q_j}[z=\ell]\right|\geq 2\sqrt{t\log(8kt/\delta)}\right]\leq\frac{\delta^2}{2k}.$$

Using a union bound over all k stages of the algorithm, this guarantees us that the empirical frequencies (and the values score $\left((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right)$ are all within $2\sqrt{t\log(8kt/\delta)}$ of their expectations with probability at least $1-\delta^2/2$.

Note that since we conditioned on a fixed random coin sequence, all the randomness in $z_{i,j}$ comes from the data sample. Hence, if we consider another sample $S'=(\mathbf{S}'_1,\mathbf{S}'_2,\ldots,\mathbf{S}'_k)$ drawn i.i.d .from D^{mkt}, we have that with probability at least $1-\delta^2$ over the draws of S and S' that for all $j\in[k]$

and all $\ell \in \mathscr{Y}$ that score $((j,\ell),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k))$ and score $\big((j,\ell),(\mathbf{S}'_1,\mathbf{S}'_2,\ldots,\mathbf{S}'_k)\big)$ are both within $2\sqrt{t\log(8kt/\delta)}$ of their expectations, and are hence within $4\sqrt{t\log(8kt/\delta)}$ of each other. We call this event E_{sample}, and fix any sample pairs (S,S') that occur with non-zero probability conditioned on this event. This allows us to argue that with probability at least $1-\delta^2$, for all $j\in[k]$,

$$|\,\mathrm{score}\,\big((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\big) - \mathrm{score}\,\big((j,c'_j),(\mathbf{S}'_1,\mathbf{S}'_2,\ldots,\mathbf{S}'_k)\big)\,| \le 4\sqrt{t\log(8kt/\delta)}. \tag{5.6}$$

This follows directly from the above argument if $c_j = c'_j$, but if they are not equal, it also holds since otherwise either c_j or c'_j would not be a plurality output in stage j of the corresponding runs (since there would be an output that occurs more times in stage j). This is because uniform convergence guarantees us that if c_j occurs a times in stage j when the algorithm is run on dataset S, then c_j occurs atleast $a-4\sqrt{t\log(8kt/\delta)}$ times in stage j when the algorithm is run on dataset S'. Hence, if c'_j occurs less than $a-4\sqrt{t\log(8kt/\delta)}$ in stage j, then we'd get that c_j would be the plurality output of stage j in the run on dataset S' and not c'_j, which is a contradiction. This shows that score $\big((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\big)$ − score $\Big((j,c'_j),(\mathbf{S}'_1,\mathbf{S}'_2,\ldots,\mathbf{S}'_k)\Big) \le 4\sqrt{t\log(8kt/\delta)}$; the other direction can be proved similarly.

Step 3: Arguing that there is at least one canonical output c_j with high score.

Conditioned on E_{coin}, we know that the run of Algorithm 26 on S has at least one coin sequence with a 0.99-canonical output z. Suppose r_j is such a coin sequence. From the settings of k and t, we get that $2\sqrt{t\log(8kt/\delta)} \le 0.09t$. Hence, conditioned further on E_{sample}, we know that this canonical output z is equal to the plurality output c_j in stage j, and that score $\big((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\big)$ is at least $0.9t$. Hence, there exists an candidate (j,c_j) with score at least $0.9t$.

Step 4: Arguing that probable outputs (j^*, c_{j^*}) are in both output sets C and C'.

By the accuracy guarantee of the exponential mechanism (Lemma 5.2.7), we have that

$$\mathbf{Pr}\left[\max_{j\in[k]} \text{score}\left((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right) - \text{score}\left((j^*,c_{j^*}),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right) \geq 2\Delta\frac{\ln k + k}{\varepsilon}\right] \leq e^{-k},$$

where $\Delta = 4\sqrt{t\log(8kt/\delta)}$. So, for the settings of t and k, we get that $2\Delta\frac{\ln k+k}{\varepsilon} = O(\frac{k\sqrt{t\log(8kt/\delta)}}{\varepsilon}) = o(t)$. Hence, we have that

$$\mathbf{Pr}\left[\text{score}\left((j^*,c_{j^*}),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right) \geq 0.9t - 2\Delta\frac{\ln k + k}{\varepsilon}\right] \leq e^{-k} = \delta,$$

which implies that for sufficiently small δ,

$$\mathbf{Pr}\left[\text{score}\left((j^*,c_{j^*}),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right) \geq 0.8t\right] \leq e^{-k} = \delta.$$

Let's consider any such c_{j^*}. By the conditioning on E_{sample}, we have that c_{j^*} occurs more than $0.8t - 4\sqrt{t\log(kt/\delta)} > 0.5t$ times in the output set of stage j^* when Algorithm 26 is run on the sample S'. Hence, c'_{j^*} is also equal to c_{j^*}.

Step 5: Proving that $\mathscr{A}'(S,r) \approx_{\varepsilon,\delta} \mathscr{A}'(S',r)$ w.h.p.

We exploit the fact that two random variables C and D are (ε,δ)-indistinguishable if w.p. $\geq 1-\delta$ over a draw o from the distribution of C, we have $e^{-\varepsilon}\mathbf{Pr}[D=o] \leq \mathbf{Pr}[C=o] \leq e^{\varepsilon}\mathbf{Pr}[D=o]$, and vice versa for a draw from the distribution of D [KS14, Lemma 3.3, Part 1].

We proved in Step 4 that fixing any coins and sample pairs that have non-zero probability of occurring conditioned on E_{coin} and E_{sample}, with probability at least $1-\delta$ from a draw of $A'(S,r)$ (where the randomness is only that of the exponential mechanism), the output (j^*, c_{j^*}) occurs in both the sets C and C'. For all such outputs, our idea is to use the differential privacy analysis of the exponential mechanism.

A technical obstacle we need to surmount is that the output sets C and C' might be different, and so the normalizing factors used in the exponential mechanism will vary accordingly. We deal with this by invoking Inequality 5.6, which points out that even though the output sets are different, the scores score$\big((j,c_j),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\big)$ and score$\Big((j,c'_j),(\mathbf{S}'_1,\mathbf{S}'_2,\ldots,\mathbf{S}'_k)\Big)$ can differ by at most the sensitivity specified in Step 8 where the exponential mechanism is invoked.

Hence, exactly mimicking the differential privacy analysis of the exponential mechanism (see e.g., [DR14], Theorem 3.10) conditioned on E_{coin} and E_{sample}, with probability at least $1-\delta$ from a draw (j^*,c'_{j^*}) of $\mathscr{A}'(S,r)$, we get that

$$e^{-\varepsilon}\mathbf{Pr}[\mathscr{A}'(S,r)=(j^*,c_{j^*})] \le \mathbf{Pr}[\mathscr{A}'(S',r)=(j^*,c'_{j^*})] \le e^{\varepsilon}\mathbf{Pr}(\mathscr{A}'(S,r)=(j^*,c_{j^*})]$$

and, moreover, $c'_{j^*}=c_{j^*}$. By symmetry (since S and S' are both independent samples from the distribution with the same properties), conditioned on E_{coin} and E_{sample}, we get that with probability at least $1-\delta$ from a draw (j^*,c_{j^*}) of $\mathscr{A}'(S',r)$,

$$e^{-\varepsilon}\mathbf{Pr}[\mathscr{A}'(S,r)=(j^*,c_{j^*})] \le \mathbf{Pr}[\mathscr{A}'(S',r)=(j^*,c_{j^*})] \le e^{\varepsilon}\mathbf{Pr}[\mathscr{A}'(S,r)=(j^*,c_{j^*})].$$

Hence, we have proved that conditioned on any fixed coins and sample pairs with non-zero probability of occurring conditioned on E_{coin} and E_{sample}, we have $\mathscr{A}'(S,r)\approx_{\varepsilon,\delta}\mathscr{A}'(S',r)$. Now, using the law of total probability, we get that

$$\mathbf{Pr}_{r_1,\ldots,r_k}\left[\mathbf{Pr}_{S,S'\sim D^{mkt}}\left[\mathscr{A}'(S;r_1,\ldots,r_k)\approx_{\varepsilon,\delta}\mathscr{A}'(S';r_1,\ldots,r_k)\right]\ge 1-\delta^2\right]\ge 1-\delta^2. \qquad (5.7)$$

Step 6: Switch quantifiers to get sample perfect generalization:

Now, we switch the quantifiers in equation 5.7.

$$\begin{aligned}
&\mathbf{Pr}_{S,S'\sim D^{mkt},r_1,\ldots,r_k}\left[\mathscr{A}'(S;r_1,\ldots,r_k)\approx_{\varepsilon,\delta}\mathscr{A}'(S';r_1,\ldots,r_k)\right]\geq 1-2\delta^2\\
\Longrightarrow\ &\mathbb{E}_{S,S'\sim D^{mkt}}\left[\mathbf{Pr}_{r_1,\ldots,r_k}\left[\mathscr{A}'(S;r_1,\ldots,r_k)\approx_{\varepsilon,\delta}\mathscr{A}'(S';r_1,\ldots,r_k)\right]\right]\geq 1-2\delta^2\\
\Longrightarrow\ &\mathbf{Pr}_{S,S'\sim D^{mkt}}\left[\mathbf{Pr}_{r_1,\ldots,r_k}\left[\mathscr{A}'(S;r_1,\ldots,r_k)\approx_{\varepsilon,\delta}\mathscr{A}'(S';r_1,\ldots,r_k)\right]\geq d\right]\geq \frac{1-2\delta^2-d}{1-d}.
\end{aligned}$$

Here, the last inequality holds by the reverse Markov inequality. Setting $d=1-\delta$, we get that

$$\mathbf{Pr}_{S,S'\sim D^{mkt}}\left[\mathbf{Pr}_{r_1,\ldots,r_k}\left[\mathscr{A}'(S;r_1,\ldots,r_k)\approx_{\varepsilon,\delta}\mathscr{A}'(S';r_1,\ldots,r_k)\right]\geq 1-\delta\right]\geq 1-2\delta.$$

Now, using the fact that if $X\approx_{\varepsilon,\delta}Y$ and $M\approx_{\varepsilon,\delta}N$, then $\alpha X+(1-\alpha)M\approx_{\varepsilon,\delta}\alpha Y+(1-\alpha)N$ for every $\alpha\in[0,1]$ (i.e., (ε,δ)-indistinguishability is preserved under convex combinations), we get that

$$\mathbf{Pr}_{S,S'\sim D^{mkt}}\left[\mathscr{A}'(S)\approx_{\varepsilon,2\delta}\mathscr{A}'(S')\right]\geq 1-2\delta.$$

This proves that $\mathscr{A}'$ with the specified inputs is $(2\delta,\varepsilon,2\delta)$-sample perfectly generalizing, as required. Next, we deal with accuracy. □

Claim 5.3.21. *If Algorithm $\mathscr{A}$ succeeds at a statistical task with probability at least $1-\gamma^2$, Algorithm $\mathscr{A}'$ succeeds at the same statistical task with probability at least $1-O(\delta)-\gamma\log(1/\delta)$.*

Proof. Recall the definition of success for a statistical task. The statistical task is defined by a set of distribution, set pairs. For every distribution D, there is an associated good set of outputs O_D. An algorithm succeeds at this task with probability at least $1-\gamma$ if it outputs a member of this good set with at least that probability (taken over random samples from D and any internal coins of the algorithm).

If $\mathscr{A}$ succeeds at the task with probability at least $1-\gamma^2$, by using reverse Markov's inequality as in Step 6 of the previous proof, we have that

$$\mathbf{Pr}_r\left[\mathbf{Pr}_{S\sim D^m}\left[\mathscr{A}(S;r)\in O_D\right]\geq 1-\gamma\right]\geq 1-\gamma \tag{5.8}$$

We say a coin sequence r_j is "accurate" if the inner inequality under the probability is satisfied. Recall that we call a coin sequence r_j "good" if it has a 0.99-canonical output. By the analysis in Step 1 of the previous proof, we have that with probability at least $1-\delta^2$, there is a good coin sequence among the runs $r_1,\ldots,r_k$. Now, the probability that all k coin sequences are "accurate" is equal to $(1-\gamma)^{\log(1/\delta)}\geq 1-\gamma\log(1/\delta)$. (This follows from Bernoulli's inequality $(1+a)^k\geq 1+ak$ for all $a\geq -1$ and non-negative integers k.) Hence, by a union bound, the probability that the set of runs $r_1,\ldots,r_k$ both contains a good coin sequence and that all the coin sequences in the set are accurate is at least $1-\gamma\log(1/\delta)-\delta^2$. Call this event $E_{coin-acc}$ and condition on it. Additionally, condition on E_{sample} as defined in Step 2 of the previous proof. Then, by the analysis in Step 4 of the previous proof, we have that

$$\mathbf{Pr}\left[\text{score}\left((j^*,c_{j^*}),(\mathbf{S}_1,\mathbf{S}_2,\ldots,\mathbf{S}_k)\right)\geq 0.8t\right]\leq e^{-k}=\delta,$$

which implies that the exponential mechanism outputs a plurality output that occurs at least $0.8t$ times in its stage with probability at least $1-\delta$. Since we have conditioned on E_{sample}, we have that empirical frequencies are close to their expected values, and hence the exponential mechanism outputs the canonical output of a coin sequence that is at least 0.25-good with probability at least $1-\delta$. Using the law of total probability to remove the conditioning on E_{sample}, we get that the exponential mechanism outputs the canonical output of a coin sequence that is at least 0.25-good with probability at least $1-\delta-\delta^2$ (since event E_{sample} happens with probability at least $1-\delta^2$). Note that since all the drawn coin sequences are accurate, we get that the canonical output for every such sequence is in the good set O_D (otherwise, the inequality inside the outer probability in equation 5.8 would not be satisfied). Hence, conditioned on $E_{coin-acc}$, we have that $\mathscr{A}'$ outputs an element of the good set with probability at least $1-\delta-\delta^2$. Using the law of total probability, we then get that $\mathscr{A}'$ outputs an element of the good set with probability at least $1-\delta-2\delta^2-\gamma\log(1/\delta) = 1-O(\delta)-\gamma\log(1/\delta)$. □

Hence, combining the two claims on perfect generalization and accuracy, we complete the proof of the theorem. □

5.4 Separating Stability: Computational Barriers

In this section, we show that standard cryptographic assumptions imply there cannot exist computationally efficient transformations from differentially private algorithms to replicable ones. Moreover, such cryptographic assumptions are necessary: if one-way functions do not exist, there exists an efficient algorithm for correlated sampling (and therefore for converting DP to replicability as well via Corollary 5.3.18).

In Section 5.4.1, we define $\Pi_{RandEnc}$, a statistical promise problem. Given a public key **pk** and a dataset of ciphertexts encrypting the same bit b under **pk**, a solution to $\Pi_{RandEnc}$ is any encryption of b under **pk**. In Section 5.4.1, we give a simple algorithm `DPRandEnc` solving $\Pi_{RandEnc}$. `DPRandEnc` is (ε,δ)-differentially private and runs in polynomial time. In Section 5.4.1, we show that the existence of an efficient replicable algorithm for $\Pi_{RandEnc}$ would violate the security guarantee of the encryption scheme. Thus, assuming randomizable encryption schemes exist, there is no efficient transformation from DP algorithms to replicable algorithms for $\Pi_{RandEnc}$. $\Pi_{RandEnc}$ can be instantiated with any PKE satisfying the requirements of Definition 5.4.2, but to demonstrate that these requirements are not unreasonable, in Section 5.4.1 we show that they are satisfied by the Goldwasser-Micali public-key encryption scheme [GM82]. Therefore the hardness of quadratic residuosity is sufficient to show hardness for the transformation from differential privacy to replicability.

In Section 5.4.2, we give an algorithm for correlated sampling that is efficient so long as no one-way functions exist. This algorithm can in turn be used in Algorithm 25, to implement the correlated sampling step of the transformation from a one-way perfectly generalizing algorithm to a replicable one, giving an efficient transformation.

5.4.1 Cryptographic Hardness of Replicability

We define a promise problem $\Pi_{RandEnc}$ (Definition 5.4.3) for a public-key encryption scheme. $\Pi_{RandEnc}$ is parameterized by a public-key $\mathbf{pk}$ for a public-key encryption scheme $\mathscr{E}$ with message space $\{0,1\}$ and ciphertext space $\mathscr{C}$. An instance of $\Pi_{RandEnc}$ consists of a sample of m elements c_i, drawn i.i.d. from an unknown distribution D over $\mathscr{C}$. Promised that either

1. D is supported entirely on encryptions of 0 under $\mathbf{pk}$ or
2. D is supported entirely on encryptions of 1 under $\mathbf{pk}$,

Problem $\Pi_{RandEnc}$ asks the algorithm to output an encryption of 0 under $\mathbf{pk}$ in the first case and an encryption of 1 under $\mathbf{pk}$ in the second case.

We show that if the public-key encryption scheme supports a strong form of rerandomization, Problem $\Pi_{RandEnc}$ can be efficiently solved with a differentially private algorithm. At the same time, $\Pi_{RandEnc}$ cannot be efficiently solved using a replicable algorithm, assuming the security of the underlying encryption scheme. Thus, in this setting, there cannot be an efficient black-box reduction from DP algorithms to replicable algorithms.

For our construction, we use a standard definition for public-key encryption.

Definition 5.4.1 (Public-Key Encryption Scheme)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and $\mathscr{E} =$ $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a tuple of algorithms running in time* $\mathrm{poly}(\lambda)$*, with* $\mathsf{KeyGen} : 1^* \to \mathscr{K}_p \times \mathscr{K}_s$*,* $\mathsf{Enc} : \mathscr{K}_p \times \{0,1\} \to \mathscr{C}$*, and* $\mathsf{Dec} : \mathscr{K}_s \times \mathscr{C} \to \{0,1\} \cup \perp$*. We say $\mathscr{E}$ is a* public-key encryption scheme *if it has the following properties.*

- *Correctness: Let* $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$*,* $b \in \{0,1\}$*, and* $c \leftarrow \mathsf{Enc}(\mathbf{pk}, b)$*. Then* $\mathsf{Dec}(\mathbf{sk}, c) = b$*.*

- *Security: There exists a negligible function* $\varepsilon(\lambda)$*, such that for all adversaries* $\mathscr{A}$ *running in time* $\mathrm{poly}(\lambda)$*, letting* $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$ *we have*

$$|\mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)]| < \varepsilon(\lambda).$$

We also require that a public-key encryption scheme allows for efficient, publicly computable ciphertext verification and rerandomization procedures.

Definition 5.4.2 (Randomizeable Encryption Scheme)**.** *Let* $\mathscr{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme. We call* $\mathscr{E}$ *a* randomizeable encryption scheme *if it supports the following additional procedures.*

- ***(Perfect) Verification of Ciphertexts:*** *There exists a deterministic polytime algorithm* $\mathscr{V}$ *such that, for an honestly generated key pair* $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$ *and value c,*
  - *If* $\mathsf{Dec}(\mathbf{sk}, c) \in \{0,1\}$*, then* $\mathscr{V}(\mathbf{pk}, c) = 1$
  - *If* $\mathsf{Dec}(\mathbf{sk}, c) = \perp$*, then* $\mathscr{V}(\mathbf{pk}, c) = 0$

- ***(Perfect) Randomization of Ciphertexts:*** *There exists a randomized polytime algorithm* $\mathsf{Rerandomize}$ *such that, for all honestly generated key pairs* $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$*, and all ciphertexts* c_1, c_2 *such that* $\mathsf{Dec}(\mathbf{sk}, c_1) = \mathsf{Dec}(\mathbf{sk}, c_2)$*,*
  - $d_{TV}(\mathsf{Rerandomize}(\mathbf{pk}, c_1), \mathsf{Rerandomize}(\mathbf{pk}, c_2)) = 0$
  - $\mathsf{Dec}(\mathbf{sk}, \mathsf{Rerandomize}(\mathbf{pk}, c)) = \mathsf{Dec}(\mathbf{sk}, c)$

Consider the following search problem $\Pi_{RandEnc}$. Given a public key **pk** for an encryption scheme $\mathscr{E}$, and an i.i.d. sample of m elements from a distribution D supported on encryptions under **pk** of a fixed bit $b \in \{0,1\}$, output an encryption of b under **pk**.

Definition 5.4.3 (Ciphertext Identification Problem)**.** *An instance of* $\Pi_{RandEnc}$ *is defined as follows. Let* $\mathscr{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a randomizeable encryption scheme (Definition 5.4.2). Let* $\lambda, m \in \mathbb{N}$*, and let* D *be a distribution over the ciphertext space* $\mathscr{C}$ *of* $\mathscr{E}$*. Given public key* $\mathbf{pk}$*, honestly generated as* $(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)$*, and a sample* $S \sim D^m$ *drawn i.i.d. from* D*, output an element* $c^* \in \mathscr{C} \cup \perp$ *such that*

1. *If* $\mathsf{Dec}(\mathbf{sk}, c) = 1$ *for all* $c \in supp(D)$*,* $\mathsf{Dec}(\mathbf{sk}, c^*) = 1$

2. *If* $\mathsf{Dec}(\mathbf{sk}, c) = 0$ *for all* $c \in supp(D)$*,* $\mathsf{Dec}(\mathbf{sk}, c^*) = 0$

DP Algorithm for $\Pi_{RandEnc}$

In this subsection, we present a differentially private algorithm `DPRandEnc` for $\Pi_{RandEnc}$. Our algorithm removes from the dataset S all c_i for which verification fails, i.e., $\mathscr{V}(\mathbf{pk}, c_i) = 0$. It then pads the remaining elements with k encryptions of 0 under $\mathbf{pk}$ and k encryptions of 1 under $\mathbf{pk}$. An element c_i from the new dataset is then chosen uniformly at random, and the algorithm outputs $\mathsf{Rerandomize}(\mathbf{pk}, c_i)$.

Padding the dataset with additional ciphertexts, balanced between encryptions of 0 and 1, guarantees privacy by ensuring that exchanging any element of S for another will not significantly change the probability that the ciphertext chosen for rerandomization encrypts a particular bit. If the distribution D is supported on one of the two promised distributions, `DPRandEnc` will be correct unless it chooses to rerandomize an inserted ciphertext which encrypts the incorrect bit. So long as the input sample is of size m much larger than k, this will happen only with small probability.

Algorithm 27. `DPRandEnc`, differentially private algorithm for $\Pi_{RandEnc}$

```
Result: Outputs a ciphertext $c \in \mathscr{C}$
Input: Sample $S$ of $m$ elements drawn i.i.d. from $D$
 Parameters:
   • Privacy $\varepsilon$, failure probability $\beta$, padding length $k = \frac{1}{\varepsilon}$
   • Sample Complexity $m = m(\varepsilon, \beta) \in O\left(\frac{1}{\varepsilon\beta}\right)$
Algorithm:
  1. For $i \in [m]$, remove $c_i$ from $S$ if $\mathscr{V}(\mathbf{pk}, c_i) = 0$
  2. Add $k$ ciphertexts $\mathsf{Enc}(\mathbf{pk}, 0)$ to the dataset
  3. Add $k$ ciphertexts $\mathsf{Enc}(\mathbf{pk}, 1)$ to the dataset
  4. Choose $c$ uniformly at random from the new dataset
return $\mathsf{Rerandomize}(\mathbf{pk}, c)$
```

Lemma 5.4.4. *Let $\varepsilon, \beta \in (0, 1/2)$. Then for $m \in \Omega(1/(\varepsilon\beta))$ and $k = 1/\varepsilon$, $\Pi_{RandEnc}$ (Algorithm 27) runs in time* $\mathrm{poly}(\lambda, 1/\varepsilon, 1/\beta)$*, is ε-DP, and correct except with probability at most β.*

Proof. We begin by showing `DPRandEnc` is ε-DP. Note that the last step of `DPRandEnc` calls $\mathsf{Rerandomize}$ on a ciphertext c that is guaranteed to be a valid encryption of a bit $b \in \{0, 1\}$, since all inputs failing verification are removed from S before c is drawn, and only valid ciphertexts under $\mathbf{pk}$ are added to the input dataset.

We will bound how much the probability that c encrypts a fixed bit b can differ across neighboring data sets. Let c be a random variable denoting the ciphertext chosen for rerandomization. For all $b \in \{0, 1\}$ and neighboring datasets S, S', let S_b, $S_{\neg b}$, and $S_\perp$ denote the subsets of S such that $\mathsf{Dec}(\mathbf{sk}, c) = b$, $\mathsf{Dec}(\mathbf{sk}, c) = \neg b$, and $\mathsf{Dec}(\mathbf{sk}, c) = \perp$ respectively, and let S'_b, $S'_{\neg b}$, and $S'_\perp$ be defined analogously. Then

$$\begin{aligned}\frac{\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=b \mid S]}{\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=b \mid S']} &= \frac{|S_b|+k}{m-|S_\perp|+2k}\cdot\frac{m-|S'_\perp|+2k}{|S'_b|+k}\\ &= \frac{|S_b|+k}{|S_b|+|S_{\neg b}|+2k}\cdot\frac{|S'_b|+|S'_{\neg b}|+2k}{|S'_b|+k}\\ &\leq \frac{|S_b|+k}{|S_b|+|S_{\neg b}|+2k}\cdot\frac{|S_b|+|S_{\neg b}|+2k}{|S_b|-1+k}\\ &= \frac{|S_b|+k}{|S_b|-1+k}\\ &\leq \frac{k+1}{k},\end{aligned}$$

where the first inequality follows from S,S' neighboring, and the fact that for $a>b$, $\frac{a}{b-1}>\frac{a+1}{b}$, so assuming $|S_b|\geq 1$ and $|S'_b|=|S_b|-1$ maximizes the rightmost fraction. Because the output distribution of $\mathsf{Rerandomize}(\mathbf{pk},c)$ is the same for all ciphertexts encrypting the same bit under $\mathbf{pk}$, it follows that for all subsets $T\subseteq\mathscr{C}$,

$$\begin{aligned}\mathbf{Pr}[\mathtt{DPRandEnc}(S)\in T] = {}&\mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk},c)\in T \mid \mathsf{Dec}(\mathbf{sk},c)=1]\cdot\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=1 \mid S]\\ &+\mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk},c)\in T \mid \mathsf{Dec}(\mathbf{sk},c)=0]\cdot\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=0 \mid S].\end{aligned}$$

Using p_b to denote $\mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk},c)\in T \mid \mathsf{Dec}(\mathbf{sk},c)=b]$, for $b\in\{0,1\}$, we then have that

$$\begin{aligned}\mathbf{Pr}[\mathtt{DPRandEnc}(S)\in T] &= p_1\cdot\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=1 \mid S]+p_0\cdot\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=0 \mid S]\\ &\leq \frac{k+1}{k}\left(p_1\cdot\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=1 \mid S']+p_0\cdot\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk},c)=0 \mid S']\right)\\ &= \frac{k+1}{k}\cdot\mathbf{Pr}[\mathtt{DPRandEnc}(S')\in T]\\ &\leq e^{\varepsilon}\cdot\mathbf{Pr}[\mathtt{DPRandEnc}(S')\in T]\end{aligned}$$

where the final inequality follows from taking $k = 1/\varepsilon$ and observing $1 + x \leq e^x$.

It remains to argue correctness of `DPRandEnc` when the sample S is drawn from one of the promised distributions. In this case, the input sample S consists of m valid encryptions of the same bit b under $\mathbf{pk}$. Because $\mathsf{Rerandomize}(\mathbf{pk}, c)$ is plaintext-preserving, the probability that $\Pi_{RandEnc}$ is incorrect given S, i.e., outputs a ciphertext encrypting $\neg b$, is exactly the probability that one of the inserted ciphertexts encrypting $\neg b$ is chosen for rerandomization. This happens with probability $\frac{k}{m+2k}$, so taking $m > k/\beta = 1/(\varepsilon\beta)$ ensures $\Pi_{RandEnc}$ is correct except with probability β. □

Cryptographic Adversary from Replicable Algorithm for $\Pi_{RandEnc}$

In this subsection, we show that if there exists a replicable polytime algorithm, $\mathscr{B}$, for $\Pi_{RandEnc}$, instantiated with a randomizable encryption scheme $\mathscr{E}$, then there exists an adversary breaking the security guarantee of $\mathscr{E}$. To break security, the adversary must be able to distinguish whether a ciphertext c encrypts a 1 or a 0 with probability noticeably better than a coin flip.

The high level idea is as follows. The adversary can first use the ciphertext rerandomization procedure to generate a dataset of ciphertexts encrypting the same bit as c. It can then generate a dataset of ciphertexts encrypting 0 by encrypting 0 under the public key and rerandomizing the resulting ciphertext. The adversary will then invoke $\mathscr{B}$ on both datasets, fixing the same randomness for both invocations. If the outputs of both invocations are equal, the adversary will guess that c encrypts a 0, and guess c encrypts 1 otherwise.

Because rerandomization is perfect and $\mathscr{B}$ is a replicable algorithm for $\Pi_{RandEnc}$, if c encrypts a 0, $\mathscr{B}$ will with high probability produce the same output ciphertext for both invocations. If c encrypts a 1, $\mathscr{B}$ will can only output the same ciphertext for both invocations if one of the two invocations is incorrect, and so with good probability, the two outputs will differ. This implies the adversary will have good distinguishing probability, breaking the security of the underlying cryptosystem.

Algorithm 28. Adversary $\mathscr{A}$, cryptographic adversary given a replicable algorithm for $\Pi_{RandEnc}$

```
Result: Outputs a bit $b'$
Input: public key $\mathbf{pk}$, ciphertext $c$
 Algorithm:
    1. Draw a random string $r$
    2. $c_0 \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)$
    3. Generate a set $S_0$ of $m$ ciphertexts by running $\mathsf{Rerandomize}(\mathbf{pk}, c_0)$ $m$ times
    4. $c_0 \leftarrow \mathscr{B}(\mathbf{pk}, S_0; r)$
    5. Generate a sample $S$ of $m$ ciphertexts by running $\mathsf{Rerandomize}(\mathbf{pk}, c)$ $m$ times
    6. $c \leftarrow \mathscr{B}(\mathbf{pk}, S; r)$
    7. if $c_0 = c$, then $b' = 0$, otherwise $b' = 1$
return $b'$
```

Lemma 5.4.5. *Let $\mathscr{E}$ be a randomizeable encryption scheme, let $\Pi^{\mathscr{E}}_{RandEnc}$ denote the instantiation of $\Pi_{RandEnc}$ with $\mathscr{E}$. Let $\mathscr{B}$ be a ρ-replicable algorithm for $\Pi^{\mathscr{E}}_{RandEnc}$ with failure probability β, running in time* $\text{poly}(\lambda, \rho, \beta)$*, and with sample complexity $m \in$* $\text{poly}(\lambda, \rho, \beta)$*. Then there exists an adversary $\mathscr{A}$ running in time* $\text{poly}(\lambda, \rho, \beta)$ *such that*

$$\mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)] \geq 1 - 2\beta - \rho.$$

Proof. The adversary $\mathscr{A}(\mathbf{pk}, c)$ outputs 1 whenever $c_0 \neq c$. The distribution from which S_0 is drawn is supported entirely on encryptions of 0 and, conditioned on $c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)$, the distribution from which S is drawn is supported entirely on encryptions of 1. Then $c_0 \neq c$ except when one of the two calls to $\mathscr{B}$ is incorrect, which happens with probability at most 2β. Conditioned on $c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)$,

S_0 and S comprise i.i.d. samples from the same distribution over encryptions of 0. In this case, $c_0 \neq c$ if either call to $\mathscr{B}$ fails to be replicable, which happens with probability at most ρ. Therefore

$$\mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)] \geq 1 - 2\beta - \rho.$$

$\square$

In particular, taking β, ρ to be constant in Lemma 5.4.5 gives an adversary breaking the security of $\mathscr{E}$, yielding the following theorem as a corollary.

Theorem 5.4.6. *Let $\mathscr{E}$ be a randomizeable encryption scheme, and let $\Pi^{\mathscr{E}}_{RandEnc}$ denote the instantiation of $\Pi_{RandEnc}$ with $\mathscr{E}$. Then there does not exist a ρ-replicable algorithm for $\Pi^{\mathscr{E}}_{RandEnc}$ with failure probability β, running in time* $\mathrm{poly}(1/\lambda)$, *for $\rho < 1/4$ and $\beta < 1/8$.*

Proof. If there exists a ρ-replicable algorithm $\mathscr{B}$ for $\Pi^{\mathscr{E}}_{RandEnc}$ with failure probability $\beta < 1/8$ and replicability parameter $\rho < 1/4$ running in time $\mathrm{poly}(\lambda)$, then by Lemma 5.4.5, there exists an adversary $\mathscr{A}$ running in time $\mathrm{poly}(\lambda)$ such that

$$\mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathscr{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)] \geq 1/2 > \mathrm{negl}(\lambda),$$

and therefore $\mathscr{A}$ breaks the security of $\mathscr{E}$. $\square$

Instantiating $\Pi_{RandEnc}$ with the Goldwasser-Micali Cryptosystem

Here we recall the high-level structure of the Goldwasser-Micali public-key cryptosystem, introduced in [GM82]. The security of the cryptosystem relies on the hardness of deciding quadratic residuosity for integers modulo a semiprime N. Informally, encryptions of 0 are quadratic residues modulo N, while encryptions of 1 are non-residues. Because multiplying an integer c by a quadratic residue modulo N preserves quadratic residuosity of c, Goldwasser-Micali ciphertexts can be efficiently rerandomized with only a public key. The rerandomization procedure will pick a quadratic residue r^2 uniformly at random, and output its product with the given ciphertext modulo N.

Definition 5.4.7 (Goldwasser-Micali Cryptosystem ([GM82])). *The Goldwasser-Micali cryptosystem is defined over a plaintext message space $\mathscr{M} = \{0,1\}$ and ciphertext space $\mathscr{C} = \mathbb{Z}_N^*$, for N a semiprime. The cryptosystem comprises the following routines.*

- $\mathsf{KeyGen}(\lambda)$*: Sample p,q distinct primes of bit-length $O(\lambda)$ and let $N = pq$. Choose x to be a quadratic non-residue modulo N with Jacobi symbol $\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = -1$. Let $\mathbf{sk} = (p,q)$, $\mathbf{pk} = (N,x)$, and output $(\mathbf{sk},\mathbf{pk})$.*

- $\mathsf{Enc}(\mathbf{pk},b)$*: To encrypt a bit $b \in \{0,1\}$, sample $u \leftarrow_{\mathscr{U}} \mathbb{Z}_N^*$ and output $u^2x^b \mod N$.*

- $\mathsf{Dec}(\mathbf{sk},c)$*: To decrypt a ciphertext c, output $\perp$ if $\gcd(c,N) \neq 1$, 1 if c is not a quadratic residue modulo N and 0 otherwise.*

We now show that the Goldwasser-Micali cryptosystem satisfies the strong rerandomization property described above. We define the verification procedure $\mathscr{V}(\mathbf{pk},c)$ to output 1 if $\gcd(c,N) = 1$ and 0 otherwise. We define $\mathsf{Rerandomize}(\mathbf{pk},c)$ to be the procedure that samples r uniformly at random from $\mathbb{Z}_N^*$ and outputs $(\mathbf{pk}, r^2c \mod N)$.

Lemma 5.4.8. *The Goldwasser-Micali cryptosystem is a rerandomizeable encryption scheme (definition 5.4.2), for the rerandomization procedure described above.*

Proof. Because $\mathsf{Dec}(\mathbf{sk},c) = \perp$ if and only if $\gcd(c,N) \neq 1$, and $\mathscr{V}(\mathbf{pk},c) = 0$ if and only if $\gcd(c,N) \neq 1$, $\mathscr{V}$ satisfies the requirement of definition 5.4.2. The rerandomization procedure multiplies a ciphertext c by a random quadratic residue modulo N, and therefore preserves quadratic residuosity of c. This in turn preserves the plaintext message encrypted by c, and so $\mathsf{Rerandomize}$ satisfies $\mathsf{Dec}(\mathbf{sk},\mathsf{Rerandomize}(\mathbf{pk},c)) = \mathsf{Dec}(\mathbf{sk},c)$.

Next, we will show that $d_{TV}(\mathsf{Rerandomize}(\mathbf{pk}, c), \mathsf{Rerandomize}(\mathbf{pk}, c')) = 0$ for all c, c' such that $\mathsf{Dec}(\mathbf{sk}, c) = \mathsf{Dec}(\mathbf{sk}, c')$. Let $b \in \{0,1\}$, $c = u^2x^b \mod N$ and $c' = v^2x^b \mod N$ be honest encryptions of b under $\mathbf{pk} = (N, x)$. It follows that

$$\begin{aligned}
\mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk}, c) = a] &= \mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk}, u^2x^b \mod N) = a] \\
&= \mathbf{Pr}[r^2u^2x^b = a \mod N] \\
&= \mathbf{Pr}[r^2 = u^{-2}x^{-b}a \mod N] \\
&= \mathbf{Pr}[(u^{-1}vr)^2 = u^{-2}x^{-b}a \mod N] \\
&= \mathbf{Pr}[r^2v^2x^b = a \mod N] \\
&= \mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk}, c') = a],
\end{aligned}$$

where the fourth equality follows from r being chosen uniformly at random from $\mathbb{Z}_N^*$. □

5.4.2 Correlated Sampling via One-Way Function Inverters

As we saw in Section 5.3.3, correlated sampling gives a generic way for converting a perfectly generalizing algorithm into a replicable one. In this section, we show that the existence of efficient one-way function inverters implies the ability to efficiently perform correlated sampling on arbitrary distributions over $\{0,1\}^n$. Specifically, we show that if there are no non-uniform one-way functions, then there is polynomial time implicit correlated sampling.

Theorem 5.4.9. *Assuming uniform one-way function inverters exist (Definition 5.4.11), Algorithm* `CorrSamp` *is an (m, n, ν)-implicit correlated sampling algorithm that runs in time polynomial in m, n, and $1/\nu$.*

Proof. The distributional accuracy property is shown in the proof of Lemma 5.4.14. The correlated sampling property is shown in the proof of Lemma 5.4.16. The runtime of `CorrSamp` is shown in the proof of Lemma 5.4.17. □

Relevant Definitions

We model samplable distributions by considering the distribution induced by giving random inputs to circuits. Furthermore, we allow for a distributional error parameter ν, giving some slack in the correctness of a correlated sampler.

Definition 5.4.10 (Implicit Correlated Sampling Algorithm)**.** *Let $m,n \in \mathbb{Z}^+$, and let $C : \{0,1\}^m \to \{0,1\}^n$ denote a circuit. Let distributional error parameter $\nu > 0$. $\mathscr{B}(C,\nu;r)$ is an (m,n,ν)-*implicit correlated sampling algorithm *if the following conditions hold:*

1. ***Inputs/Outputs:** $\mathscr{B}$ takes as input a circuit $C : \{0,1\}^m \to \{0,1\}^n$, a distributional error parameter ν, and a random string r. $\mathscr{B}$ outputs a string in $\{0,1\}^n$.*

2. ***ν-distributional accuracy:** For all circuits $C : \{0,1\}^m \to \{0,1\}^n$, the distributions D_C and $D_{\mathscr{B}(C,\nu)}$ satisfy $d_{\mathrm{TV}}(D_C, D_{\mathscr{B}(C,\nu)}) \leq O(\nu)$.*

   Here, D_C denotes the distribution over $\{0,1\}^n$ induced by querying $C(r)$ on uniformly random inputs r, i.e., probability density function $p_{D_C}(x) = \mathbf{Pr}_{r\sim U^m}[C(r) = x]$. Similarly, $D_{\mathscr{B}(C,\nu)}$ denotes the distribution over $\{0,1\}^n$ induced by querying $\mathscr{B}(C,\nu;r)$ with uniformly random strings r.

3. ***Correlated sampling:** For all pairs of circuits $C_1, C_2 : \{0,1\}^m \to \{0,1\}^n$, $\mathbf{Pr}_r[\mathscr{B}(C_1,\nu;r) \neq \mathscr{B}(C_2,\nu;r)] \in O(d_{\mathrm{TV}}(D_{C_1}, D_{C_2}) + \nu)$.*

We assume we can invert any one-way function on almost all inputs. Specifically, we assume that there is no non-uniform one-way function family, so that there is a uniform way of inverting any circuit computing a function via a polynomial-time inverter.

Definition 5.4.11 (Uniform One-Way Function Inverters)**.** *Let $\nu' > 0$. $\mathscr{I}_{\nu'}(C,y)$ is a* uniform one-way function inverter with error ν' *if $\mathscr{I}$ runs in randomized polynomial time in m, n, and $1/\nu'$ and if, for any circuit $C : \{0,1\}^m \to \{0,1\}^n$, $\mathbf{Pr}_{r'\sim\{0,1\}^m}[C(\mathscr{I}_{\nu'}(C, C(r'))) = C(r')] \geq 1 - \nu'$.*

In this argument, we will choose ν' to be inverse polynomially small in m, n, and $1/\nu$. In addition, we assume that $C(r)$ can be efficiently computed.[8] Thus, we can check if and when the inverter succeeds. For notational convenience, we say that the inverter $\mathscr{I}$ returns "$\perp$" if it does not succeed.

Our correlated sampler randomly samples from pairwise-independent hash families in its subroutines.

Definition 5.4.12 (Pairwise-Independent Hash Family)**.** *A family of Boolean functions* $\mathscr{H} = \{H | H : \{0,1\}^m \to \{0,1\}^n\}$ *is* pairwise-independent *if, for all* $r_1 \neq r_2 \in \{0,1\}^m$ *and* $x_1, x_2 \in \{0,1\}^n$, $\mathbf{Pr}_{H \in \mathscr{H}}[H(r_1) = x_1 \wedge H(r_2) = x_2] = 2^{-2n}$.

Algorithm Overview

A correlated sampling algorithm $\mathscr{B}$ accomplishes two goals. First, $\mathscr{B}(C, \nu; r)$ needs to accurately sample from the distribution D_C. Second, $\mathscr{B}$ must convert a random string r into the same output when run on distributionally close circuits C_1 and C_2, with high probability. In other words, $\mathscr{B}$ must choose a consistent way to map random strings r to elements in the support of D_C.

For intuition, consider a restricted case of correlated sampling problems in which the distributions D_C induced by random inputs to circuits $C : \{0,1\}^m \to \{0,1\}^n$ are promised to be uniformly supported on 2^ℓ elements $x \in \{0,1\}^n$ for some fixed ℓ. Let k be a small slack parameter, and consider the following sampler:

1. Draw a random hash function $H : \{0,1\}^n \to \{0,1\}^{\ell+k}$
2. Draw a random string $u \in \{0,1\}^{\ell+k}$
3. Run the inverter on u: $r = \mathscr{I}_{\nu'}(C \circ H, u)$
4. If $r = \perp$ (i.e. u is not in the support of $C \circ H$), repeat. Else return $y^* = C(r)$

$$\begin{tikzcd}
\begin{array}{c}\textbf{Step 3.}\\ r = \mathscr{I}_{v'}(C \circ H, u)\end{array} & \begin{array}{c}\textbf{Step 4.}\\ y^* = C(r)\end{array} & \begin{array}{c}\textbf{Step 1.}\\ \text{Draw } H\end{array} \quad \begin{array}{c}\textbf{Step 2.}\\ u \sim \mathscr{U}(\{0,1\}^{\ell+k})\end{array} \\
\begin{array}{c}\text{Randomness}\\ \{0,1\}^m\end{array} \arrow[r, "C"] & \begin{array}{c}\text{supp}(D_C)\\ \{0,1\}^n\end{array} \arrow[r, "H"] & \begin{array}{c}\text{Hash range}\\ \{0,1\}^{\ell+k}\end{array} \arrow[ll, bend left, "\mathscr{I}_{v'}(C \circ H, \cdot)"]
\end{tikzcd}$$

Figure 5.2. High-level structure of correlated sampler CorrSamp for uniform D_C

The high level idea is that k can be chosen large enough such that $\text{supp}(D_C)$ has few collisions with good probability (for random H), but small enough s.t. 2^k (and therefore the runtime) remains polynomial in the relevant parameters. Assuming no collisions occur, it is easy to see this process is a correlated sampler since each element in the support of D_C is sampled uniformly at random, and moreover applied to distinct circuits C_1 and C_2, y^* only differs if the sampler hits a hash value u that contains an element in the symmetric difference $\text{supp}(C_1)\Delta\text{supp}(C_2)$. Since there are no collisions, this occurs exactly with probability $d_{TV}(D_{C_1}, D_{C_2})$ as desired.

Moving to the general case, our algorithm CorrSamp applies this idea as follows. CorrSamp divides the distribution D_C into "levels" ℓ, such that each level contains elements in the support with probability density near $2^{-\ell}$ (specifically, those in the range $(2^{-\ell-1}, 2^{-\ell}]$). We now pick a level uniformly at random, and hope to apply the above process. Intuitively, the main challenge is that given the output y^*, we need to ensure y^* actually belongs at level ℓ. This is done through the introduction of a *second hash function* $H_2 : \{0,1\}^m \to \{0,1\}^{m-\ell+k}$. In particular, fixing u and y^* as in the simplified variant, we wish to estimate $|C^{-1}(y^*)| = |(H \circ C)^{-1}(u)|$ (assuming no collisions).

[8]Note that if there is no such efficient circuit C that produces a sample from a distribution D_C (on a uniformly random input), then D_C is hard to sample from, and designing an efficient correlated sampling algorithm whose marginal distribution is D_C (when given circuit C) is hopeless.

To do this, we call the inverter on the concatenated function $F_{C,l,H,H_2}(r) \overset{\text{def}}{=} H(C(r)) \,||\, H_2(r)$[9] on many pairs of the form $(u||v)$, where $v \in \{0,1\}^{m-\ell+k}$ is chosen uniformly at random. Since we have fixed u, the inverter can only succeed on this call when $v = H_2(r)$ for some $r \in (C \circ H)^{-1}(u)$. Since v is chosen uniformly at random, the success probability of the inverter is then directly proportional to the density of y^*, allowing us to determine whether or not y^* is in level ℓ with high probability.[10] We can then return y^* if it is in the chosen level, and repeat the process from the beginning if not. Because we have chosen u uniformly at random from $\{0,1\}^{\ell+k}$, for any x it holds that $x = H^{-1}(u)$ with probability $2^{-\ell-k}$ (assuming no collisions). Then this approach allows us to sample uniformly from level ℓ, where every x in level ℓ is output with probability proportional to $2^{-\ell}$. Note that since the true density may be a constant factor away from $2^{-\ell}$, this is not yet quite enough to achieve our true target distributional accuracy—we will address this detail in the next section.

Algorithm Description and Pseudocode

We now give pseudocode for the `CorrSamp` algorithm and its subroutines, in addition to a more detailed description. The main algorithm `CorrSamp` takes as input a circuit C and an error parameter ν. As described in Section 5.4.2, at each iteration of the main loop, `CorrSamp` picks a level ℓ uniformly at random, and then tries to sample an element that has probability roughly $2^{-\ell}$ under D_C. In addition to drawing a hash function H_1 with range $\{0,1\}^{\ell+k}$, and an element u from that range, it will also sample a random threshold parameter $\beta \in (1,2]$, and invoke the subroutine $\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1,u)$ with these parameters and inputs. We will properly motivate this new parameter β shortly, but looking ahead, it will help us avoid the distributional accuracy issues present in Section 5.4.2.

In the pseudocode for `CorrSamp` (Algorithm 29), k is chosen to be large enough that we will be able to avoid problematic collisions for all hash functions with high probability, but small enough

[9] We use '$||$' to denote concatenation of strings.

[10] Of course this only holds assuming few collisions. To handle the general case, we actually draw a new H_2 with every choice v to ensure this holds across all rounds.

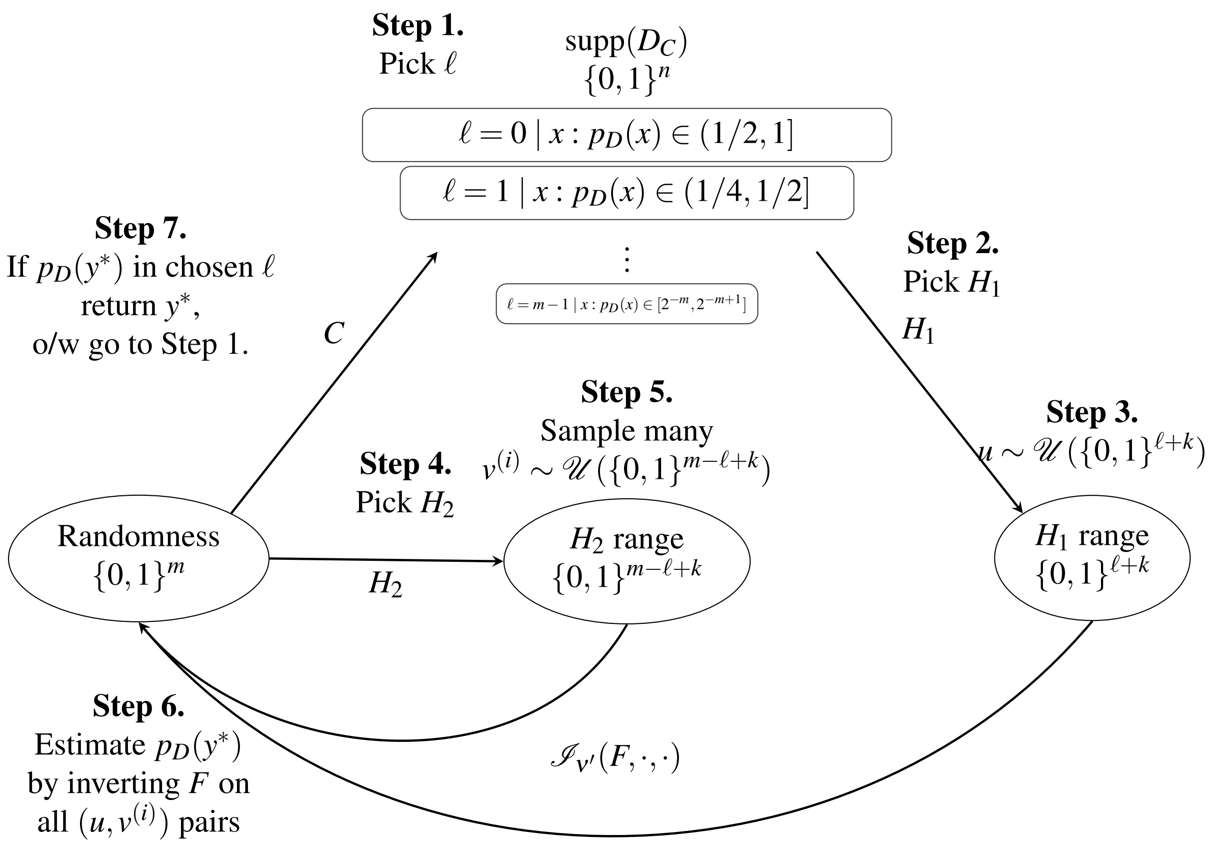

Figure 5.3. High-level structure of correlated sampler `CorrSamp` for general D_C

to ensure polynomial runtime. The value T_1 is chosen to be large enough that `CorrSamp` returns $x \neq \perp$ with high probability, but also small enough such that we can guarantee certain simplifying assumptions will hold with high probability across all rounds of `CorrSamp`.

Algorithm 29. `CorrSamp`$(C, \nu; r)$, correlated sampling algorithm
Input: Circuit $C : \{0,1\}^m \to \{0,1\}^n$, distributional error parameter ν, and random string r
Output: An element $x \in \{0,1\}^n$

```
$k \leftarrow \Theta(\log(m) + \log(1/\nu))$
$T_1 \leftarrow \Theta(mk2^k \log(1/\nu))$
**for** $t = 1$ to $t = T_1$ **do**
    $\beta \leftarrow_r [1,2]$
    $\ell \leftarrow_r \{0,1,\ldots,m\}$
    $H_1 \leftarrow_r$ pairwise independent hash function from $n$ bits to $\ell + k$ bits
    $u \leftarrow_r \{0,1\}^{\ell+k}$
    $x \leftarrow \texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u; r)$
    **if** $x \neq \perp$ **then**
        **return** $x$
**return** $\perp$
```

The subroutine $\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u; r)$ follows the approach described in Section 5.4.2, estimating the probability of $y^* = H_1^{-1}(u)$ under D_C by drawing many pairs (H_2, v) of hash functions with elements from their range, and invoking the subroutine $\texttt{HashCheck}_{C,\nu,\ell,H_1,u}(H_2, v)$ to invert $H(C(r))||H_2(r)$ on each (u, v). This procedure approximates the density of random strings r mapped to y^* by C. If y^* is in or nearly in level ℓ, then `ElemFind` will obtain an estimate $\widehat{q}(y^*)$ that is close to $p_D(y^*)2^{\ell-k}$. It will then return y^* only if $\frac{\beta}{2}2^{-k} < \widehat{q}(y^*) \leq \beta 2^{-k}$, i.e. roughly when $\beta 2^{-\ell-1} < p_D(y^*) \leq \beta 2^{-\ell}$.

We now reach the core reason for choosing our threshold β randomly. As in Section 5.4.2, if we did fix β, then any y^* at level ℓ will be sampled whenever $y^* \in H_1^{-1}(u)$, which happens with probability proportional to $2^{-\ell}$. Since this occurs for any $p_D(y^*) \in [\beta 2^{-\ell-1}, \beta 2^{-\ell}]$, this is not a good enough estimate. The key is to observe that choosing β randomly allows us to avoid this kind of uniform sampling over any particular level. Instead we sample from "fuzzy" levels, where the choice of β shifts the boundaries while maintaining that the fuzzy levels partition $[0,1]$. In slightly more detail, observe that for any y^* there is some j such that $p_D(y^*) = \alpha 2^{-j-1} + (1-\alpha)2^{-j}$ for $\alpha \in [0,1]$. This means we want y^* to belong to 'level' $(j+1)$ with probability α, and to level j with probability $(1-\alpha)$. We will show in Lemma 5.4.14 that choosing β uniformly at random exactly achieves this.

In the pseudocode for `ElemFind` (Algorithm 30), k is chosen to balance the same constraints we have described for `CorrSamp`. The value for T_2, the number of hash function and element pairs (H_2, v) used to estimate the probability $p_D(y^*)$, is large enough to ensure a good empirical estimate for y^*, but small enough to ensure `ElemFind` still runs in time polynomial in m, n, and $1/\nu$.

Algorithm 30. $\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u; r)$, find an x to return in "fuzzy" level l
(Explicit) Input: Hash function $H_1 : \{0,1\}^n \to \{0,1\}^{\ell+k}$, string $u \in \{0,1\}^{\ell+k}$, random string r
(Implicit) Input: Circuit $C : \{0,1\}^m \to \{0,1\}^n$, distributional error parameter ν, integer $\ell \in \{0,1,\ldots,m\}$, interval rescaling parameter β
Output: String $x \in \{0,1\}^n$ and probability $q_x \in ((\beta/2)2^{-k}, \beta 2^{-k}]$.

```
$k \leftarrow \Theta(\log(m) + \log(1/\nu))$
$T_2 \leftarrow \Theta(\nu^{-2}\log(1/\nu)T_1^{-1}\log(T_1^{-1}))$
**for** $i = 1$ to $i = T_2$ **do**
    $H_2^i \leftarrow_r$ pairwise independent hash function from $m$ bits to $m - \ell + k$ bits
    $v^i \leftarrow_r \{0,1\}^{m-\ell+k}$
    Run $\texttt{HashCheck}_{C,\nu,\ell,H_1,u}(H_2^i, v^i; r)$
Let $\widehat{q}_x$ denote the fraction of times $x$ was returned by HashCheck
**if** $\exists$ unique $x$ s.t. $x \neq \perp$ and $\widehat{q}_x \in ((\beta/2)2^{-k}, \beta 2^{-k}]$ **then**
    **return** $x$
**else**
    **return** $\perp$
```

Algorithm 31. $\texttt{HashCheck}_{C,\nu,\ell,H_1,u}(H_2,v;r)$, use inverter to check for valid inverses of H_1 and H_2
(Explicit) Input: Hash function (circuit) $H_2 : \{0,1\}^m \to \{0,1\}^{m-\ell+k}$, string $v \in \{0,1\}^{m-\ell+k}$, and random string r
(Implicit) Input: Circuit $C : \{0,1\}^m \to \{0,1\}^n$, distributional error parameter ν, integer $\ell \in \{0,1,\dots,m\}$, hash function $H_1 : \{0,1\}^n \to \{0,1\}^{\ell+k}$ and string $u \in \{0,1\}^{\ell+k}$
Output: String $x \in \{0,1\}^n$

```
Define circuit $F_{C,\ell,H_1,H_2}(r') = H_1(C(r')) \,\|\, H_2(r')$
$\nu' \leftarrow$ inverse polynomial quantity in $m,n,1/\nu$.
$r' \leftarrow \mathscr{I}_{\nu'}(F_{C,\ell,H_1,H_2},(u \,\|\, v);r)$
if $r' = \perp$ then
    return $\perp$
else
    return $C(r')$
```

Analysis – Structure and Simplifying Assumptions

In this section, we analyze the CorrSamp algorithm. Before proceeding with the analysis, we first introduce several simplifying assumptions below we will use throughout. In Section 5.4.2, we analyze the distributional accuracy of CorrSamp, showing that its distribution over outputs is close to the target distribution. In Section 5.4.2, we analyze the success probability of CorrSamp as a correlated sampler, showing that for two circuits C_1 and C_2, $\texttt{CorrSamp}(C_1,\nu;r) = \texttt{CorrSamp}(C_2,\nu;r)$ except with probability proportional to $d_{\mathrm{TV}}(D_{C_1},D_{C_2})$. In Section 5.4.2, we show that CorrSamp runs in time polynomial in the m, n, and $1/\nu$. Finally, in Section 5.4.2 we show that our simplifying assumptions hold for the entire execution of CorrSamp, except with high probability. Thus, all statements about the behavior of CorrSamp under the ideal conditions will still hold without these assumptions, except with probability $O(\nu)$.

Definition 5.4.13 (Ideal Conditions)**.** *We collectively refer to the following as the* ideal conditions.

- *The inverter $\mathscr{I}_{\nu'}$ never fails: for all x and r on which* `CorrSamp` *invokes $\mathscr{I}_{\nu'}(x;r)$, $\mathscr{I}_{\nu'}(x;r) \neq \perp$.*
- *For every ℓ, there is at most one $y^* \in H_1^{-1}(u)$ s.t.*

$$p_D(y^*) \in [2^{-\ell-4}, 2^{-\ell+4}]$$

- `ElemFind` *always returns y^* or '$\perp$'. Furthermore, for y^* returned by* `ElemFind`*, we have the stronger condition that $p_D(y^*) \in [2^{-\ell-2}, 2^{-\ell+2}]$.*
- *The empirical estimate of $\widehat{q}(y^*)$ is good:*

$$\hat{q}(y^*) \in (1 \pm O(\nu))\, p_D(y^*) 2^{\ell-k}$$

Analysis – Distributional Accuracy.

In this section, we analyze the distributional accuracy of `CorrSamp`. Denote by $D_{\texttt{CorrSamp}}$ the distribution over outputs of $\texttt{CorrSamp}(C,\nu)$. We show that $d_{\text{TV}}(D_{\texttt{CorrSamp}}, D_C) \leq O(\nu)$, assuming the ideal conditions of Definition 5.4.13.

Lemma 5.4.14 (Distributional Accuracy of `CorrSamp`)**.** *For all circuits $C : \{0,1\}^m \to \{0,1\}^n$, the distributions D_C and $D_{\texttt{CorrSamp}}$ satisfy $d_{\text{TV}}(D_C, D_{\texttt{CorrSamp}}) \leq O(\nu)$, assuming the ideal conditions of Definition 5.4.13 hold for all rounds of* `CorrSamp`*. Here, D_C and $D_{\texttt{CorrSamp}}$ denote the distributions over $\{0,1\}^n$ induced by querying $C(r)$ and $\texttt{CorrSamp}(C,\nu;r)$ respectively with uniformly random strings r.*

We first prove the following useful lemma, bounding the probability that any $x \in \text{supp}(D_C)$ is returned in a single round.

Lemma 5.4.15. *Fix an $x \in supp(D_C)$. Then $\mathbf{Pr}_{H_1,u,\ell,\beta}[\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1,u) = x] \in \frac{p_D(x)(1+O(\nu))}{(m+1)2^k}$.*

Proof. For any ℓ, we have that $\mathbf{Pr}_{H_1,u}[x = H_1^{-1}(u)] = 2^{-\ell-k}$. Conditioned on x being selected by H_1 and u, by construction, $\texttt{ElemFind}_{C,\nu,\ell,\beta}$ returns x whenever $\widehat{q}(x) \in (\frac{\beta}{2}2^{-k}, \beta 2^{-k}]$. Rewriting $p_D(x) = \gamma 2^{-j}$ for $\gamma \in [1/2, 1]$, we observe that x will only ever have non-zero probability of being returned by $\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u)$ when $j-1 \leq \ell \leq j+2$, from the assumptions of Definition 5.4.13. Since we have $\widehat{q}(x) \in (1 \pm O(\nu))p_D(x)2^{\ell-k}$ for any ℓ in this range, it follows that for these ℓ,

$$\begin{aligned}\mathbf{Pr}_\beta\left[\widehat{q}(x) \in (\tfrac{\beta}{2}2^{-k}, \beta 2^{-k}]\right] &\in \mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell(1 \pm O(\nu)) \leq \beta] \\ &\in \mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leq \beta] \pm O(\nu)p(x)2^\ell.\end{aligned}$$

Recalling that ℓ is chosen uniformly at random from $\{0, \ldots, m\}$, we can then write the probability that ElemFind returns x as

$$\begin{aligned}\mathbf{Pr}_{H_1,u,\ell,\beta}[\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1,u) = x] &\in \sum_{\ell \in [j-1, j+2]} \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^\ell \leq \beta] \pm O(\nu)p_D(x)2^\ell}{(m+1)2^{\ell+k}} \\ &\in \left(\sum_{\ell \in [j-1, j+2]} \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^\ell \leq \beta]}{(m+1)2^{\ell+k}}\right) \pm \frac{O(\nu)p_D(x)}{(m+1)2^k}.\end{aligned}$$

Observing that $\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^\ell \le \beta] = 0$ except for $\ell \in \{j, j+1\}$, we can simplify the series:

$$
\begin{aligned}
\sum_{\ell \in [j-1, j+2]} \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^\ell \le \beta]}{(m+1)2^{\ell+k}} &= \sum_{\ell \in [j, j+1]} \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^\ell \le \beta]}{(m+1)2^{\ell+k}} \\
&= \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^j \le \beta]}{(m+1)2^{j+k}} + \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < p_D(x)2^{j+1} \le \beta]}{(m+1)2^{j+k+1}} \\
&= \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < \gamma \le \beta]}{(m+1)2^{j+k}} + \frac{\mathbf{Pr}_\beta[\frac{\beta}{2} < 2\gamma \le \beta]}{(m+1)2^{j+k+1}} \\
&= \frac{\mathbf{Pr}_\beta[\beta < 2\gamma]}{(m+1)2^{j+k}} + \frac{\mathbf{Pr}_\beta[\beta \ge 2\gamma]}{(m+1)2^{j+k+1}} \\
&= \frac{2\gamma - 1}{(m+1)2^{j+k}} + \frac{2 - 2\gamma}{(m+1)2^{j+k+1}} \\
&= \frac{2\gamma - 1 + 1 - \gamma}{(m+1)2^{j+k}} \\
&= \frac{p_D(x)}{(m+1)2^k}.
\end{aligned}
$$

Plugging back into the probability that `ElemFind` returns x, we have

$$
\mathbf{Pr}_{H_1, u, \ell, \beta}[\mathtt{ElemFind}_{C, \nu, \ell, \beta}(H_1, u) = x] \in \frac{p_D(x)(1 + O(\nu))}{(m+1)2^k}.
$$

□

Finally, we can show that the output distribution of `CorrSamp` and that of circuit C are $O(\nu)$-close in variation distance.

Proof. Proof of Lemma 5.4.14:

From Lemma 5.4.15, we have that in each round, x is returned by `ElemFind` with probability $\frac{p_D(x)(1 \pm O(\nu))}{(m+1)2^k}$.

Summing over all x, the probability that `CorrSamp` terminates in any individual round is in the range $\frac{1 \pm O(\nu)}{(m+1)2^k}$. So, conditioned on a round of `CorrSamp` returning, the algorithm returns x with probability in $(1 \pm O(\nu))p_D(x)$. Finally, `CorrSamp` does not return by the final T_1'th round with probability at most $O(\nu)$, by the choice of T_1. Altogether, this implies $d_{\mathrm{TV}}(D_C, D_{\texttt{CorrSamp}}) \in O(\nu)$ as desired. $\square$

Analysis — Correlated Sampling.

Next, we show that `CorrSamp` satisfies the correlated sampling requirement of Definition 5.4.10. To simplify notation in this subsection, we will denote the probability of an element x under D_{C_1} and D_{C_2} by $p_1(x)$ and $p_2(x)$ respectively.

Lemma 5.4.16 (Correlated Sampling of `CorrSamp`)**.** *For all circuits $C_1, C_2 : \{0,1\}^m \to \{0,1\}^n$, assuming the ideal conditions hold for all rounds of* $\texttt{CorrSamp}(C_1, \nu; r)$ *and* $\texttt{CorrSamp}(C_2, \nu; r)$*,*

$$\mathbf{Pr}_r[\texttt{CorrSamp}(C_1, \nu; r) \neq \texttt{CorrSamp}(C_2, \nu; r)] \leq O(d_{\mathrm{TV}}(D_{C_1}, D_{C_2}) + \nu).$$

Proof. Let E_0 denote the event that $\texttt{CorrSamp}(C_1, \nu; r) \neq \texttt{CorrSamp}(C_2, \nu; r)$, and for simplicity of notation, shorten `ElemFind` to `EF`, and write $\texttt{EF}_{C_j}^{(i)}$ to denote the output of `ElemFind` in the ith round. We start by making some simplifying assumptions. First, observe that since the probability `CorrSamp` returns $\perp$ is always at most $O(\nu)$, we can condition on the fact that both $\texttt{CorrSamp}(C_2, \nu; r) \neq \perp$ and $\texttt{CorrSamp}(C_1, \nu; r) \neq \perp$ without loss of generality. Second, we can assume by symmetry that $\texttt{CorrSamp}(C_1, \nu; r)$ does not return before $\texttt{CorrSamp}(C_2, \nu; r)$ (else relabel C_1 as C_2).

With this in mind, let E_i denote the event that $\texttt{CorrSamp}(C_2, \nu; r)$ returns in round i. The event E_0 can then be bounded by

$$\mathbf{Pr}[E_0] \leq \sum_i \mathbf{Pr}[E_i] \mathbf{Pr}_{\beta,\ell}[\texttt{EF}_{C_1}^{(i)} \neq \texttt{EF}_{C_2}^{(i)} \mid E_i].$$

To bound the probability of this inner event, observe that under our assumptions, this occurs exactly when $x = \texttt{EF}_{C_2}^{(i)}$, but for the empirical estimate computed by $\texttt{EF}_{C_1}^{(i)}$, $\widehat{q}_1(x) \notin [\frac{\beta}{2} 2^{-k}, \beta 2^{-k}]$. We will show conditioned on any value of x and ℓ, this occurs with probability at most:

$$\mathbf{Pr}_\beta[\mathtt{EF}_{C_1}^{(i)} \neq x \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x, \ell] \leq O(2^\ell |p_2(x) - p_1(x)| + \nu). \tag{5.9}$$

In this case, we can bound E_0 by conditioning further on x and ℓ as:

$$\begin{aligned}
\mathbf{Pr}_r[E_0] &\leq \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \mathbf{Pr}_{\beta,\ell}[\mathtt{EF}_{C_1}^{(i)} \neq \mathtt{EF}_{C_2}^{(i)} \mid E_i] \\
&\leq \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} \mathbf{Pr}_{\beta,\ell}[\mathtt{EF}_{C_1}^{(i)} \neq \mathtt{EF}_{C_2}^{(i)} \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x] \mathbf{Pr}_{\beta,\ell}[\mathtt{EF}_{C_2}^{(i)} = x | E_i] \\
&\leq O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} \mathbf{Pr}_{\beta,\ell}[\mathtt{EF}_{C_1}^{(i)} \neq \mathtt{EF}_{C_2}^{(i)} \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x] p_2(x) \\
&\leq O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} p_2(x) \sum_{\ell} \mathbf{Pr}[\ell \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x] \mathbf{Pr}_\beta[\mathtt{EF}_{C_1}^{(i)} \neq \mathtt{EF}_{C_2}^{(i)} \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x, \ell] \\
&\leq O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} p_2(x) \sum_{\ell} \mathbf{Pr}[\ell \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x] O(2^\ell |p_2(x) - p_1(x)|).
\end{aligned}$$

Under the ideal conditions, the posterior of ℓ is 0 unless $p_2(x) \in [2^{-\ell-2}, 2^{-\ell+2}]$, so altogether:

$$\begin{aligned}
\mathbf{Pr}_r[E_0] &\leq O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} p_2(x) \sum_{\ell} \mathbf{Pr}[\ell \mid E_i, \mathtt{EF}_{C_2}^{(i)} = x] O\left(\frac{1}{p_2(x)} |p_2(x) - p_1(x)|\right) \\
&\leq O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} O(|p_2(x) - p_1(x)|) \\
&\leq O(d_{TV}(D_{C_1}, D_{C_2}) + \nu)
\end{aligned}$$

as desired.

It is therefore left to prove Equation (5.9), which we analyze the probability by splitting into two cases based on $p_1(x)$:

1. $p_2(x)/4 \leq p_1(x) \leq 4p_2(x)$

2. $p_1(x) < p_2(x)/4$ or $p_1(x) > 4p_2(x)$

Case 1:

In this case, because we are assuming the ideal conditions of Definition 5.4.13 and have conditioned on $\texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r) = x$ for some x in this round, it must be the case that for this x we have:

- $p_2(x) \in [2^{-\ell-2}, 2^{-\ell+2}]$
- $p_1(x) \in [2^{-\ell-4}, 2^{-\ell+4}]$ (from our assumption that $p_2(x)/4 \leq p_1(x) \leq 4p_2(x)$)
- $\widehat{q}_2(x) \in (1 \pm O(\nu))p_2(x)2^{\ell-k}$
- $\widehat{q}_1(x) \in (1 \pm O(\nu))p_1(x)2^{\ell-k}$.

From the uniqueness of x satisfying $x = H_1^{-1}(u)$ and $p_1(x) \in [2^{\ell-4}, 2^{-\ell+4}]$, we can assume that $\texttt{ElemFind}_{C_1,\nu,\ell,\beta}(H_1,u;r)$ outputs either x or $\perp$ in this round. Therefore, we can bound the probability $\texttt{ElemFind}_{C_1,\nu,\ell,\beta}(H_1,u;r) \neq \texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r)$ by the probability that $\texttt{ElemFind}_{C_1,\nu,\ell,\beta}(H_1,u;r)$ outputs $\perp$ conditioned on $\texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r)$ returning x. This occurs whenever β is chosen so that either

$$(1+O(\nu))p_1(x)2^{\ell-k} < \frac{\beta 2^{-k}}{2} \leq p_2(x)2^{\ell-k}(1+O(\nu))$$

or

$$p_2(x)2^{\ell-k}(1-O(\nu)) \leq \beta 2^{-k} < p_1(x)2^{\ell-k}(1+O(\nu)).$$

Observing that these cases are mutually exclusive and the first interval is the largest, we consider only that worst case and rearrange to obtain the condition

$$\begin{aligned}\beta &\leq 2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu)(p_2(x)+p_1(x))2^{\ell} \\ &\leq 2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu),\end{aligned}$$

where the last inequality follows from our previously stated bounds $p_1, p_2 \in [2^{-\ell-4}, 2^{-\ell+4}]$.

Since β is chosen uniformly at random from the interval $[1,2]$, it follows that β satisfies the condition above with probability no greater than $2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu)$. Conditioning on $\texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r) = x$ and $p_2(x) \in [2^{-\ell-4}, 2^{-\ell+4}]$, we have

$$\begin{aligned}\mathbf{Pr}_\beta[\texttt{ElemFind}_{C_1,\nu,\ell\beta}(H_1,u;r) = \bot] &\leq \mathbf{Pr}_\beta[\beta \leq 2^{\ell+1}|p_2(x_2) - p_1(x_2)| + O(\nu)] \\ &\leq 2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu).\end{aligned}$$

Case 2:

In this case, we have either $p_1(x) < p_2(x)/4$ or $p_1(x) > 4p_2(x)$, and so $p_2(x) \in O(|p_2(x) - p_1(x)|)$. Conditioning on $\texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r) = x$ in this round also gives us that $p_2(x) = \Theta(2^{-\ell})$, so

$$\begin{aligned}\mathbf{Pr}_\beta[\texttt{ElemFind}_{C_1,\nu,\ell,\beta}(H_1,u;r) \neq x \mid \texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r) = x] &\leq 1 \\ &\leq O(2^\ell p_2(x_2)) \\ &\leq O(2^\ell|p_2(x_2) - p_1(x_2)|).\end{aligned}$$

Then for any value of $p_1(x)$ (either in Case 1 or Case 2) we have that

$$\mathbf{Pr}_\beta[\texttt{ElemFind}_{C_1,\nu,\ell,\beta}(H_1,u;r) \neq x \mid \texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1,u;r) = x] \in O(2^\ell|p_2(x) - p_1(x)| + \nu)$$

as desired. $\square$

Analysis — Runtime

Lemma 5.4.17 (Runtime of `CorrSamp`). *Assuming inverter $\mathscr{I}$ runs in time polynomial in m, n, and $1/\nu$, algorithm* `CorrSamp` *also runs in time polynomial in m, n, and $1/\nu$.*

Proof. Algorithm `CorrSamp` runs for at most $O(mk2^k \log(1/\nu))$ rounds. Parameter k is chosen in

$\Theta(\log(m)+\log(1/\nu))$, so $2^k \in O(\text{poly}(m,1/\nu))$. Randomly sampling a pairwise-independent hash function from n bits to $\ell+k$ bits can be done in $\text{poly}(m,n,1/\nu)$ time.

Each round of `CorrSamp` invokes `ElemFind`. In `ElemFind`, there are $O(\text{poly}(m,1/\nu))$ rounds in which a hash function from m bits to $m-\ell+k$ bits is randomly sampled ($O(\text{poly}(m,1/\nu))$ time). Furthermore, each round contains a call to `HashCheck`, which runs in time $\text{poly}(m,n,1/\nu)$ by the assumption that inverter $\mathscr{I}_{\nu'}$ does as well.

Multiplying these nesting terms together, `CorrSamp` runs in time polynomial in m, n, and ν.

Note that the randomness management, which ensures that the same bits of the random string r are always used across multiple executions of `CorrSamp`, can also be done in time polynomial in m, n, and $1/\nu$. Each algorithm and subroutine has a finite number of randomness calls, and each call can be made using $\text{poly}(m,n,1/\nu)$ random bits. Thus, r can efficiently be canonically proportioned for all uses of randomness in the algorithm. For more details, see Appendix 5.9.1.

Parameter β is chosen uniformly randomly in $[1,2]$ in each loop of `CorrSamp`. Only polynomial in m, n, and $1/\nu$ bits of precision are needed to choose β so that the errors introduced by not using uniformly random values are altogether small relative to distributional error parameter ν. □

Analysis – Removing Assumption of Ideal Conditions

Proposition 5.4.18. *The ideal conditions of Definition 5.4.13 hold across all steps of* `CorrSamp` *with probability at least* $1-O(\nu)$.

We break the proof into its four constituent part.

Lemma 5.4.19 (Inverter Never Fails)**.** *Let $\mathscr{S}$ denote the set of strings $(x;r)$ on which* `CorrSamp` *invokes $\mathscr{I}_{\nu'}(x;r)$. The probability the inverter fails on $\mathscr{S}$ is negligible:*

$$\mathbf{Pr}[\exists (x;r)\in\mathscr{S} : \mathscr{I}_{\nu'}(x;r)=\perp] \leq O(\nu),$$

as long as $\nu' = \mathrm{poly}(\nu^{-1}, m, n)$ *is sufficiently small.*

Proof. Recall our inverter has the following guarantee

$$\mathbf{Pr}_{r' \sim \{0,1\}^m}[C(\mathscr{I}_{\nu'}(C, C(r')); r) = C(r')] \geq 1 - \nu',$$

where r stands for the internal randomness of $\mathscr{I}$ and ν' can be taken to be polynomially small in m, n and ν. It will be enough to take the failure rate $\nu' \leq O(\frac{\nu^2}{T_1 T_2}) = \mathrm{poly}(\nu^{-1}, m, n)$. We will bound the probability such an inverter fails on a random input $(u||v)$.

First, observe that by Markov's inequality, most choices of internal randomness r for the inverter random work for almost all $r' \in \{0,1\}^m$:

$$\mathbf{Pr}_r\left[\mathbf{Pr}_{r'}\left[\mathscr{I}_{\nu'}(C, C(r')); r) \neq C(r')\right] \geq O\left(\frac{\nu}{T_1 T_2}\right)\right] \leq O(\nu).$$

Assume then this event does not occur. Our algorithm only fails if the random choice of $(u||v)$ is hashed to by a string for which $\mathscr{I}_{\nu'}(C, C(r')); r) \neq C(r')$. In the worst case, these bad strings each correspond to unique $(u||v) \in \{0,1\}^{m+2k}$, in which case we have a total of $\frac{\nu}{T_1 T_2} 2^m$ out of 2^{m+2k} bad inputs. Union bounding over the $T_1 T_2$ applications of the inverter in `CorrSamp` gives a total failure probability of $1 - O(\nu)$ as desired. □

To prove the remaining conditions, it will be useful first to bound the number of collisions experienced by our hash functions.

Claim 5.4.20 (Collision Avoidance)**.** *With probability at least* $1 - O(\nu)$*, for all choices of* H_1*, u, and* ℓ *in* `CorrSamp`*:*

1. H_1 *has no relevant collisions:*

$$\forall p_D(x), p_D(x') \in [2^{-\ell-4}, 2^{-\ell+4}] : H_1(x) \neq H_1(x'),$$

2. *For all $x \in H_1^{-1}(u)$, the total number of collisions across choices of H_2 is at most:*

$$\left|\{H_2, (r,r') \in C^{-1}(x) : H_2(r) = H_2(r')\}\right| \leq \delta_x T_2 |C^{-1}(x)|,$$

where $\delta_x = O\left(2^\ell p_D(x) \cdot \frac{mk\log(1/\nu)}{\nu} \cdot 2^{-k}\right)$

Proof. To prove the first condition, observe there are at most $2^{\ell+4}$ elements $x \in \{0,1\}^n$ with measure in the range $p_D(x) \in [2^{-\ell-4}, 2^{-\ell+4}]$. Since our hash function is pairwise-independent, the probability a collision exists in this range is therefore bounded by $\frac{2^{2\ell+8}}{2^{2\ell+2k}} = 2^{4-2k}$. Union bounding over the T_1 choices of H_1, u, and ℓ, a collision still occurs with probability at most $2^{-\Omega(k)} \leq O(\nu)$ as desired.

To prove the second condition, fix $x \in H_1^{-1}(u)$ and observe that by pairwise independence, the expected number of total collisions across all choices of H_2 is at most

$$T_2 \frac{|C^{-1}(x)|^2}{2^{2m-2\ell+2k}}.$$

by linearity of expectation, so by Markov's inequality the probability there are more than

$$T_2 \frac{|C^{-1}(x)|^2}{2^{2m-2\ell+2k}} \cdot \frac{T_1 2^m}{\nu} \leq T_2 |C^{-1}(x)| \cdot \frac{2^\ell p_D(x) mk \log(1/\nu)}{\nu 2^k}$$

total collisions is at most $O(\nu)/(T_1 2^m)$, so union bounding over all choices of H_1, u, and x gives the desired result. □

Lemma 5.4.21 (Uniqueness of y^*)**.** *With probability at least $1 - O(\nu)$ over all choices of H_1 and u, there is at most one element $y^* \in H_1^{-1}(u)$ satisfying*

$$p_D(y^*) \in [2^{-\ell-4}, 2^{-\ell+4}].$$

Proof. This is immediate from the fact that H_1 has no collisions on $[2^{-\ell-4}, 2^{-\ell+4}]$ with high probability. $\square$

Lemma 5.4.22 (Correctness of $\hat{q}$)**.** *With probability at least* $1 - O(\nu)$*, every run of* `ElemFind` *accurately estimates* $p_D(y^*)$ *in the following sense:*

$$\widehat{q}(y^*) \in \left((1 \pm O(\nu))p_D(x)2^{\ell-k}\right)$$

Proof. For intuition, first consider the setting where no collisions occur in any H_2. In this case, observe that the density of the pre-image of x mapped into the range of H_2 is exactly $p_D(x)2^{\ell-k}$ by construction, that is:

$$\frac{|H_2(C^{-1}(y^*))|}{2^{m-\ell+k}} = p_D(x)2^{\ell-k}.$$

As such $\hat{q}(y^*)$ is distributed as a Binomial distribution $Bin(p_D(y^*)2^{\ell-k}, T_2)$, and Chernoff promises that

$$\mathbf{Pr}_{\nu,H_2}[|\hat{q}(x) - p_D(x)2^{\ell-k}| \geq O(\nu)p_D(y^*)2^{\ell-k}] \leq e^{-\Omega(\nu^2 T_2)} \leq \frac{O(\nu)}{T_1}$$

by our choice of T_2. The desired result then follows from union bounding over all choices of H_1 and u.

We now modify this analysis under the assumption that at most

$$C_{col} := O\left(\frac{2^\ell p_D(y^*)mk\log(1/\nu)}{\nu 2^k}\right) \cdot |C^{-1}(y^*)|T_2$$

total collisions occur, which holds across all rounds except with probability $O(\nu)$ (Claim 5.4.20). In this case, since $p_D(y^*) \leq 2^{\ell+4}$, we have

$$C_{col} \leq O(\nu)|C^{-1}(y^*)|T_2$$

for $k = \Theta(\log(m/\nu))$ sufficiently large, and therefore that the expectation of our adjusted binomial trial is close enough to its ideal expectation:

$$\sum_{H_2} \frac{|H_2(C^{-1}(y^*))|}{2^{m-\ell+k}} \in T_2 \cdot \left[(1-O(\nu))p_D(y^*)2^{\ell-k}, p_D(y^*)2^{\ell-k}\right],$$

that the collisions have no asymptotic effect on the original Chernoff bound. □

Lemma 5.4.23 (Correctness of `ElemFind`)**.** *With probability at least* $1-O(\nu)$*, all calls to* `ElemFind` *return* y^* *or '*$\perp$*'. Furthermore, for* y^* *returned by* `ElemFind`*, we have the stronger condition that* $p_D(y^*) \in [2^{-\ell-2}, 2^{-\ell+2}]$.

Proof. For the first claim, it is enough to argue that any $x \in H_1^{-1}(u)$ distinct from y^* satisfies:

$$\hat{q}(x) \notin [(\beta/2)2^{-k}, \beta 2^{-k}].$$

Assuming y^* is unique (Lemma 5.4.21), we have either that $p_D(x) < 2^{-\ell-4}$ or $p_D(x) > 2^{-\ell+4}$. Consider the former. We will show $\hat{q}(x) < 2^{-k-1} \leq (\beta/2)2^{-k}$. Note that since collisions only lower $\hat{q}$, they can be ignored in this setting. By construction, the pre-image of x consists of at most $2^{m-\ell-4}$ strings r' map to x, so $|H_2(C^{-1}(x))|$ is at most a

$$\frac{2^{m-\ell-4}}{2^{m-\ell+k}} = \frac{2^{-k}}{16}$$

fraction of the range of H_2. A Chernoff and Union bound give that all such x have empirical estimates less than $(\beta/2)2^{-k}$ except with probability $2^m e^{-\Omega(2^{-k}T_2)} \leq O(\nu)$.

Finally, consider x satisfying $p_D(x) \geq 2^{-\ell+4}$. In this case, we will aim show $\hat{q}(x) > \frac{2}{2^k} \geq \beta 2^{-k}$, so it is sufficient to consider the worst case when $p_D(x) = 2^{-\ell+4}$. By Claim 5.4.20, we can assume there are at most

$$C_{col} \leq \frac{1}{4} T_2 |C^{-1}(x)|$$

total collisions over the choices of H_2, thus as in Lemma 5.4.22, the collision-corrected Chernoff bound still promises $\hat{q}(x) > \frac{2}{2^k}$ with probability at least $e^{-\Omega(2^{-k}T_2)} \leq \frac{O(\nu)}{2^m T_1}$. Union bounding over all choices of H_1, u, and values of x completes the proof of the first claim.

To prove the second claim, we observe that $\widehat{q}(y^*) \in (1 \pm O(\nu))p_D(y^*)2^{\ell-k}$ by Lemma 5.4.22. Bounding $(1 \pm O(\nu)) \in (1/2, 2)$, we have

$$p_D(y^*) \in (\widehat{q}(y^*)2^{k-\ell-1}, \widehat{q}(y^*)2^{k-\ell+1}).$$

If `ElemFind` returned y^*, it must be the case that

$$\widehat{q}(y^*) \in (\tfrac{\beta}{2}2^{-k}, \beta 2^{-k}] \in (2^{-k-1}, 2^{-k+1}],$$

and so

$$p_D(y^*) \in [2^{-\ell-2}, 2^{-\ell+2}],$$

conditioned on `ElemFind` returning y^*, as claimed.

□

5.5 Separating Stability: Statistical Barriers

5.5.1 Quadratic Separation: One-way Marginals

We start by defining the one-way marginals problem over d coordinates, which corresponds to outputting a good estimate of the expectation of a product of Rademacher distributions in ℓ_∞-distance.

Definition 5.5.1. *Consider a product of d Rademacher distributions with expectations $(p_1, \dots, p_d)$ respectively. Let $p = (p_1, \dots, p_d)$. A vector $v \in \mathbb{R}^d$ is said to be an α-accurate solution to the one-way marginals problem if $\|v - p\|_\infty \leq \alpha$.*

Definition 5.5.2. *Let C be the class of products of d Rademacher distributions. Fix any distribution D in C. We say that an algorithm (α, β)-accurately solves the one-way marginals problem over d coordinates, if it observes samples from the distribution D, and with probability at least $1-\beta$ (over the randomness of the samples and the algorithm), produces an α-accurate solution $v \in \mathbb{R}^d$.*

In this section, we show that any 0.0001-replicable, $(0.01, 0.01)$-accurate algorithm for the one-way marginals problem over d coordinates requires at least $\widetilde{\Omega}(d)$ samples.

On the other hand, under the constraint of $(1, \frac{1}{n^2})$-differential privacy, this problem can be solved using $\widetilde{O}(\sqrt{d})$ samples, via the Gaussian mechanism. This gives a quadratic separation between differential privacy and replicability, and proves that our reduction is asymptotically tight (up to logarithmic factors) in some settings (since our reduction would give a $\tilde{O}(d)$-sample replicable algorithm for this task).

The main theorem we prove in this section is the following.

Theorem 5.5.3. *Fix sufficiently large $d > 0$. If algorithm $\mathscr{A}$ is 0.0001-replicable and $(0.01, 0.001)$-accurately solves the one-way marginals problem over d coordinates with m samples, then $m = \tilde{\Omega}(d)$.*

Sketch of our approach

Our techniques for proving the lower bound for replicability draw inspiration from those used to prove lower bounds in privacy. Specifically, tight lower bounds for the one-way marginals problem over d coordinates under the constraint of differential privacy are obtained using the *fingerprinting method* [DSS+15, BUV18, BSU19]. The fingerprinting method captures the idea that there is a trade-off between accuracy and correlation with the input sample. It quantifies the idea that if the algorithm obtains a sample of small size, and is also very accurate, then it must be heavily correlated with one of its input examples, which is prohibited by differential privacy. Since replicability also prohibits such correlation (at least at a high level), one might expect the same method to be useful toward this end.

More formally, given an algorithm $\mathscr{A}$ solving the one-way marginals problem, the *correlation* of coordinate j of the output with the input sample S can be measured by the quantity

$$Z = \sum_{j \in [d]} \mathscr{A}^j(S) \sum_{i \in [m]} (S_i^j - p_j).$$

Note that the quantity $(S_i^j - p_j)$ represents the drift between an input example coordinate and the expectation of the distribution it's drawn from. $\mathbb{E}[Z]$ is large when on average, for many j, $\mathscr{A}^j(S)$ is on the same side of 0 as the sample drift $\sum_i (S_i^j - p_j)$, implying that the algorithm's outputs are on average correlated with its input.

We now recall the formal statement of the fingerprinting lemma.

Lemma 5.5.4 (Fingerprinting Lemma, Lemma 3.6 in [BSU19]). *Let f be any function from $\{-1,1\}^m \to [-1,1]$. Suppose r is sampled from the uniform distribution over $[-1,1]$ and $q \in \{-1,1\}^m$ is a vector of m independent Rademacher RVs each with expectation r. Then, if μ_q is the empirical average of q, we get that*

$$\mathbb{E}_{r,q}[f(q)\sum_i (q_i - r) + 2|f(q) - \mu_q|] \geq \frac{1}{3}.$$

The lower bound for differential privacy proceeds by arguing $\mathbb{E}[Z] = \mathbb{E}[\sum_j \mathscr{A}^j(S) \sum_i (S_i^j - p_j)]$ is large (via an appropriate application of the fingerprinting lemma), and hence, by an averaging argument, there exists an example $S_{i^*} \in S$ correlated with the output, i.e. some $i^* \in [m]$ such that

$$\mathbb{E}[Z_{i^*}] = \mathbb{E}[\sum_{j \in [d]} \mathscr{A}^j(S)(S_{i^*}^j - p_j)]$$

is large. With this in hand, consider an independently drawn example $g = (g_{i^*}^1, \ldots, g_{i^*}^d)$, and the neighboring dataset S' obtained by replacing S_{i^*} in the original dataset with g. Since g is independent of S_{i^*}, $\mathscr{A}^j(S')$ is uncorrelated with $(S^j_{i^*} - p_j)$, and hence the "1-neighboring" quantity

$$\mathbb{E}[Z'_{i^*}] = \mathbb{E}[\sum_{j\in[d]} \mathscr{A}^j(S')(S^j_{i^*} - p_j)]$$

should be small. On the other hand, differential privacy promises that Z_{i^*} is distributionally close to Z'_{i^*}, and hence $\mathbb{E}[Z'_{i^*}]$ and $\mathbb{E}[Z_{i^*}]$ must be close. Balancing these considerations gives a lower bound on the number of samples needed for differential privacy.

For replicability, the idea is to obtain a stronger lower bound by avoiding averaging. Specifically, for Z defined as above, we can argue that $\mathbb{E}[Z]$ is large (as we would for the differential privacy lower bound). Then, we can consider a freshly sampled dataset S' (drawn from a product distribution with the same expectation $p = (p_1, \dots, p_d)$), and consider the quantity $Z' = \sum_j \mathscr{A}^j(S') \sum_i (S^j_i - p_j)$. We can argue that Z' is a sum of uncorrelated random variables, and hence that $\mathbb{E}[Z']$ is small. On the other hand, by replicability, Z and Z' are distributionally close, since they correspond to post-processing of the algorithm applied to independent datasets. Note that this does not follow from differential privacy, since datasets S and S' may differ in many entries. Now, following a similar approach to the differential privacy lower bound, we'd get a stronger lower bound for replicability (since we have eliminated the averaging argument).

Unfortunately, this approach does not work directly for technical reasons. Specifically, ρ-replicability tells us that Z and Z' are distributionally close, but their expectations can have absolute value difference as large as ρdm (since $\mathscr{A}(S)$ and $\mathscr{A}(S')$ could differ completely with probability ρ). Since we are interested in constant ρ, this turns out to be too large for the lower bound technique to work.

We deal with this by instead applying the fingerprinting method to prove a lower bound against $(\frac{1}{m^3}, 1, \frac{1}{m^3})-$*perfectly generalizing* algorithms. We find this lower bound interesting in its own right, as it gives the first sample complexity separation between approximate differential privacy and perfect generalization. Perfect generalization roughly asks that the algorithm's output distributions be $(1, \frac{1}{m^3})$-close on two independent datasets drawn from the product distribution. This can be used to argue that $\mathbb{E}[|Z|]$ is within $\frac{1}{m^2}$ of (a constant multiple of) $\mathbb{E}[|Z'|]$, which turns out to be sufficient for the lower bound technique to apply.

Finally, appealing to our generic method of converting replicable algorithms to perfectly generalizing ones, this method extends to a tight lower bound on replicability (up to the loss of logarithmic factors in the number of coordinates d). It remains an interesting problem whether such a lower bound can be shown directly, ideally in a manner that avoids the resulting logarithmic loss.

Formal argument

We start by proving our new lower bound for perfectly generalizing algorithms.

Theorem 5.5.5. *Fix any $m > 0$ and sufficiently large $d > 0$. Let $\mathscr{A}$ be a $(\frac{1}{m^3}, 1, \frac{1}{m^3})$-perfectly generalizing, $(0.01, 0.01)$-accurate algorithm for the 1-way marginals problem over d attributes using m samples. Then, $m = \Omega(d)$.*

Proof. Assume without loss of generality that $m = \Omega(\log d)$.[11] Let $p \sim [-1,1]^d$, and draw $S, S' \sim D_p^m$ independently where D_p is a product of Rademachers with expectation $p = (p_1, \ldots, p_d)$. Define the random variables

$$Z = \sum_{j \in [d]} \mathscr{A}^j(S) \sum_{i \in [m]} (S_i^j - p_j), \quad Z' = \sum_{j \in [d]} \mathscr{A}^j(S) \sum_{i \in [m]} (S_i'^j - p_j).$$

As discussed above, we will argue that $\mathbb{E}[Z]$ is large (by the fingerprinting lemma):

$$\mathbb{E}[|Z|] \geq \frac{d}{10}, \tag{5.10}$$

[11] We will show under this condition that $m = \Omega(d)$. Any algorithm on $O(\log d)$ samples implies one between $O(\log d)$ and $O(d)$, which would give a contradiction.

that $\mathbb{E}[|Z'|]$ is small (since S' is independent of S):

$$\mathbb{E}[|Z'|] \leq 2\sqrt{dm}, \tag{5.11}$$

and finally that $\mathbb{E}[|Z|]$ and $\mathbb{E}[|Z'|]$ are close (by perfect generalization):

$$\mathbb{E}[|Z|] \leq e^2 \mathbb{E}[|Z'|] + \frac{8d}{m^2}. \tag{5.12}$$

Combining the inequalities we get

$$\frac{d}{10} \leq \mathbb{E}[|Z|] \leq e^2 \mathbb{E}\left[|Z'|\right] + \frac{8d}{m^2} \leq 2e^2\sqrt{dm} + \frac{8d}{m^2}$$

which implies $m \geq \Omega(d)$ as desired.

It remains to show Inequalities (5.10), (5.11), (5.12). We start with the first. Apply the fingerprinting lemma to the function corresponding to the j^{th} coordinate of the output of $\mathscr{A}$, when run on the j^{th} column of the input S with all other columns set to any fixed values. Then, we get that

$$\mathbb{E}_{p_j \sim [-1,1], S^j \sim Rad(p_j)^m}[\mathscr{A}^j(S;r)\sum_i (S_i^j - p_j) + 2|\mathscr{A}^j(S;r) - \mu_j|] \geq \frac{1}{3},$$

where μ_j is the empirical average of the j^{th} column of the dataset. Since this is true for all fixed coins of the algorithm and fixed values of the other columns, by the law of total expectation, it is also true for random coin tosses and any distribution over the values of the other columns, and we get that

$$\mathbb{E}_{p, S \sim D_p^m, \mathscr{A}}[\sum_j \mathscr{A}^j(S) \sum_i (S_i^j - p_j) + 2|\mathscr{A}^j(S) - \mu_j|] = \mathbb{E}[Z] + 2\mathbb{E}[\sum_j |\mathscr{A}^j(S) - \mu_j|] \geq \frac{d}{3}$$

where we have used that S is drawn from a product distribution. It is therefore enough to argue that

$\mathbb{E}[\sum_j |\mathscr{A}^j(S) - \mu_j|]$ is small.

Since $\mathscr{A}$ is $(0.01, 0.01)$-accurate, we can say that with probability at least 0.99, for all $j \in [d]$, $|\mathscr{A}^j(S) - p_j| \leq \frac{1}{100}$. Taking expectation, we get that $\mathbb{E}[\sum_{j\in[d]} |\mathscr{A}^j(S) - p_j|] \leq \frac{3d}{100}$. By a Chernoff bound, we can argue that since $m = \Omega(\log d)$, $\max_j |p_j - \mu_j| \leq 0.01$ with probability at least 0.99. By the triangle inequality, this gives us that $\mathbb{E}[\sum_{j\in[d]} |\mathscr{A}^j(S) - \mu_j|] \leq \frac{6d}{100}$. Since this holds for a fixed product of Rademachers, it also holds when the expectation of the Rademacher random variables are chosen at random which proves Equation (5.10).

Next, we show Equation (5.11), that $\mathbb{E}[|Z'|]$ is small. Towards this end, first note that Z' is a sum of mean 0 uncorrelated random variables. To see this, consider random variables $M = \mathscr{A}^j(S)(S_i'^j - p_j)$ and $N = A^j(S)(S_{i'}'^j - p_j)$ for indices $i \neq i'$. We claim that $\mathbb{E}[M] = \mathbb{E}[N] = 0$. This is by the following sequence of inequalities (we prove this for M, the same argument holds for N).

$$\begin{aligned}\mathbb{E}[M] &= \mathbb{E}_p \mathbb{E}_{S,S',\mathscr{A}}[M \mid p] \\ &= \mathbb{E}_p \mathbb{E}_{S,S',\mathscr{A}}[\mathscr{A}^j(S)(S_i'^j - p_j) \mid p] \\ &= \mathbb{E}_p \Big[\mathbb{E}_{S,S',\mathscr{A}}[\mathscr{A}^j(S) \mid p]\, \mathbb{E}_{S,S',\mathscr{A}}[(S_i'^j - p_j) \mid p]\Big] = 0,\end{aligned}$$

where the last equality follows because conditioned on the vector p, the expectation of the Rademacher S_i^j is exactly equal to p_j. Hence, by linearity of expectation, we get that the expectation of Z' is also 0.

Next, we show that M and N are uncorrelated. First, conditioning on p we can write

$$\begin{aligned}\mathbb{E}_{p,S,S',\mathscr{A}}[MN] &= \mathbb{E}_p \mathbb{E}_{S,S',\mathscr{A}}[MN \mid p] \\ &= \mathbb{E}_p \Big[\mathbb{E}_{S,S',\mathscr{A}}[\mathscr{A}^j(S)^2 \mid p]\, \mathbb{E}_{S,S',\mathscr{A}}[(S_i'^j - p_j)(S_{i'}'^j - p_j) \mid p]\Big]\end{aligned}$$

since S and S' are independent after conditioning. Since this is also the case for $(S_i'^j - p_j)$ and

$(S'^j_{i'} - p_j)$ we have

$$\mathbb{E}[(S'^j_i - p_j)(S'^j_{i'} - p_j) \mid p] = \mathbb{E}[(S'^j_i - p_j) \mid p]\ \mathbb{E}[(S'^j_{i'} - p_j) \mid p] = 0.$$

Since we have already seen that $\mathbb{E}[M]\mathbb{E}[N] = 0$ (since $\mathbb{E}[M] = 0$), this implies that M and N are uncorrelated as desired.

Now assuming without loss of generality that $\mathscr{A}$ outputs values between $[-1,1]$ (rounding inputs to this range only improves the accuracy and doesn't affect perfect generalization, which is robust to post-processing), we have that

$$\begin{aligned}
\mathbb{E}^2[|Z'|] &= \mathbb{E}^2_{p,S,S',\mathscr{A}}\Big[\Big|\sum_{j\in[d]} \mathscr{A}^j(S)\sum_i (S'^j_i - p_j)\Big|\Big] && (S \text{ and } S' \text{ are i.i.d}) \\
&\le \mathbb{E}_{p,S,S',\mathscr{A}}\Big[\Big(\sum_{j\in[d]} \mathscr{A}^j(S)\sum_i (S'^j_i - p_j)\Big)^2\Big] && (\text{Jensen's Inequality}) \\
&= Var(Z'). && (\mathbb{E}[Z'] = 0)
\end{aligned}$$

Since Z' is a sum of uncorrelated random variables and $\mathscr{A}^j(S)^2 \le 1$, we then get

$$\begin{aligned}
Var(Z') &= \sum_j \sum_i Var(\mathscr{A}^j(S)(S'^j_i - p_j)) \\
&\le \sum_j \sum_i \mathbb{E}[(S'^j_i - p_j)^2] \\
&\le 4dm
\end{aligned}$$

as desired.

It is left to show Equation (5.12). Let Z_p be the random variable Z conditioned on fixed p (and likewise for Z'). Let $Z_{p,S,S'}$ be the random variable Z conditioned on fixed p, S and S' (and likewise for Z'). If A is perfectly generalizing, then by Lemma 5.3.5 and Lemma 5.2.5, for all fixed p, with probability at least $1 - \frac{1}{m^3}$ over the draw of S, S', we have that $Z_{p,S,S'}$ and $Z'_{p,S,S'}$

are distributionally close, as are $|Z_{p,S,S'}|$ and $|Z'_{p,S,S'}|$. For any fixed p, let E be the event that $|Z_{p,S,S'}| \approx_{2,\frac{3}{m^3}} |Z'_{p,S,S'}|$, where the randomness in E comes from the randomness of sampling S and S'. Then, by the guarantee of perfect generalization, we have that for all fixed p, E occurs with probability at least $1-\frac{1}{m^3}$ and for any fixed p we can write:

$$\begin{aligned}
\mathbb{E}[|Z_p|] &= \mathbb{E}_{S,A}\left[\Big|\sum_{j\in[d]} \mathscr{A}^j(S)\sum_i (S_i^j - p_j)\Big|\right] \\
&= \int_0^{2dm} \mathbf{Pr}[|Z_p| > z]dz \\
&= \int_0^{2dm} \left[\mathbf{Pr}[|Z_p| > z \mid E]\mathbf{Pr}[E] + \mathbf{Pr}[|Z_p| > z \mid \overline{E}]\mathbf{Pr}[\overline{E}) \right] dz \\
&\leq \int_0^{2dm} (e^2\mathbf{Pr}[|Z'_p| > z \mid E] + \frac{3}{m^3})\mathbf{Pr}[E] + \mathbf{Pr}[\overline{E}]dz \\
&\leq \int_0^{2dm} e^2\mathbf{Pr}[|Z'_p| > z \mid E]\mathbf{Pr}[E]dz + \int_0^{2dm}\left[\frac{3}{m^3} + \mathbf{Pr}[\overline{E}]\right] dz \\
&\leq \int_0^{2dm} e^2\mathbf{Pr}[|Z'_p| > z]dz + \int_0^{2dm}\left[\frac{3}{m^3} + \frac{1}{m^3}\right] dz \\
&= e^2\mathbb{E}_{S,S',\mathscr{A}}[|Z'_p|] + \frac{8d}{m^2},
\end{aligned}$$

where the first inequality follows since $|Z_p|$ and $|Z'_p|$ are distributionally close conditioned on E, the second inequality is by the fact that $\mathbf{Pr}(E) \leq 1$, and the third since $\overline{E}$ corresponds to the probability of failure in the definition of perfect generalization.

Finally taking expectation with respect to p, we get that

$$\mathbb{E}[|Z|] \leq e^2\mathbb{E}[|Z'|] + \frac{8d}{m^2}$$

as desired. □

Now, we are ready to prove the lower bound for replicable algorithms.

Proof of Theorem 5.5.3. Let m be larger than an absolute constant K, without loss of generality.[12]

[12]We will show under this condition that $m = \tilde{\Omega}(d)$. Any algorithm taking fewer than K samples implies one taking

By Claim 5.2.12, we have that $\mathscr{A}$ is $(0.01, 0.01)$-replicable and $(0.01, 0.001)$-accurate when given m samples. Consider any sufficiently small $\gamma > 0$, and sufficiently large constant $c > 0$. Applying Theorem 5.2.13, we get that there is a $\frac{1}{c\log(1/\gamma)}$-replicable and $(0.01, 0.008 + \frac{1}{\log(1/\gamma)}) = (0.01, 0.01)$-accurate algorithm $\mathscr{A}'$ for one-way marginals over d coordinates, which takes $O\left(m\log^2(1/\gamma)\right)$ samples.

Next, we give a way of replicably amplifying the failure probability to γ. We run the algorithm $\mathscr{A}'$ $k = 20\log(1/\gamma)$ times on different samples, and take the coordinate-wise median of the outputs. Observe that for each coordinate, if more than half the values in that coordinate are within 0.01 of the true bias, then the median is correct. Consider the probability that more than half the output values in a coordinate are not within 0.01 of the true bias. By a Chernoff bound, we have that the number of outputs which are within 0.01 of the true expectation in l_∞ norm are more than $0.5k$ with probability at least $1 - \gamma^2$, which guarantees that we get a $(0.01, \gamma^2)$-accurate algorithm for one-way marginals. Using composition of replicability, we have that the resulting algorithm is $(0.01, 0.01)$-replicable and takes $O\left(m\log^3(1/\gamma)\right)$ samples.

Consider any sufficiently small $\delta > 0$. By Theorem 5.3.19, there is a $(2\delta, 1, 2\delta)$-PG algorithm with failure probability at most $\delta + \gamma\log(1/\delta)$ when given $m' = O(m\log^3(1/\gamma)\text{polylog}(1/\delta))$ samples. Setting $\gamma = \frac{0.005}{\log 1/\delta}$, we get that that for sufficiently small $\delta > 0$, there is a $(2\delta, 1, 2\delta)$-PG algorithm with failure probability at most $\delta + 0.005$ (i.e. $(0.01, \delta + 0.005)$-accurate), when given $m' = O(m \cdot \text{polylog}(1/\delta))$ samples. Setting $\delta = \frac{1}{2m'^3}$ and simplifying, we get that $m' = Cm \cdot \text{polylog}(m)$ for some constant C. Then, since $m' > m$ is larger than K, we get that δ is smaller than $\frac{1}{K}$ and setting K sufficiently large gives us a $(0.01, 0.01)$-accurate algorithm with m' samples.

Now, using the lower bound for perfect generalization in Theorem 5.5.5, we get that $m' = \Omega(d)$, which gives us that $m = \tilde{\Omega}(d)$, completing the proof. □

between K and $\tilde{O}(d)$ samples, which would give a contradiction.

5.5.2 Quadratic separation: Agnostic Learning

In this section, we prove a lower bound for agnostic learning (See Section 5.2.4 for the definition of agnostic learning) under the constraint of replicability.

Theorem 5.5.6. *Fix sufficiently large $d > 0$ and a hypothesis class H with VC dimension d. Any $(0.01, 0.001)$-accurate, 0.0001-replicable agnostic learner $\mathscr{A}$ for H requires at least $\tilde{\Omega}(d^2)$ examples.*

The key idea is that we will reduce a variant of the one-way marginals problem over d coordinates to the problem of agnostically learning any hypothesis class with VC dimension d (with quadratically more samples). The variant we consider loosely corresponds to predicting the signs of the biases of the product distribution. We show that this is possible using an agnostic learner as a subroutine. We start by defining this problem more precisely.

Sign-One-Way Marginals

Definition 5.5.7. *Consider a product of d Rademacher distributions with expectations $p_1, \ldots, p_d$. A vector $v \in [-1,1]^d$ is said to be an α-accurate solution to the sign-one-way marginals problem for this distribution if $\frac{1}{d}\sum_{j=1}^{d} v_j p_j \geq \frac{1}{d}\sum_{j=1}^{d} |p_j| - \alpha$.*

Observe that if every p_j is either -1 or 1, an accurate solution requires the v_j's to do a very good job of predicting the signs on average. On the other hand, if the p_j values are all 0, then every value of v is a 0-accurate solution. Thus, this definition of error scales depending on how biased the expectation is to either 1 or -1, penalizing solutions more when they do a poor job of predicting heavily biased coordinates. (Indeed, we'd expect biased coordinates to be easier to predict, so it makes sense to penalize solutions more on these coordinates.) Now, we are ready to define the accuracy of an algorithm for the sign-one-way marginals problem.

Definition 5.5.8. *Let C be the class of products of d Rademacher random variables. We say that an algorithm $\mathscr{A} : \{\{-1,1\}^d\}^m \to [-1,1]^d$ (α, β)-accurately solves the sign-one-way marginals* *problem over class C if for all fixed distributions D in C, with probability at least* $1-\beta$ *over the randomness of the examples it obtains from D and the internal randomness of the algorithm, it outputs an* α*-accurate vector v for D.*

Solving Sign-One-Way Marginals using Agnostic Learning

Our reduction in Algorithm 32 shows how to use an agnostic learner $\mathscr{A}_{ag}$ for any class H of VC dimension d to construct an algorithm $\mathscr{A}$ for the sign-one-way marginals problem.

The main idea of the algorithm is as follows. Fix a distribution D that is a product of Rademachers and let its expectation be $p = (p_1, \ldots, p_d)$. Consider a shattered set $x_1, \ldots, x_d$ for hypothesis class H. Consider a distribution D' corresponding to sampling a uniformly random point x_j from the shattered set and then sampling a label in $(-1, 1)$ from a Rademacher with expectation p_j. Note that given a sample S of d independently drawn examples from D, we can create a dataset S_{ag} of size roughly d^2 that looks like an i.i.d. sample from D', by sampling a uniformly random x_j and labeling it with a new unused entry from coordinate j of S (we won't run out of entries with high probability). Now, note that since the set $x_1, \ldots, x_d$ is shattered by H, there is a hypothesis h in H that outputs $\text{sign}(p_j)$ on input x_j (such a hypothesis also achieves lowest possible error on D' among hypotheses in H). If the agnostic learner is accurate when given d^2 samples, then the function f it outputs is a good approximation to h and as a result $f(x_j)$ is also likely to be an accurate prediction of the sign of p_j. Hence, function f can be used to obtain an accurate solution to the sign-one-way marginals problem.

Algorithm 32. Algorithm $\mathscr{A}$ for sign-one-way marginals

```
Input: Sample access to a product distribution $D$ over $\{-1,1\}^d$, agnostic learner $\mathscr{A}_{ag}$ for hypothesis class $H$ with VC dimension $d$
Output: Estimated biases $(v_1,\ldots,v_d)$.
1: Draw $\frac{d}{\log^c d}$ i.i.d. examples from $D$ for some $c>0$. Call the corresponding sample $S_{inp}$.
2: Let $x_1,\ldots,x_d$ be a shattered set of points for $H$. Let $U_d$ be the uniform distribution over $x_1,\ldots,x_d$.
3: Draw $m=\frac{d^2}{100\log^{2c} d}$ examples $S_j$ from $U_d$. Call the sample $S_{ag}$. If any element $x_i$ occurs more than $\frac{d}{\log^c d}$ times, then output $(1,\ldots,1)$, else move to the next step.
4: For each example $S_j$, label it with a new entry from coordinate $j$ of the input sample $S_{inp}$. Call the labeled sample $S_{ag,lab}$.
5: Run agnostic learner $\mathscr{A}_{ag}$ on the labeled sample $S_{ag,lab}$. Let the output function be $f$.
6: return $(f(x_1),f(x_2),\ldots,f(x_d))$.
```

Theorem 5.5.9. *Fix sufficiently large $d>0$. Let $\mathscr{A}_{ag}$ be a $(0.01,0.001)$-accurate, 0.0001-replicable agnostic learner for a hypothesis class H with VC dimension d. Then algorithm $\mathscr{A}$ is a $(0.02,0.002)$-accurate, 0.0003-replicable algorithm for the sign-one-way marginals problem over d coordinates.*

Proof. Let D be a Rademacher distribution with expectation $(p_1,\ldots,p_d)$. Define a distribution D_{ideal} over $\{x_1,\ldots,x_d\}\times\{-1,1\}$ as follows. First, uniformly draw $s\in\{x_1,\ldots,x_d\}$. Then, if $s=x_j$, draw y from a Rademacher with expectation p_j. Let D_{ideal} be the distribution of the random variable (s,y) obtained using this procedure.

First, we observe that by a Chernoff bound and union bound, the probability that any element x_i occurs more than $\frac{d}{\log^c d}$ times in S_{ag} (where S_{ag} is sampled as described in Step 3) is exponentially small in d (hence less than 0.01 for sufficiently large d). Call this bad event E.

Next, consider the following method for sampling a dataset S_{ideal} of $\frac{d}{100\log^{2c} d}$ i.i.d. samples from D_{ideal}: first draw $\ell=\frac{d}{100\log^{2c} d}$ i.i.d. examples $s_i\sim U_d$ and then for each of them, if the value obtained is x_j, sample $y_j \sim Rad(p_j)$. Consider the event E_{ideal} that the number of occurrences of any example x_i is larger than $\frac{d}{\log^c d}$. The probability of this event is exactly equal to $\mathbf{Pr}[E]$. Notice that the distribution of $S_{ag,lab}$ is identical to the distribution of S_{ideal} conditioned on event $\overline{E_{ideal}}$. For any distribution D' and event $\overline{E}$, a simple calculation shows $d_{\mathrm{TV}}(D', D'|_{\overline{E}}) \leq \mathbf{Pr}[E]$. Hence, we get that the total variation distance between the distribution of S_{ideal} conditioned on event $\overline{E_{ideal}}$ and the distribution of S_{ideal} is at most 0.0001 for sufficiently large d.

Now, we know that with probability at least 0.999 over the coins of the algorithm and the sample, the agnostic learner produces an output that is accurate with respect to its input sample. By the data-processing inequality for total variation distance, we have that $d_{\mathrm{TV}}(\mathscr{A}_{ag}(S_{ideal}), \mathscr{A}_{ag}(S_{ag,lab})) \leq 0.0001$. Consider the distribution D_{ideal} and the subset O of 0.01-accurate functions w.r.t. the best function in the class H (i.e. the function that minimizes $\mathbf{Pr}_{(x,y) \in D_{ideal}}[h(x) \neq y]$) . By the definition of total variation distance, we have that the probability that learner $\mathscr{A}_{ag}$ produces outputs in this subset O on seeing $S_{ag,lab}$ is within 0.0001 of the probability that $\mathscr{A}_{ag}$ produces outputs in this subset O on seeing S_{ideal}. Since the latter happens with probability at least 0.999, we have that with probability at least 0.9989, the agnostic learner is 0.01-accurate when fed the sample $S_{ag,lab}$. This implies by the definition of the accuracy guarantee that with probability at least 0.9989 over the randomness of the learner $\mathscr{A}_{ag}$ and sample $S_{ag,lab}$, that

$$\mathbf{Pr}_{(x,y) \sim D_{ideal}}[f(x) \neq y] \leq \inf_{h \in H} \mathbf{Pr}_{(x,y) \sim D_{ideal}}[h(x) \neq y] + 0.01,$$

where f is the output function of the agnostic learner. For $x \in \{x_1, \ldots, x_d\}$, let p_x be p_j if $x = x_j$. Observe that the function that predicts $\mathrm{sign}(p_x)$ achieves the infimum on the right hand side of the above equation.

Now, using the fact that expectation of an indicator is the probability of the indicated event, and that $\mathbb{1}[a \neq b] = \frac{1-ab}{2}$ when a and b are in $\{-1, 1\}$, we get that with probability at least 0.9989 over the randomness of the learner $\mathscr{A}_{ag}$ and sample $S_{ag,lab}$,

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim D_{ideal}}[\mathbb{1}[f(x)\neq y]] \leq \inf_{h\in H}\mathbb{E}_{(x,y)\sim D_{ideal}}[\mathbb{1}[h(x)\neq y]]+0.01\\
\implies &\mathbb{E}_{(x,y)\sim D_{ideal}}\left[\frac{1-f(x)y}{2}\right] \leq \inf_{h\in H}\mathbb{E}_{(x,y)\sim D_{ideal}}\left[\frac{1-h(x)y}{2}\right]+0.01\\
\implies &\mathbb{E}_{(x,y)\sim D_{ideal}}\left[\frac{1-f(x)y}{2}\right] \leq \mathbb{E}_{(x,y)\sim D_{ideal}}\left[\frac{1-\operatorname{sign}(p_x)y}{2}\right]+0.01\\
\implies &\mathbb{E}_{(x,y)\sim D_{ideal}}[f(x)y] \geq \mathbb{E}_{(x,y)\sim D_{ideal}}[\operatorname{sign}(p_x)y]-0.02.
\end{aligned}$$

Now, Algorithm $\mathscr{A}$ calls the agnostic learner except with probability 0.0001. Unraveling the expectations, accounting for the fact that $\mathscr{A}$ outputs $(1,1\ldots,1)$ when it doesn't call the agnostic learner, and using the fact that the randomness of sample $S_{ag,lab}$ is from the randomness of the algorithm $\mathscr{A}$ as well as the randomness of S_{inp}, we get that with probability at least 0.998 over the randomness of the algorithm $\mathscr{A}$ and input sample, S_{inp}, that

$$\frac{1}{d}\sum_{j=1}^{d} f(x_j)p_j \geq \frac{1}{d}\sum_{j=1}^{d}\operatorname{sign}(p_j)p_j - 0.02,$$

proving that $\mathscr{A}$ is a $(0.02, 0.002)$-accurate algorithm for sign-one-way marginals over d coordinates.

Next, we prove that $\mathscr{A}$ inherits the replicability of the agnostic learner $\mathscr{A}_{ag}$. Consider two sets of independent samples $S_{inp,1}$ and $S_{inp,2}$. Consider any set of random coins r drawn for algorithm $\mathscr{A}$. Note that when the coins dictate that when some point in the shattered set occurs too many times, the algorithm always outputs $(1,\ldots,1)$. Recall that this is event E, which we previously showed occurs with probability at most 0.0001. Hence, in this case $\mathscr{A}(S_{inp,1};r) = \mathscr{A}(S_{inp,2};r)$ with probability 1. Hence, it is sufficient to consider coins such that every point in the shattered set occurs fewer than $\frac{d}{\log^{2c} d}$ times in the sample S_{ag}. Now, as argued previously, the total variation distance between the distribution of $S_{ag,lab}$ (call it D_{ag}) and the same number of i.i.d. samples from D_{ideal} is at most 0.0001 for sufficiently large d. Thus, we have that the probability of any event changes by at most 0.0002 under samples $S_{ag,lab}, S'_{ag,lab} \sim D^m_{ideal}$ versus samples $S_{ag,lab}, S'_{ag,lab} \sim D^m_{ag}$. This allows us to conclude that

$$
\begin{aligned}
&\mathbf{Pr}_{S_{inp,1},S_{inp,2}\sim D^m,r}[\mathscr{A}(S_{inp,1};r) = \mathscr{A}(S_{inp,2};r)] \\
&\qquad \geq \mathbf{Pr}_{S_{inp,1},S_{inp,2}\sim D^m,r}[\mathscr{A}(S_{inp,1};r) = \mathscr{A}(S_{inp,2};r) \mid \overline{E}] \\
&\qquad \geq \mathbf{Pr}_{S_{ag,lab},S'_{ag,lab}\sim D_{ag},r_{ag}}[\mathscr{A}_{ag}(S_{ag,lab};r_{ag}) = \mathscr{A}_{ag}(S'_{ag,lab};r_{ag})] \\
&\qquad \geq \mathbf{Pr}_{S_{ag,lab},S'_{ag,lab}\sim D^m_{ideal},r_{ag}}[\mathscr{A}(S_{ag,lab};r_{ag}) = \mathscr{A}(S'_{ag,lab};r_{ag})] - 0.0002 \\
&\qquad \geq 0.9999 - 0.0002 = 0.9997.
\end{aligned}
$$

Hence, we have proved that $\mathscr{A}$ is 0.0005-replicable. □

Lower Bound for Sign-One-Way Marginals

In this section, we show that accurately and replicably solving the sign-one-way marginals problem over d coordinates requires a number of samples that is nearly linear in d. We will use a variant of the *fingerprinting method* used to prove the lower bound for the one-way marginals problem for perfectly generalizing algorithms, and then extend this to a lower bound for replicable algorithms. This argument is similar to that used to prove the lower bound for the one-way marginals problem (Theorem 5.5.5), except that the notion of accuracy is different, and so we use a different version of the fingerprinting lemma, given below.

Lemma 5.5.10 ([BSU19], Lemma A.1 and A.2)**.** *Let f be a function from $\{-1,1\}^m \to \mathscr{R}$. Let p be a uniformly random variable between -1 and 1, and $\vec{x}$ be a random vector of length m, consisting of i.i.d. Rademacher random variables with expectation p.*

Then,

$$\mathbb{E}_{p,\vec{x}}[f(\vec{x})\sum_{i=1}^{m}(x_i - p)] = \mathbb{E}_p[2pg(p)]$$

where $g(p) = \mathbb{E}_{\vec{x}\sim p}[f(\vec{x})]$.

Theorem 5.5.11. *Fix any* $m > 0$*, sufficiently large* $d > 0$*. Let* $\mathscr{A}$ *be a* $(\frac{1}{m^3}, 1, \frac{1}{m^3})$*-perfectly generalizing,* $(0.05, 0.05)$*-accurate algorithm for the sign-one-way marginals problem over* d *coordinates using* m *samples that always outputs a vector in* $[-1,1]^d$*. Then,* $m = \Omega(d)$*.*

Proof. Without loss of generality, let m be larger than a constant K.[13] Let S be the input dataset to the algorithm $\mathscr{A}$. Following the framework in Theorem 5.5.5, we will first argue the expected correlation of our algorithm and its input is large:

$$\sum_{j=1}^{d}\mathbb{E}_{p,\mathscr{A},S\sim D_p^m}[\mathscr{A}^j(S)\sum_{i=1}^{m}(S_i^j - p_j)] \geq 0.35d.$$

To see this, observe that by the accuracy of the algorithm, for any fixed distribution D_p that is a product of Rademachers with expectation $p = (p_1, \dots, p_d)$, we have that we have that with probability at least 0.95 over the randomness of the algorithm $\mathscr{A}$ and its input sample,

$$\frac{1}{d}\sum_{j=1}^{d} v_j p_j \geq \frac{1}{d}\sum_{j=1}^{d}\mathrm{sign}(p_j)p_j - 0.05,$$

where v is the vector output by the algorithm.

Now, taking expectation over the randomness of the algorithm $\mathscr{A}$ and the input sample S,

[13] We will show under this condition that $m = \Omega(d)$. Any algorithm on $\leq K$ samples implies one taking between K and $O(d)$ samples, which would give a contradiction.

we get that

$$\mathbb{E}_{\mathscr{A},S\sim D_p^m}\left[\frac{1}{d}\sum_{j=1}^{d}\mathscr{A}^j(S)p_j\right] \geq 0.95\left[\frac{1}{d}\sum_{j=1}^{d}|p_j| - 0.05\right] - 0.05,$$

which implies that

$$\mathbb{E}_{\mathscr{A},S\sim D_p^m}\left[\frac{1}{d}\sum_{j=1}^{d}\mathscr{A}^j(S)p_j\right] \geq \frac{1}{d}\sum_{j=1}^{d}|p_j| - 0.15,$$

Now, consider each coordinate of expectation vector p drawn uniformly from $[-1,1]$. Then, we have that conditioned on any fixed p, the above equation holds. Hence, using the law of total expectation, we get that

$$\mathbb{E}_{p,\mathscr{A},S\sim D_p^m}\left[\sum_{j=1}^{d}\mathscr{A}^j(S)p_j\right] \geq \mathbb{E}_p\left[\sum_{j=1}^{d}|p_j|\right] - 0.15d \tag{5.13}$$

$$\implies \sum_{j=1}^{d}\mathbb{E}_{p,\mathscr{A},S\sim D_p^m}\left[\mathscr{A}^j(S)p_j\right] \geq \sum_{j=1}^{d}\mathbb{E}_{p_j\sim[-1,1]}\left[|p_j|\right] - 0.15d = 0.35d, \tag{5.14}$$

where we have used the fact that $\mathbb{E}_{p_j\sim[-1,1]}[|p_j|] = \frac{1}{2}$. Now, fix a coordinate $j \in [d]$. For any fixed internal randomness r of algorithm $\mathscr{A}$, and for any values of columns of S that are not the j^{th} column, we get from Lemma 5.5.10 applied to the function f corresponding to the algorithm $\mathscr{A}$ on the complete dataset S with internal randomness r, that

$$\mathbb{E}_{p_j,S^j\sim Rad(p_j)^m}\left[\mathscr{A}^j(S;r)p_j\right] = \mathbb{E}_{p_j,S^j\sim Rad(p_j)^m}[\mathscr{A}^j(S;r)\sum_{i=1}^{m}(S_i^j - p_j)].$$

Now, since this holds for any fixed values of internal randomness r and for any values of columns of S that are not the j^{th} column, it holds for any distribution over the internal randomness r and any distribution over values of other columns of S. Hence, we get that

$$\mathbb{E}_{p,\mathscr{A},S\sim D_p^m}\left[\mathscr{A}^j(S)p_j\right] = \mathbb{E}_{p,\mathscr{A},S\sim D_p^m}[\mathscr{A}^j(S)\sum_{i=1}^{m}(S_i^j - p_j)],$$

where we have used that D_p is a product distribution. Now, summing over all coordinates $j \in [d]$, we get that

$$\sum_{j=1}^{d}\mathbb{E}_{p,\mathscr{A},S\sim D_p^m}[\mathscr{A}^j(S)\sum_{i=1}^{m}(S_i^j - p_j)] = \sum_{j=1}^{d}\mathbb{E}_{p,\mathscr{A},S\sim D_p^m}\left[\mathscr{A}^j(S)p_j\right] \geq 0.35d, \tag{5.15}$$

where we have used Equation (5.14).

Now, we can proceed exactly as in the proof of Theorem 5.5.5 (we repeat high-level details for completeness; for more details, see that proof).

Let S' be another dataset drawn from the same distribution D_p. Let $Z = \sum_{j\in[d]}\mathscr{A}^j(S)\sum_i(S_i^j - p_j)$ and $Z' = \sum_{j\in[d]}\mathscr{A}^j(S')\sum_i(S_i^j - p_j)$.

First, note that Z' is a sum of uncorrelated random variables with mean 0 (see the proof of Theorem 5.5.5 for a proof of this).

We can then prove (as in the proof of Theorem 5.5.5) that

$$\mathbb{E}[|Z'|] = \mathbb{E}_{p,\mathscr{A},S,S'\sim D_p^m}\left[\Big|\sum_{j\in[d]}\mathscr{A}^j(S')\sum_i(S_i^j - p_j)\Big|\right] \leq 2\sqrt{dm},$$

Then, we can invoke the perfect generalization guarantee to prove (as in the proof of Theorem 5.5.5) that

$$\mathbb{E}[|Z|] \leq e^2\mathbb{E}[|Z'|] + \frac{8d}{m^2},$$

Hence, combining the inequality above with equation 5.15, we get that

$$0.35d \leq \mathbb{E}[|Z|] \leq e^2\mathbb{E}[|Z'|] + \frac{8d}{m^2} \leq 2e^2\sqrt{dm} + \frac{8d}{m^2}.$$

Simplifying, this gives that

$$m = \Omega(d).$$

$\square$

Now, we are ready to apply our conversion from replicability to perfect generalization to prove a similar lower bound for replicable algorithms.

Theorem 5.5.12. *Fix sufficiently large $d > 0$. For any 0.0005-replicable algorithm $\mathscr{A}$ that is $(0.02, 0.002)$-accurate on the sign-one-way marginals problem over d coordinates with m samples,*

$$m = \tilde{\Omega}(d).$$

Proof. Without loss of generality, let m be larger than a constant K.[14] By Claim 5.2.12, we have that $\mathscr{A}$ is $(0.01, 0.05)$-replicable and $(0.02, 0.002)$-accurate when given m samples. Consider any sufficiently small $\gamma > 0$. Next, applying Theorem 5.2.13, we get that for sufficiently large constant $c > 1$, there is a $\frac{1}{c\log(1/\gamma)}$-replicable and $(0.01, 0.008 + \frac{4}{c\log(1/\gamma)}) = (0.01, 0.01)$-accurate algorithm $\mathscr{A}'$ for sign-one-way marginals over d coordinates, which takes $O\left(m\log^2(1/\gamma)\right)$ samples.

Next, we give a way of replicably amplifying the failure probability to γ. We run the algorithm $\mathscr{A}$ for $k = 20\log(1/\gamma)$ times on different samples, and take the mean of the outputs. Using composition of replicability, we have that the resulting algorithm is 0.0001-replicable and takes $O\left(m\log^3(1/\gamma)\right)$ samples. Now, we analyze the failure probability of this algorithm. Let the output vectors of $\mathscr{A}$ on the k runs be $v^1, \ldots, v^k$. We are interested in the quantity $\frac{1}{d}\sum_{j=1}^{d}\left[|p_j| - \frac{p_j}{k}\sum_{i=1}^{k} v_j^i\right]$.

[14]We will show under this condition that $m = \tilde{\Omega}(d)$. Any algorithm on $\leq K$ samples implies one taking between K and $\tilde{O}(d)$ samples, which would give a contradiction.

First, we analyze the expectation of this quantity.

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{d}\sum_{j=1}^{d}\left[|p_j| - \frac{p_j}{k}\sum_{i=1}^{k} v_j^i\right]\right] &= \mathbb{E}\left[\frac{1}{d}\sum_{j=1}^{d} p_j\left[\operatorname{sign}(p_j) - \frac{1}{k}\sum_{i=1}^{k} v_j^i\right]\right] \\
&= \mathbb{E}\left[\frac{1}{d}\sum_{j=1}^{d}\frac{1}{k}\sum_{i=1}^{k} p_j\left[\operatorname{sign}(p_j) - v_j^i\right]\right] \\
&= \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[\frac{1}{d}\sum_{j=1}^{d} p_j\left[\operatorname{sign}(p_j) - v_j^i\right]\right] \le 0.03
\end{aligned}$$

where the last inequality is because the quantity inside the last expectation is less than 0.02 with probability at least 0.99 (and because all the v^i are identically distributed). Next, observe that the quantity $\frac{1}{k}\sum_{i=1}^{k}\left(\frac{1}{d}\sum_{j=1}^{d} p_j\left[\operatorname{sign}(p_j) - v_j^i\right]\right)$ is a sum of k independent random variables (since the i^{th} term in the sum only depends on random variable v^i) in the interval $[-2,2]$. Hence, using Hoeffding's inequality, we have that the probability that the sum is larger than 0.05 is less than γ^2.

Now, by Theorem 5.3.19, we have that there is a $(2\delta, 1, 2\delta)$-PG algorithm with failure probability at most $\delta + \gamma \log(1/\delta)$ when given $m' = O(m \log^3(1/\gamma) \operatorname{poly}\log(1/\delta))$ samples (where failure in this case means outputting a solution that is not 0.05-accurate). Setting $\gamma = \frac{0.005}{\log(1/\delta)}$, we get that that for sufficiently small $\delta > 0$, there is a $(2\delta, 1, 2\delta)$-PG algorithm with failure probability at most $\delta + 0.005$ (i.e., one that is $(0.05, \delta + 0.005)$-accurate), when given $m' = O(m \cdot \operatorname{poly}\log(1/\delta))$ samples. Setting $\delta = \frac{1}{2m'^3}$ and simplifying, we get that $m' = C \cdot m \cdot \operatorname{poly}\log m$ for some constant C. Hence, since $m' > m$ is larger than K, we get that δ is smaller than $\frac{1}{K}$ and setting K to be sufficiently large, we get a $(0.05, 0.05)$-accurate algorithm with m' samples.

Now, using the lower bound for perfect generalization in Theorem 5.5.11, we get that $m' = \Omega(d)$, which gives us that $m = \tilde{\Omega}(d)$, completing the proof. □

Now, we can use the reduction from sign-one-way marginals to agnostic learning to obtain the sample complexity lower bound for agnostic learning.

Proof of Theorem 5.5.6. If there were a $(0.01, 0.001)$-accurate, 0.0001-replicable agnostic learning algorithm using fewer than $\frac{d^2}{100\log^{2c} d}$ (where c is some sufficiently large constant) samples, then by Theorem 5.5.9, there would be a $(0.02, 0.002)$-accurate, 0.0005-replicable algorithm for the sign-one-way marginals problem over d coordinates taking only $\frac{d}{\log^c d}$ samples, which contradicts Theorem 5.5.12. $\square$

We note that our agnostic learning lower bound as stated only holds only for constant accuracy, and might not give the optimal dependence on the accuracy parameter α for general (α, β)-agnostic learning. We leave it as an open problem to determine the right dependence on α.

5.5.3 Closing the Gap: Realizable Learning

Now that we've seen natural settings in which our reduction is tight (and therefore exhibited a quadratic statistical separation between privacy and replicability), it is reasonable to ask whether there are any settings under which the reduction is loose, or even where privacy and replicability might have the same statistical cost. In this section, we'll show this is indeed the case for (certain regimes of) a closely related problem: realizable PAC-learning. In particular, in this section we exhibit a replicable algorithm for PAC-learning that gives a quadratically improved dependence on the accuracy and confidence parameters over applying our reduction from privacy (see Theorem 5.6.13).

Theorem 5.5.13 (Finite Classes are Replicably Learnable)**.** *Any class H is replicably Agnostic learnable with sample complexity:*

$$m(\rho, \alpha, \beta) \leq O\left(\frac{\log^2 |H| + \log \frac{1}{\rho\beta}}{\alpha^2 \rho^2} \log^3 \frac{1}{\rho}\right).$$

In the realizable setting, the α-dependence can be improved to linear:

$$m(\rho, \alpha, \beta) \leq O\left(\frac{\log^2 |H| + \log \frac{1}{\rho\beta}}{\alpha \rho^2} \log^3 \frac{1}{\rho}\right).$$

Theorem 5.5.13 gives a quadratic improvement over the sample complexity via reduction from private learning in both confidence and accuracy, and in particular has the same asymptotic dependence as in private PAC-learning (and hence avoids any statistical blowup in the setting where $\log|H|$ is thought of as small). In fact, it's worth noting the result is tight in these parameters, as even standard PAC-learning requires the same dependencies.

Algorithm

At its core, the algorithm achieving Theorem 5.5.13 relies on a simple random thresholding trick. In particular, the idea is roughly to estimate the risk of each concept in the class H by standard uniform convergence bounds, choose a random error threshold $v \in [OPT, OPT+\alpha]$, and finally output a random $f \in H$ with empirical error $err_S(f) = \frac{1}{|S|} \sum\limits_{(x,y)\in S} \mathbf{1}[f(x) \neq y]$ at most v. Implementing this strategy requires a bit more effort, and is achieved formally by the following algorithm.

Algorithm 33. (Intermediate) Replicable Learner for Finite Classes

Result: Replicably outputs hypothesis with error at most $OPT + \alpha$

Input: Finite Class H, Joint Distribution D over $X \times \{0,1\}$ (Sample Access)

Parameters:

- Replicability, Accuracy, Confidence $\rho, \alpha, \beta > 0$
- Sample Complexity $m = m(\rho,\alpha,\beta) \leq O\left(\frac{\log^2 |H| \log \frac{1}{\rho} + \rho^2 \log \frac{1}{\beta}}{\alpha^2 \rho^4}\right)$
- Replicability bucket size $\tau \leq O(\frac{\alpha\rho}{\ln|H|})$

Algorithm:

```
1. Draw a labeled sample $S \sim D^m$ and compute $err_S(f)$ for every $f \in H$.
2. Replicably output initialization $v_{\text{init}} \in [OPT, OPT + \alpha/2]$ (see Algorithm 35)
3. Select random threshold $v \leftarrow_r \{v_{\text{init}} + \frac{3}{2}\tau, v_{\text{init}} + \frac{5}{2}\tau, \ldots, v_{\text{init}} + \alpha/4 - \tau/2\}$
4. Randomly order all $f \in H$
return Output the first hypothesis $f$ in the order s.t. $err_S(f) \leq v$.
```

We note that Step 2, estimating OPT, follows essentially the same argument as the basic replicable statistical query algorithm of [ILPS22]. We give the argument in Section 5.7 for completeness.

We note that while Algorithm `rFiniteLearner` is a replicable agnostic PAC learner, it is not quite sufficient to prove Theorem 5.5.13 due to its poor dependence on ρ. We'll see in the next section how to obtain the stated parameters by separately amplifying `rFiniteLearner` starting from good constant replicability.

Analysis

We'll start by proving the following weaker bound for our intermediate learner.

Theorem 5.5.14 (Intermediate Learnability of Finite Classes)**.** *Let H be any finite concept class. Algorithm* `rFiniteLearner` *is a (proper) agnostic replicable learning algorithm for H with sample complexity:*

$$m(\rho,\alpha,\beta) \leq O\left(\frac{\log^2|H|\log(\frac{1}{\rho})+\rho^2\log\frac{1}{\beta}}{\alpha^2\rho^4}\right).$$

In the realizable setting, the (α,β)-dependence can be improved to:

$$m(\rho,\alpha,\beta) \leq O\left(\frac{\log^2|H|\log(\frac{1}{\rho})+\rho^4\log\frac{1}{\beta}}{\alpha\rho^4}\right).$$

The main challenge in Theorem 5.5.14 is proving replicability. (Accuracy and failure probability are essentially immediate from standard uniform convergence arguments.) To this end, note that the randomness r used by `rFiniteLearner` is largely broken into three parts: estimating OPT, choosing a random threshold, and ordering the concepts in H. We'll focus first on the latter two, where the choice of v restricts H to two subsets H_1 and H_2 (those with empirical error at most v), depending on input samples S_1 and S_2. We first appeal to the classical observation of Broder [Bro97] to argue that as long as the symmetric difference of H_1 and H_2 are small, outputting the first concept from these sets (according to the random ordering) is a replicable procedure.

Observation 5.5.15. *Let $O(H,r)$ be a random ordering of concept class H. Let $\emptyset \subset H_1, H_2 \subseteq H$, and let f_1 and f_2 be the first elements of H_1 and H_2 respectively according to $O(H,r)$. Then $\mathbf{Pr}_r[f_1 \neq f_2] = \frac{|H_1 \Delta H_2|}{|H_1 \cup H_2|}$, where Δ denotes the symmetric difference.*

The key to proving replicability is then to observe that most choices of v induce small symmetric difference between the corresponding H_1 and H_2. Namely, the idea is to observe that for any fixed joint distribution D, intervals

$$I_0 = [OPT, OPT+\tau], \ldots, I_{\alpha/(2\tau)} = [OPT+\alpha/2-\tau, OPT+\alpha/2],$$

and corresponding threshold positions $v_i = OPT + \frac{(2i+1)}{2}\tau$, the sets

$$H_1^{(i)} = \{h \in H : err_{S_1}(h) \leq v_i\}, \quad H_2^{(i)} = \{h \in H : err_{S_2}(h) \leq v_i\}$$

are close for most choices of v_i, S_1, and S_2. To adjust for the fact that we don't know the value of OPT, we will in fact prove something slightly more general that allows our starting point to range anywhere from OPT to $OPT+\alpha/2$.

Lemma 5.5.16. *Let $v_{init} \in [OPT, OPT+\alpha/2]$ and $\tau \leq O\left(\frac{\alpha\rho^2}{\log|H|}\right)$ a parameter that divides $\alpha/4$. Define the intervals*

$$I_0 = [v_{init}, v_{init}+\tau),\ I_1 = [v_{init}+\tau, v_{init}+2\tau),\ \ldots\ ,\ I_{\frac{\alpha}{4\tau}} = \left[v_{init}+\frac{1}{4}\alpha-\tau, v_{init}+\frac{1}{4}\alpha\right]$$

and corresponding thresholds $v_i = v_{init} + \frac{(2i+1)}{2}\tau$, and let

$$H_1^{(i)} = \{h \in H : err_{S_1}(h) \leq v_i\}, \quad H_2^{(i)} = \{h \in H : err_{S_2}(h) \leq v_i\}$$

denote the hypotheses with empirical error at most v_i across two independent samples S_1 and S_2 of size $O(\frac{\log\rho^{-1}}{\tau^2})$. Then with probability at least $1-\rho/4$, a uniformly random choice of $i \in [\frac{\alpha}{4\tau}]$ satisfies:

$$\frac{|H_1^{(i)} \Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|} \leq \rho/4.$$

Proof. For convenience of notation, let $|I_i|$ denote the number of hypotheses whose true risk lies in interval I_i, and $|I_{[i]}|$ the number of hypotheses in intervals up through I_i. We call a threshold v_i "bad" if any of the following conditions hold.

1. The ith interval has too many elements:

$$|I_i| > \frac{\rho}{30}|I_{[i-1]}|.$$

2. The number of elements beyond I_i increases too quickly:

$$\exists j \geq 1 : |I_{i+j}| \geq e^j |I_{[i-1]}|.$$

and "good" otherwise. We will argue the following two claims.

1. If v_i is a good threshold, then $H_1^{(i)}$ and $H_2^{(i)}$ are probably close

$$\mathop{\mathbf{Pr}}_{S_1,S_2}\left[\frac{|H_1^{(i)} \Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|} \leq \frac{\rho}{4}\right] \geq 1 - \frac{\rho}{8}.$$

2. At most a $\frac{\rho}{8}$ fraction of thresholds are bad.

Since we pick a threshold uniformly at random, it is good with probability at least $1 - \rho/8$ and a union bound gives the desired result.

It remains to prove the claims. For the first, observe that for any fixed hypothesis h with true risk $err_D(h) \in I_{i+j}$, the probability that the empirical risk of h is less than v_i is at most

$$\mathbf{Pr}[err_S(h) \leq v_i] \leq e^{-\Omega(j^2\tau^2|S|)} \tag{5.16}$$

by a Chernoff bound. Let x_i denote the variable which counts the number of hypotheses with true risk beyond I_i that cross the threshold v_i empirically. If v_i is "good," we can bound $\mathbb{E}[x_i]$ by

$$\mathbb{E}[x_i] \leq |I_{[i-1]}| \sum_{j>0} e^{-\Omega(j^2\tau^2|S|-j)} \leq \frac{\rho^2}{2000}|I_{[i-1]}|$$

for our choice of $|S|$. Markov's inequality then promises

$$\mathbf{Pr}\left[x_i \geq \frac{\rho}{30}|I_{[i-1]}|\right] \leq \frac{\rho}{64}.$$

On the other hand, the probability any hypothesis in $I_{[i-1]}$ crosses v_i is at most $e^{-\Omega(\tau^2|S|)}$, so similarly the probability that more than a $\frac{\rho}{30}$ fraction of such hypotheses cross v_i is at most $\frac{\rho}{64}$. Finally, since v_i is 'good,' I_i itself contributes at most $\frac{\rho}{30}|I_{[i-1]}|$ hypotheses that cross the threshold in the worst case, so in total we have that with probability at least $1-\frac{\rho}{32}$, at most $\frac{\rho}{10}|I_{[i-1]}|$ hypotheses cross the threshold in either direction. Considered over two runs of the algorithm, this implies that with probability at least $1-\frac{\rho}{16}$, $|H_1^{(i)}\Delta H_2^{(i)}|$ cannot be too big

$$|H_1^{(i)}\Delta H_2^{(i)}| \leq \frac{\rho}{5}|I_{[i-1]}|.$$

Furthermore, since the probability that more than a $\frac{\rho}{30}$ fraction of hypotheses in $I_{[i-1]}$ cross v_i is at most $\frac{\rho}{64}$, we also have that $|H_1^{(i)} \cup H_2^{(i)}|$ cannot be too small:

$$|H_1^{(i)} \cup H_2^{(i)}| \geq \left(1-\frac{\rho}{15}\right)|I_{[i-1]}|$$

with probability at least $1-\frac{\rho}{64}$. Thus altogether a union bound gives

$$\mathop{\mathbf{Pr}}_{S_1,S_2}\left[\frac{|H_1^{(i)}\Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|} \leq \frac{\rho}{4}\right] \geq 1-\frac{\rho}{8}$$

as desired.

Finally, we need to show that almost all thresholds are good. To see this, first observe that since $v_{\text{init}} \geq OPT$, $|I_{[i]}| > 0$ for all $i \geq 0$. To count the number of bad thresholds, let $i_1 \geq 1$ be the position of the first bad threshold, and t_1 denote the largest index such that $i_1 + t_1$ fails a condition. Define i_j and t_j recursively as the first bad threshold beyond $i_{j-1}+t_{j-1}$ and its corresponding latest failure. Observe that by construction, any interval that does not lie in any $[i_j.i_j + t_j]$ is good, so there are at most $\sum t_j$ bad thresholds.

Let ℓ denote the final index of the above greedy process. By definition of a bad interval, each t_j multiplicatively increases the number of total hypotheses from $I_{[i_j]}$ by at least $\left(1+\frac{\rho}{30}\right)^{t_j}$. Since $|I_0| \geq 1$ and the total number of hypotheses is $|H|$ by definition, we may therefore write:

$$|H| \geq |I_{[i_\ell + t_\ell]}| \geq \left(1+\frac{\rho}{30}\right)^{\sum\limits_{j=1}^{\ell} t_j}$$

and thus that the total number of bad intervals is at most

$$\sum_{j=1}^{\ell} t_j \leq O\left(\frac{\log(|H|)}{\rho}\right).$$

Since we have chosen τ such that the total number of intervals altogether is at least $\Omega\left(\frac{\log(|H|)}{\rho^2}\right)$, the appropriate choice of constant gives that at most a $\rho/8$ fraction are bad as desired. □

To complete the argument, it is enough to show we can find a good starting point v_{init}.

Lemma 5.5.17. *There exists a ρ-replicable algorithm over $O\left(\frac{\log(\frac{|H|}{\rho\beta})}{\rho^2\alpha^2}\right)$ samples that outputs a good estimate of OPT with high probability:*

$$\mathbf{Pr}_{r,S}\left[\mathscr{A}(S) \in [OPT, OPT+\alpha/2]\right] \geq 1-\beta$$

Proving this Lemma largely follows from prior techniques but is a bit tedious, so we leave the proof for Section 5.7. With these tools in hand, we are finally ready to prove Theorem 5.5.13.

Proof of Theorem 5.5.14. We first show `rFiniteLearner` is ρ-replicable. `rFiniteLearner` starts by running a replicable subroutine (with parameters $\rho' = \rho/2$ an $\beta' = \beta/2$) to find an estimate for OPT. Using new (independent) randomness, it then selects a threshold v_i and a random ordering over H, and outputs the first hypothesis in $H^{(i)} = \{h : R_{\text{emp}}(h,S) \leq v_i\}$. By Lemma 5.5.16 and Observation 5.5.15, this latter process is $\rho/2$-replicable. By composition of replicability, the entire algorithm is therefore ρ-replicable as desired.

Correctness of `rFiniteLearner` follows from standard uniform convergence type arguments. In particular, by our choice of $|S|$, any hypothesis with empirical risk at most $OPT + \alpha/2$ has true risk less than $OPT + \alpha$ with probability at least $1 - \beta/2$. Furthermore, as long as our estimation of OPT is successful (which occurs with probability at least $1 - \beta/2$), we always output such a hypothesis. Thus altogether we output a hypothesis with true error at most $OPT + \alpha$ with probability at least $1 - \beta$ as desired.

Finally, we need to argue that the dependence on α can be improved to linear in the realizable setting. Note that in this case, we can simply set v_{init} to 0, and ignore the estimation of OPT. The improvement then follows immediately from noting that when $OPT = 0$, a standard Chernoff bound improves Equation (5.16) to

$$\mathbf{Pr}[err_S(h) \leq v_i] \leq e^{-\Omega(\frac{j^2\tau|S|}{(i+j)})} \leq e^{-\Omega(\frac{j^2\tau^2|S|}{\alpha})}.$$

Similarly, only $O(\frac{\log\frac{|H|}{\beta}}{\alpha})$ examples are needed to ensure hypotheses with $O(\alpha)$ empirical risk have $O(\alpha)$ true risk with high probability, and the rest of the proof follows as in the agnostic case. $\square$

Finally, we amplify the above to prove Theorem 5.5.13.

Proof of Theorem 5.5.13. Our amplification algorithm is a modification of the original technique introduced in [ILPS22], designed to take advantage of the fact that the dependence on β (failure) and ρ (replicability) are highly unbalanced in learning tasks. Draw $k = O(\log(1/\rho))$ random strings $r_1, \ldots, r_k$, and consider the distributions $\{D_i\}_{i=1}^k$ generated by running `rFiniteLearner`(r_i, S) with parameters $\rho' = .01$ and $\beta' = \beta \cdot \text{poly}(\rho)$ on a large enough sample S. The idea is to argue that with good probability over the choice of random strings r, at least one of these distributions has an $\Omega(1)$-heavy-hitter (which is also a good hypothesis with extremely high probability). Roughly speaking, we can then use the heavy hitters algorithm of [ILPS22] across these distributions to $\rho/2$-replicably output a good hypothesis, and union bound over all applications to argue correctness of the final output.

Let's formalize this argument. First, observe since our setting of `rFiniteLearner` is .01-replicable, at least 90% of the random strings have a 'canonical element,' i.e. one that appears across at least 90% of random samples. Call such strings good, and observe that any good string r_i corresponds to a distribution $\mathscr{D}_i$ with a .9-heavy-hitter by construction. Over a random choice of $O(\log \rho^{-1})$ such strings, the former guarantee then promises at least one of these strings is good with probability greater than $1-\rho/4$ and therefore that at least one distribution in $\{D_i\}_{i=1}^k$ has a .9-heavy-hitter. With this in mind, we now appeal to the heavy-hitter algorithm of [ILPS22], which draws $O(\frac{\log \rho^{-1}}{\rho^2})$ samples from a distribution to replicably output a list of all $\Omega(1)$-heavy-hitters.[15] To make our entire process ρ-replicable, we will run the above process for $\rho' = O(\frac{\rho}{\log \rho^{-1}})$. To this end, we draw samples $S_1, \dots, S_t$ for $t = O(\frac{\log^3 \rho^{-1}}{\rho^2})$ and generate t corresponding samples from each $\{D_i\}_{i=1}^k$ (re-using S_j between distributions), which we use to run [ILPS22]'s heavy-hitters algorithm. Union bounding over all applications, this process is $\rho/4$-replicable, and with probability at least $1-\rho/4$ outputs a non-empty list of hypotheses. Finally, we break in to one of two cases. If the result list is indeed non-empty, simply output a random element of the list (a fully replicable procedure). Otherwise, run `rFiniteLearner` on a fresh random string and sample, and output the result.

Finally, we argue replicability and correctness of the above process. First, note the output list is non-empty with probability at least $1-\rho/4$, and two independent samples produce the same list (with fixed randomness) with probability at least $1-\rho/2$ by replicability of the repeated heavy hitter process discussed above. Therefore the entire process is ρ-replicable as desired. For

[15]We note the technique actually outputs a list of some weight close to c, but this is largely irrelevant in our setting where c (and the shift in c) are constant.

correctness, note that we have used at most $\mathrm{poly}(\rho^{-1})$ instances of `rFiniteLearner`. Recall that we set the failure probability of `rFiniteLearner` to be very small, with $\beta' = \beta \cdot \mathrm{poly}(\rho^{-1})$. Since each individual application of `rFiniteLearner` fails with probability less than β', union bounding over all applications of the algorithm implies every output hypothesis is 'good' (within α of OPT) with probability at least $1-\beta$. Since we only output hypotheses generated by this process, the probability of outputting a bad hypothesis is then less than β as desired.

Altogether, the sample complexity of the above algorithm is given by the size of samples

$$t|S_i| = O\left(\frac{\log^3 \rho^{-1}}{\rho^2} \cdot \frac{\log^2 |H| + \log \frac{1}{\rho\beta}}{\alpha^2}\right),$$

or

$$O\left(\frac{\log^3 \rho^{-1}}{\rho^2} \cdot \left(\frac{\log |H| + \log \frac{1}{\rho\beta}}{\alpha} + \log^2 |H|\right)\right)$$

in the realizable case.

□

5.6 Applications

In this section we take advantage of our reductions between notions of stability to resolve (or otherwise make progress on) several open problems in the algorithmic stability literature.

5.6.1 Item-level to User-level Privacy Transformation

The original motivation of introducing the definition of pseudo-global stability in [GKM21] was to come up with PAC learning algorithms in the example-rich, user-level privacy setting. In this setting, there are a number of users with many samples (the regime we will be interested in is when there are a few users who have enough samples to solve the problem for themselves). The motivation behind this setting is to leverage the data of example-rich users to obtain statistical insights without compromising their privacy. Such statistical analyses could then be released and used widely, even by users who didn't have as much data.

The technique of [GKM21] involves coming up with pseudo-globally stable algorithms for PAC learning, and then having each user run a pseudo-globally stable algorithm with the same coins. Then, you can privately identify a heavy hitter among the outputs of the users (such a heavy hitter exists with high probability because of the property of pseudo-global stability), and since changing an entire user's sample will affect only one output, this procedure will be user-level differentially private. One open question raised in their paper was whether their techniques could be extended beyond the PAC setting.

Our argument that pseudo-global stability and differential privacy are two sides of the same coin answers this question in the affirmative, and has the additional benefit of eliminating the need to cleverly design pseudo-globally stable (replicable) algorithms. We showed previously that item-level differentially private algorithms can be compiled into replicable algorithms with only a quadratic overhead in sample complexity (See Sections 5.3.2 and 5.3.3). Hence, our results allow for a general transformation from item-level to user-level privacy for statistical tasks in the example-rich setting; each user applies correlated sampling to the same item-level differentially private algorithm applied to their specific sample, and then a heavy hitter is identified via differentially private selection.

Theorem 5.6.1. *There are universal constants $c, K > 0$ such that the following holds. Let $\mathscr{T}$ be a statistical task with a finite output space. Given a $(0.1, \frac{c}{n^3})$-item level differentially private algorithm $\mathscr{A}$ that solves $\mathscr{T}$ using n samples and with failure probability β, for every $1 \geq \varepsilon, \delta > 0$, there exists an (ε, δ)-user level differentially private algorithm $\mathscr{A}_u$ that solves $\mathscr{T}$ with failure probability $O(\beta \log \frac{1}{\beta})$, when given access to the data of $O(\frac{\log 1/\beta}{\varepsilon} \log \frac{\log 1/\beta}{\delta})$ users, each of whom have at least $Kn^2 \log(1/\beta)$ examples.*

Proof. Firstly, we can amplify the privacy parameters of $\mathscr{A}$ by subsampling. By Lemma 5.2.9, the algorithm $\mathscr{A}'$ that, given $m = Kn^2$ samples, subsamples n items without replacement and runs $\mathscr{A}'$

on the result is $(1/\sqrt{Km}, cK/m^2)$-differentially private. Moreover, when run on inputs consisting of i.i.d. samples from a distribution D, the output of $\mathscr{A}'$ is identically distributed to that of $\mathscr{A}$, so $\mathscr{A}'$ also solves $\mathscr{T}$ with m examples and failure probability β.

For a sufficiently large constant K and sufficiently small constant c, Corollary 5.3.18 then implies that $\mathscr{A}'$ gives rise to a c-replicable algorithm solving $\mathscr{T}$ with m examples and failure probability β.

Now, consider Algorithm 24 adapted to the user-level setting as follows (with number of users as specified in the theorem): instead of partitioning a centralized sample (as done in that algorithm), each of the users applies algorithm $\mathscr{A}'$ to their own set of $O(\log(1/\beta))$ i.i.d. samples (with the same set of $O(\log(1/\beta))$ different coins—this can be achieved through a common random string that they share). Then, the outputs are sent to a central server, and $(\varepsilon' = \frac{\varepsilon}{C \log 1/\beta}, \delta' = \frac{\delta}{2 \log 1/\beta})$-DP selection is then applied to choose a heavy hitter (as discussed in Algorithm 24). Let's call this algorithm $\mathscr{A}_u$.

The accuracy guarantees proved for Algorithm 24 give us that the failure probability of this Algorithm $\mathscr{A}_u$ is at most $O(\beta \log 1/\beta)$. Hence, we are left to argue privacy. Note that changing a single user's sample can change at most $C \log 1/\beta$ outputs to which the (ε', δ')-DP selection algorithm is applied to. Hence, by group privacy (Lemma 5.2.8), we have that $\mathscr{A}_u$ is $(\varepsilon' \log 1/\beta, \delta' \frac{e^{\varepsilon' \log 1/\beta} - 1}{e^{\varepsilon'} - 1})$-user level differentially private. Substituting the value of ε' and δ' then gives an (ε, δ)-user level private algorithm. This completes the proof. $\square$

5.6.2 Parameter Amplification for Differential Privacy and Perfect Generalization

The equivalence between replicability and differential privacy gives us the first generic amplification theorem for the δ parameter of approximate differential privacy for general statistical tasks. Prior to our work, it was known that the ε parameter could be amplified algorithmically and efficiently. That is, using random sampling (Lemma 5.2.9), one can improve an $(\varepsilon_0, \delta_0)$-

differentially private algorithm to a $(p\varepsilon_0, p\delta_0)$-differentially private one with an $O(1/p)$ blowup in the sample complexity. However, this technique is unable to improve the δ parameter of such an algorithm asymptotically as a function of the number of samples n, e.g., from $\delta(n) = 1/n^{10}$ to $\delta'(n) = \exp(-n^{0.99})$.

The recent characterization of private PAC learnability in terms of Littlestone dimension [ALMM19, BLM20, GGKM21] implies that such an amplification of δ is (at least, in principle) possible for private PAC and agnostic learning and for private query release. Given a target class $\mathscr{C}$ in one of these settings, the existence of a $(\varepsilon = 0.1, \delta = O(1/n^2 \log n))$-differentially private algorithm using a finite number of samples n implies that $\mathscr{C}$ has some finite Littlestone dimension d. This in turn implies that, for every $\varepsilon, \delta > 0$, there is an (ε, δ)-differentially private agnostic PAC learning algorithm for $\mathscr{C}$ using $\mathrm{poly}(d, 1/\varepsilon, \log(1/\delta))$ samples and a private query release algorithm for $\mathscr{C}$ using $\mathrm{poly}(2^{2^d}, 1/\varepsilon, \log(1/\delta))$ samples. Unfortunately, the first part of this argument is non-constructive, and in the worst-case, leads to a final algorithm using a number of samples that is an exponential tower in $\Omega(n)$! [BLM20] posed the open question of whether such amplification could be done algorithmically, even for the special case of private PAC learning.

Our approach is to first convert a differentially private algorithm with weak parameters to a replicable one. We may then use the fact that replicable algorithms can be converted back to differentially private ones with excellent privacy parameters. Altogether we obtain a constructive amplification theorem that achieves only a modest blowup in sample complexity, and which applies to general statistical tasks with finite output spaces.

Theorem 5.6.2. *There is a universal constant $c > 0$ such that the following holds. Let $\mathscr{T}$ be a statistical task with a finite output space. Suppose there is an $(\varepsilon = 0.1, \delta = c/n^3)$-differentially private algorithm that solves $\mathscr{T}$ using n samples and with failure probability β. Then for every*

$\varepsilon, \delta > 0$*, there exists an* (ε, δ)*-differentially private algorithm solving* $\mathscr{T}$ *using*

$$O\left(\frac{\log(1/\delta)\log(1/\beta)}{\varepsilon} + \log^2(1/\beta)\right) \cdot n^2$$

samples and with failure probability $O(\beta \log 1/\beta)$.

Proof. Let A be a $(0.1, c/n^3)$-differentially private algorithm solving $\mathscr{T}$ with n samples and failure probability $O(1/\log(1/\beta))$. By Lemma 5.2.9, the algorithm A' that, given $m = Kn^2$ samples, subsamples n items without replacement and runs A on the result is $(1/\sqrt{Km}, cK/m^2)$-differentially private. Moreover, when run on inputs consisting of i.i.d. samples from a distribution P, the output of A' is identically distributed to that of A, so A' also solves $\mathscr{T}$ with m samples and failure probability β.

For a sufficiently large constant K and sufficiently small constant c, Corollary 5.3.18 implies that A' gives rise to a c-replicable algorithm solving $\mathscr{T}$ with m samples and failure probability β. Applying Theorem 5.3.1 thus results in an (ε, δ)-differentially private algorithm solving $\mathscr{T}$ with

$$m \cdot O\left(\frac{\log(1/\delta)\log(1/\beta)}{\varepsilon} + \log^2(1/\beta)\right)$$

samples and failure probability $O(\beta \log 1/\beta)$. □

We can also show a similar amplification of the parameters for perfect generalization, indeed, even from one-way perfect generalization to perfect generalization itself.

Theorem 5.6.3. *There is a universal constant* $c > 0$ *such that the following holds. Let* $\mathscr{T}$ *be a statistical task with a finite output space. Suppose there is an* $(\beta' = 10^{-5}, \varepsilon = 10^{-5}, \delta = 10^{-5})$*-one-way perfectly generalizing algorithm that solves* $\mathscr{T}$ *using n samples and with failure probability* β*. Then for every* $\varepsilon, \delta > 0$*, there exists an* $(\delta, \varepsilon, \delta)$*- perfectly generalizing algorithm solving* $\mathscr{T}$ *using*

$$O\left(n \cdot \frac{\log(1/\varepsilon)\text{poly}\log(1/\delta)}{\varepsilon^2}\right)$$

samples and with failure probability $O(\delta)+\sqrt{\beta}\log(1/\delta)$.

Proof. Let A be a $(10^{-5},10^{-5},10^{-5})$-perfectly generalizing algorithm solving $\mathscr{T}$ with n samples and failure probability β. By Lemma 5.3.17, there is a (0.0001)-replicable algorithm solving $\mathscr{T}$ with the same failure probability and the same number of samples. By Claim 5.2.12 and Theorem 5.3.19, there exists a $(\delta^2/4,\varepsilon,\delta^2/4)$-sample perfectly generalizing algorithm solving $\mathscr{T}$ using $O(n\cdot\frac{\log(1/\varepsilon)\text{poly}\log(1/\delta)}{\varepsilon^2})$ samples with failure probability $O(\delta)+\sqrt{\beta}\log(1/\delta)$. Lemma 5.3.6 converting sample perfect generalization to perfect generalization gives the result. □

5.6.3 An Agnostic-to-Realizable Reduction for Structured Distributions

One of the main running examples throughout this work (and indeed a focal point in [ILPS22, GKM21] as well) is the PAC-learning paradigm. Traditionally, PAC-learning has two main settings, *realizable* learning (where the adversary must choose a hypothesis in the class), and the *agnostic* setting (which allows an arbitrary adversary). It is a well known fact in the study of traditional statistical learning that realizable and agnostic learning are equivalent up to polynomial blowup in sample complexity [VC74, BEHW89, Hau92]. Furthermore, an analog of this fact holds for most supervised paradigms (see e.g. [BI91, BLW96, Lon01, BDPSS09, DMY16, MHS19, AHHM21]), including privacy [BNS16b, ABMS20]. This was originally shown by Beimel, Nissim, and Stemmer [BNS16b], who gave a sample-efficient agnostic-to-realizable reduction for approximately differentially private PAC-learning. Their result has since been used extensively (see e.g. [BNS16c, BMNS19, BMA19, ABMS20, BLM20]), and is often used to justify focus on the realizable setting.

While certainly impactful, Beimel, Nissim, and Stemmer's reduction (and later improvements on the same [ABMS20, BLM20]) are complicated and limited in application. Like the results that came before them (in the traditional setting), their techniques rely heavily on *uniform convergence*, and therefore always incur a cost in VC dimension of the class (or analogously in $\log|H|$ in many settings we consider). Such bounds are typically only useful in the *distribution-free* setting, where the adversary is free to choose arbitrary (often strange, combinatorial) distributions over the data that don't appear in practice. Outside of such cases, it is typically possible to learn in many fewer samples than VC dimension would predict (see e.g. [NK19]), so it is reasonable to ask whether this efficiency can be generically maintained in the agnostic setting. Towards this end, Hopkins, Kane, Lovett, and Mahajan [HKLM22] recently gave a more generic reduction independent of VC dimension, but their techniques do not adapt directly to the private setting, which was left as an open problem in their work.

We resolve this problem (at least in finite domains) via reduction to and from replicability: agnostic private learning requires only a small polynomial blowup over the realizable case that is independent of class-size, even under arbitrary distributional assumptions. With this in mind, we briefly introduce the *distribution-family* variant of the PAC-model, which first appeared (implicitly) in seminal work of Benedek and Itai [BI91] on learning under fixed distributions. We use bold font below to highlight the differences from the standard model.

Definition 5.6.4 (Distribution-Family Model [BI91])**.** *A learning problem is defined by a hypothesis class H **and family of distributions $\mathscr{D}$** over the instance space $\mathscr{X}$. We say an algorithm $\mathscr{A}$ is an (α,β)-accurate Agnostic learner for the hypothesis class $(H,\mathscr{D})$ if for all distributions D over input, output pairs **whose marginal $\mathbf{D}_{\mathscr{X}} \in \mathscr{D}$**, $\mathscr{A}$ on being given a sample of size m drawn i.i.d. from D outputs a hypothesis h such that with probability greater than or equal to $1-\beta$ over the randomness of the sample and the algorithm,*

$$\mathrm{err}_D(h) \leq \inf_{f\in H} \mathrm{err}_D(f) + \alpha,$$

*where $\mathrm{err}_D(h) = \mathbf{Pr}_{(x,y)\in D}[h(x) \neq y]$. When the adversary is additionally restricted to choosing D s.t. $\inf_{f\in H} \mathrm{err}_D(f) = 0$, we call the problem **realizable**.*

We give the first private agnostic-to-realizable reduction from the distribution-family model, and in general the first reduction with no reliance on uniform convergence or VC dimension.

Theorem 5.6.5. *Let $(H,\mathscr{D})$ be a hypothesis class that is $(1,\frac{1}{poly(n)})$-privately $(\alpha,0.01)$-PAC learnable in $n=n(\alpha)$ samples in the realizable setting. Then $(H,\mathscr{D})$ is $(1,\frac{1}{poly(m)})$-privately (α,β)-agnostically learnable in*

$$m(\alpha,\beta) \le O\left(\frac{n^2\alpha^2+\log^3(\Pi_H(cn^2))}{\alpha^2}\log\beta^{-1}\log\left(\frac{n\log\beta^{-1}}{\alpha}\right)\right)$$

samples for some universal constant $c>0$.

In the statement of Theorem 5.6.5, $\Pi_H(n)$ denotes the growth function of (X,H). The growth function captures the maximum number of labelings functions from H can induce on samples of size n, and is at most $n^{O(d)}$ for classes with VC dimension d. Moreover, since $\Pi_H(cn^2) \le 2^{cn^2}$, this means agnostic learning experiences at most a polynomial blowup over the realizable setting:

Corollary 5.6.6. *Let $(H,\mathscr{D})$ be a hypothesis class that is $(1,\frac{1}{poly(n)})$-privately $(\alpha,0.01)$-PAC learnable in $n=n(\alpha)$ samples in the realizable setting. Then $(H,\mathscr{D})$ is $(1,\frac{1}{poly(m)})$-privately (α,β)-agnostically learnable in*

$$m(\alpha,\beta) \le \tilde{O}\left(\frac{n^6\log\beta^{-1}}{\alpha^2}\right)$$

samples.

At a high level, the proof of this result is (comparatively) simple. Given a realizable private learner $\mathscr{A}$, we will transform $\mathscr{A}$ into a replicable learner, apply a variant of [HKLM22]'s agnostic-to-realizable reduction, and finally lift the resulting agnostic learner back to differential privacy. The main challenge lies in adapting [HKLM22] to the replicable setting, which is roughly done via the following procedure (see Algorithm 35):

1. **Sample:** Draw an unlabeled sample $S \sim D^n$, and random string r
2. **Generate Candidates:** Run $\mathscr{A}$ on all labelings of S with internal randomness r
3. **Prune:** Using fresh samples, remove any high error candidates

The result then follows from replicably outputting a heavy hitter of this procedure.

List Heavy-Hitters

It is this final step, it turns out, that contains most of the subtlety in this reduction. While estimating heavy hitters of a given distribution is a core subroutine in many replicable algorithms (used, e.g., in Theorem 5.5.13), and was studied in [ILPS22], our setting is more challenging since our goal is to output a heavy hitter from a distribution over *lists*, where the list size may be exponential in the desired parameters. In this section, we show how similar arguments used for our efficient finite learner can also be used to output a heavy hitter with cost only *polylogarithmic* in the list size.[16]

More formally, let Ω be a finite set, and $\mathscr{D}$ a distribution over subsets of H. We call $h \in H$ an η-heavy-hitter of $\mathscr{D}$ if

$$\mathbf{Pr}_{S\sim\mathscr{D}}[h \in S] \geq \eta.$$

We prove it is possible to replicably output a heavy hitter with complexity scaling that is only polylogarithmic in the largest set supported by $\mathscr{D}$.

Theorem 5.6.7. *For any finite set Ω, $\rho, \eta, \beta > 0$, and distribution $\mathscr{D}$ over subsets of Ω with an η-heavy-hitter, there exists a ρ-replicable algorithm $\mathscr{A}$ with the following guarantees:*

1. *$\mathscr{A}$ outputs a $\frac{\eta}{2}$-heavy-hitter with probability at least $1-\beta$*
2. *$\mathscr{A}$ uses at most $O\left(\frac{\log^2 \frac{|\mathscr{D}|\log\frac{1}{\rho\beta}}{\eta}\log\frac{1}{\rho}+\log\frac{1}{\rho\beta}}{\eta^2\rho^2}\log^3\frac{1}{\rho}\right)$ samples from $\mathscr{D}$,*

where $|\mathscr{D}|$ is the maximum size subset supported by $\mathscr{D}$.

We note that it is easy to modify this result to remove the assumption that $\mathscr{D}$ has a heavy hitter (the algorithm instead outputs '$\perp$' in this case, or can test for the heaviest element), but the

[16]We note that a similar result also appears implicitly in [GKM21, Theorem 20], albeit with worse sample complexity.

simpler version above is sufficient for our applications. We now give the algorithm itself, which combines [ILPS22]'s heavy hitters with our thresholding technique for finite learning.

Algorithm 34. Replicable List Heavy Hitters

Result: Replicably outputs a heavy hitter

Input: Distribution $\mathscr{D}$ over subsets of universe Ω (Sample Access)

Parameters:

- Replicability, confidence, and heaviness $\rho, \beta, \eta > 0$
- Sample sizes $t_1 = O\left(\frac{\log(\frac{|\mathscr{D}|}{\rho\beta})}{\eta}\right)$, $t_2 = O\left(\frac{\log^2 \frac{|\mathscr{D}| \log \frac{1}{\rho\beta}}{\eta} \log \frac{1}{\rho} + \log \frac{1}{\beta}}{\eta^2 \rho^4}\right)$
- Threshold accuracy $\tau \leq O(\frac{\rho\eta}{\log|\mathscr{D}|})$

Algorithm:

1. Sample t_1 subsets $C \sim \mathscr{A}$, and call their union T.
2. Sample an additional t_2 subsets $C \sim \mathscr{A}$, and call their collection $S = \{C_i\}$.
3. For each $t \in T$, let $\hat{p}_t$ denote its empirical measure over S:
$$\hat{p}_t = \frac{1}{|S|} |\{C \in S : t \in C\}|.$$
4. Choose a random threshold $v \in \{\eta/4 + 2\tau, \eta/4 + 6\tau, \ldots 3\eta/4 - 2\tau\}$
5. Randomly order Ω

return first $t \in T$ with respect to the order satisfying $\hat{p}_t \geq v$

It is not hard to see this algorithm succeeds via the same analysis as for Theorem 5.5.13.

Proof. By a Chernoff and union bound, we first note that with probability at least $1 - \beta\rho/4$, T

contains every η-heavy-hitter of $\mathscr{D}$.

Similar to the proof of Theorem 5.5.13, we consider intervals of the form

$$I_0 = [\eta/4, \eta/4 + 4\tau], \dots, I_{1/(2\tau)} = \left[\frac{3\eta}{4} - 4\tau, \frac{3\eta}{4}\right],$$

with corresponding threshold positions $v_i = \eta/4 + \frac{(2i+1)}{2}\tau$, and the sets

$$H_1^{(i)} = \{t \in T : \hat{p}(t, S_1) \leq v_i\}, \quad H_2^{(i)} = \{t \in T : \hat{p}(t, S_2) \leq v_i\}.$$

In the proof of Theorem 5.5.13, we argued that replicability followed from bounding the quantity

$$\frac{|H_1^{(i)} \Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|}$$

with high probability, as this promised that choosing the first element from a joint random ordering of Ω usually gives the same answer over $H_1^{(i)}$ and $H_2^{(i)}$. Here we need to be slightly more careful, in that we need to ensure not only that the same element is chosen, but also that it is truly an $\eta/4$-heavy-hitter. This ensures replicability despite the fact that our set T depends on samples, because we are promised that all $\eta/4$-heavy-hitters lie in T except with probability $1 - \frac{\beta\rho}{4}$ (and therefore have no dependence T itself).

Thankfully, this is already implicit in the proof of Lemma 5.5.16, since it is actually proved that, with high probability, the number of elements of T that cross threshold v_i is at most $O(\rho|I_{[i-1]}|)$, where we recall $|I_{[i-1]}|$ denotes the number of elements with true weight in buckets $I_1, \dots, I_{i-1}$. This followed from the fact that any element $t \in T$ whose true weight lay in the jth interval for $j > i$ satisfied:

$$\mathbf{Pr}[\hat{p}(t, S_j) \leq v_i] \leq e^{-\Omega(j^2\tau^2|S_j|)},$$

which remains true in this setting for our choice of $|S|$ by Chernoff. As such, our full process remains ρ-replicable for the correct choice of constants as desired. Furthermore, correctness holds with probability at least $1-\beta$, since all weight estimates are correct up to $\eta/4$. Finally, to get the correct dependence on ρ we simply apply the amplification technique used in the proof of Theorem 5.5.13.

□

We note that this result is similar to the pseudo-globally stable learner of [GKM21], which also uses a method of replicably finding a heavy hitter from a distribution on lists. Their algorithm uses a variant of the exponential mechanism instead of random thresholding, and loses polynomial factors over our bound as a result. Both [GKM21] and our algorithm have the downside of only working over finite universes (or more generally in settings where correlated sampling is possible). On the other hand, the problem can be solved *privately* without this assumption. This raises a natural question: does list heavy hitters give an exponential separation between privacy and replicability over infinite domains?[17]

Before moving on, we note the following immediate implication of list heavy hitters for learning classes with low Littlestone dimension, giving a moderate improvement over the analogous result of [GKM21].

Corollary 5.6.8. *Let (X,H) be a class with Littlestone dimension d. Then the sample complexity of realizably replicably learning (X,H) is at most:*

$$n(\rho,\alpha,\beta) \leq \tilde{O}\left(\frac{d^{12}\log^3(1/\beta)}{\alpha^2\rho^2}\right)$$

We give the proof in Section 5.8.

[17]Recall the problem can be solved replicably with $\text{poly}(|\mathscr{D}|)$ dependence by [ILPS22] even in the infinite setting.

Replicable Agnostic-to-Realizable Reduction

We now show how to combine List Heavy-Hitters with [HKLM22]'s agnostic-to-realizable technique to generalize their reduction to replicable learning.

Theorem 5.6.9. *Let (X,H) be a replicably learnable class with sample complexity $n(\rho,\alpha,\beta)$, and $n' = n(1/4,\alpha/4,\beta/4)$. Then (X,H) is agnostically learnable in*

$$m(\rho,\alpha,\beta) \leq O\left(\frac{\log^2(\Pi_H(n')\log\frac{1}{\rho\beta})\log\frac{1}{\rho}+\log\frac{1}{\rho\beta}}{\alpha^2\rho^2}\left(\alpha^2 n' + \log\frac{\Pi_H(n')}{\beta}\right)\log^3\frac{1}{\rho}\right)$$

samples, where $\Pi_H(n)$ is the growth function of (X,H).

Since $\Pi_H(n) \leq 2^n$, this means agnostic learning experiences only a small polynomial blow-up in sample complexity compared to the easier realizable setting. Furthermore, this bound holds even with distributional assumptions, since it does not rely on external quantities such as VC-dimension.

The proof of Theorem 5.6.9 is based on the following variant of a sub-routine from [HKLM22]'s agnostic-to-realizable reduction that generates a small list of good hypotheses.

Algorithm 35. List Distribution Generator

Result: Outputs a list of good hypotheses

Input: Replicable learner $\mathscr{L}$ on $n = n(\rho, \alpha, \beta)$ samples, family of $O(\log(1/\beta))$ random strings $\{r\}$

Parameters:

- Accuracy and confidence parameters $\alpha, \beta > 0$
- Labeled sample size $t = O\left(\frac{\log(\Pi_H(n(1/4,\alpha/4,\beta/4))+\log(1/\beta)}{\alpha^2}\right)$

Algorithm:

1. Sample n (unlabeled) samples $S_U \sim D_X^n$, and t labeled samples $S_L \sim D^t$
2. Run $\mathscr{L}$ across all strings in r on all possible labelings of S_U to receive:
$$C_r(S_U) := \{\mathscr{L}((S_U, h(S_U)); r_i) : h \in H, r_i \in \{r\}\}$$
3. Prune sub-optimal hypotheses from $C_r(S_U)$:
$$C_r^\alpha(S_U, S_L) := \left\{h \in C_r(S_U) : R_{S_L}(h) \le \min_{h' \in C_r(S_U)} (R_{S_L}(h')) + \alpha/2\right\}$$

return $C_r^\alpha(S_U, S_L)$

Algorithm 35 generates a sample from a distribution over families of hypotheses with near-optimal error. The accuracy of $\mathscr{A}$ promises that $C_r^\alpha(S_U, S_L)$ will be non-empty, and its replicability guarantees the distribution will have heavy-hitters. This means we can apply list heavy hitters to find a good hypothesis replicably.

Proof of Theorem 5.6.9. It is enough to prove that for any distribution D over $X \times Y$, the distribution over lists defined by $C_r^\alpha(S_U, S_L)$ satisfies

1. Correctness:

$$\mathbf{Pr}[\forall h \in C_r^\alpha(S_U, S_L) : R(h) \leq OPT + \alpha] \geq 1 - \beta/2$$

2. Heaviness:

$$\exists h \in H : \mathbf{Pr}[h \in C_r^\alpha(S_U, S_L)] \geq 1/2$$

For β a small enough constant, any $\Omega(1)$-heavy-hitter has error at most $OPT + \alpha$, so the result then follows immediately from applying list heavy hitters.

The first of these facts, correctness, is essentially trivial and just follows from observing that the size of $C_r(S_U)$ (the pre-pruned set) is at most $O(\log(1/\beta)\Pi_H(n))$ by construction. Since we use empirical estimates over $t = O(\frac{\log(\Pi_H(n)/\beta)}{\alpha^2})$ samples, standard Chernoff and union bounds imply that the empirical error of every element is estimated within $\alpha/4$ of its true value which implies the desired correctness guarantee.

It is left to show that $C_r^\alpha(S_U, S_L)$ has a heavy hitter. Fix some $h_{OPT} \in H$ achieving error OPT. Any $\frac{1}{4}$-replicable learner has the property that over at least half its random strings r, there exists some $h_r \in H$ such that:

$$\mathbf{Pr}_{S_U}[\mathscr{L}((S_U, h_{OPT}(S_U)), r) = h_r] \geq 1/2.$$

Furthermore, since $\mathscr{L}$ is additionally a PAC-learner, it must also be the case that h_r is within $\alpha/4$ of h_{OPT} over a $1 - \beta/4$ fraction of these "good" strings. Since our family consists of $O(\log(1/\beta))$ random strings and $\mathscr{L}((S_U, h_{OPT}(S_U)), r) \in C_r(S_U)$ by construction, this means the pre-pruned set $C_r(S_U)$ has a $1/2$-heavy-hitter with error at most $OPT + \alpha/4$ over a $1 - \beta/2$ fraction of families $\{r\}$. Finally, since our empirical estimates are good with high probability, h_r also appears in the pruned set $C_r^\alpha(S_U, S_L)$ with at least constant probability as desired. Finally, the sample complexity bound then follows from combining Theorem 5.6.7 with the observation that generating a sample from

$C_r^\alpha(S_U, S_L)$ requires $O\left(\frac{\alpha^2 n' + \log(\Pi_H(n')) + \log(1/\beta)}{\rho^2\alpha^2}\right)$ samples and $|C_r^\alpha(S_U, S_L)| \leq O(\log(1/\beta)\Pi_H(n'))$ by construction. $\square$

Private Agnostic-to-Realizable Reduction

We are finally ready to prove our agnostic-to-realizable reduction for private learning. We restate the Theorem for ease of reading.

Theorem 5.6.10 (Theorem 5.6.5 Restated)**.** *Let $(H, \mathscr{D})$ be a hypothesis class that is $(1, \frac{1}{\text{poly}(n)})$-privately $(\alpha, 0.01)$-PAC learnable in $n = n(\alpha)$ samples in the realizable setting. Then $(H, \mathscr{D})$ is $(1, \frac{1}{\text{poly}(m)})$-privately (α, β)-**Agnostically** learnable in*

$$m(\alpha, \beta) \leq O\left(\frac{n^2\alpha^2 + \log^3(\Pi_H(cn^2))}{\alpha^2}\log\beta^{-1}\log\left(\frac{n\log\beta^{-1}}{\alpha}\right)\right)$$

samples for some universal constant $c > 0$.

Proof. Recall we are given a realizable $(1, \text{poly}(n^{-1}))$-DP, $(\alpha, 0.01)$-accurate PAC learner $\mathscr{A}$ on n samples. We will convert $\mathscr{A}$ into an agnostic learner via the following 5 step process:

1. Amplify privacy to $(m^{-1/2}, \text{poly}(m^{-1}))$ by "secrecy of the sample" for $m \approx n^2$
2. Convert $\mathscr{A}$ to a 0.01-replicable realizable learner $\mathscr{R}$
3. Convert $\mathscr{R}$ into an agnostic learner $\mathscr{R}_{agn}$
4. Convert $\mathscr{R}_{agn}$ back into an agnostic private learner $\mathscr{A}_{agn}$
5. Privately amplify correctness of $\mathscr{A}_{agn}$

Let's formalize this procedure. In the first step, we convert $\mathscr{A}$ into a $(cm^{-1/2}, \text{poly}(m^{-1}))$-DP, $(\alpha, .01)$-accurate PAC learner on m samples for some small enough constant $c > 0$. This can be done by the so-called "secrecy of the sample" method (Lemma 5.2.9): draw $m = O(n^2)$ examples, construct a subset S by selecting m elements uniformly at random, and return $\mathscr{A}(S)$. For an appropriate choice of constants this is $(cm^{-1/2}, \text{poly}(m^{-1}))$-DP. Accuracy is maintained since S is equidistributed with a standard size m sample.

Now that we have our $(cm^{-1/2}, \text{poly}(m^{-1}))$-DP, $(\alpha, .01)$-accurate PAC learner, we invoke Corollary 5.3.18 (applying correlated sampling) to build the .01-replicable learner $\mathscr{R}$ on $O(n^2)$ samples that maintains $(\alpha, .01)$-correctness. Applying our agnostic-to-replicable reduction for replicable learning, this gives an .01-replicable $(\alpha, .01)$-correct agnostic learner on

$$m' \leq O\left(n^2 + \frac{\log^3(\Pi_H(c'n^2))}{\alpha^2}\right)$$

samples for some constant $c' > 0$. Finally, we move back to the private regime via Theorem 5.3.1, which gives an (ε, δ)-DP, $(\alpha, .1)$-correct agnostic learner on $O(\frac{m' \log \delta^{-1}}{\varepsilon})$ samples.

It is left to amplify the correctness probability β. This can be done by running the above algorithm independently $\log(1/\beta)$ times, and privately outputting the best hypothesis on the output set via the exponential mechanism, which one can check results in a $(2\varepsilon,\delta)$-DP $(2\alpha, \beta)$-accurate learner (see e.g. [SBG21, Theorem A.1]).

We have now seen how to build a (ε,δ)-DP (α,β)-accurate learner on

$$m'' \leq O\left(\frac{m' \log \delta^{-1} \log \beta^{-1}}{\varepsilon}\right)$$

samples. To give the form of the result in the theorem statement, it is enough to choose sample size t satisfying the recurrence $t \geq \Omega(m' \log t \log \beta^{-1})$. Selecting $t = c_2 m' \log \beta^{-1} \log(m'\beta^{-1})$ for large enough $c_2 > 0$ then completes the proof. □

5.6.4 Replicable Algorithms from Reduction

In this section, we show how we can use our reduction from replicability to differential privacy to obtain new replicable algorithms. Our reduction preserves accuracy, because on any fixed dataset, the output distribution of the replicable algorithm is identical to that of the differentially private algorithm (since our reduction simply applies correlated sampling to the output distribution of the differentially private algorithm on the input dataset—see Sections 5.3.2 and 5.3.3).

PAC Learning

We note that what we term "replicable" PAC learning corresponds to settings where the algorithm A is a PAC learner, and additionally is replicable for all input distributions D.

We show that our reduction gives the best known sample complexity bounds for replicable realizable and agnostic PAC learning for many hypothesis classes. Prior work also had to prove sample complexity bounds separately for all of these frameworks, whereas we are able to translate bounds proved for differential privacy directly through our reduction.

Thresholds/Approximate Median:

Fix any integer $d \geq 0$. We apply our framework to the hypothesis class $Thresh_d$ consisting of thresholds over the domain $\{0, 1, \ldots, d\}$. A threshold function f_z parameterized by integer $0 \leq z \leq d$, is defined as follows.

$$f_z(x) = \begin{cases} 1 & \text{if } x > z \\ 0 & \text{if } x \leq z \end{cases} \tag{5.17}$$

Impagliazzo et al. [ILPS22] asked whether a PAC learner could be obtained for this class with sample complexity polynomial in $\log^* d$. Our reduction answers this question by using a result of [KLM$^+$20] on learning thresholds privately. [18]

We first introduce the interior point problem.

Definition 5.6.11. *An algorithm solves the interior point problem over a totally ordered domain*

[18]We note that a similar approach (taking a differentially private algorithm for learning distributions under Kolmogorov distance (guaranteed by a reduction in [BNSV15] to the interior point problem), and applying our conversion from approx DP to replicability) also gives a replicable algorithm for releasing approximate median of a distribution with accuracy α and sample complexity $O_\alpha(\text{poly}\log^*(d))$. This closes an exponential gap in [ILPS22].

$\mathscr{X}$ with error probability β, if for all datasets $S \in \mathscr{X}^m$, if

$$\mathbf{Pr}[\min_i S_i \leq \mathscr{A}(S) \leq \max_i S_i] \geq 1-\beta.$$

Now we are ready to apply our reduction to obtain the improved sample complexity.

Theorem 5.6.12. *For all sufficiently small $\rho, \alpha, \beta \in (0,1)$, there exists a ρ-replicable, (α,β)-accurate realizable PAC learner for the hypothesis class $Thresh_d$ with sample complexity*

$$m = \tilde{O}\left(\frac{(\log^* d)^3 \log^2(1/\beta) \log^4(1/\rho)}{\alpha^2\rho^2}\right)$$

Proof. Let ε and δ be set as specified in Corollary 5.3.18. The work of [KLM$^+$20] (Theorem 4.1 in their paper) gives an $(\varepsilon/2, \delta/2)$ algorithm for solving the interior point problem with sample complexity $O(\frac{1}{\varepsilon}(\log^* d \log(1/\delta))^{1.5})$ and error probability at most $1/10$. By a result of Bun et al. [BNSV15, Theorem 5.6, Part 1], this gives an (ε,δ)-DP, $(\alpha, 2/10)$-proper PAC learner for $Thresh_d$ with sample complexity $O(\frac{1}{\varepsilon\alpha}(\log^* d \log(1/\delta))^{1.5})$. Now, by work of [BCS20] (See [SBG21, Theorem A.1] for a formal statement we use directly) this can be boosted to give an (ε,δ)-DP, (α,β)-accurate proper PAC learner for $Thresh_d$ with sample complexity $O(\frac{1}{\varepsilon\alpha}(\log^* d \log(1/\delta))^{1.5} \log(1/\beta))$.

First, we note that correlated sampling does not affect the accuracy guarantees since it maintains the distribution of the differentially private algorithm. Now, applying Corollary 5.3.18, and substituting in the values of ε and δ we get that there is a ρ-replicable (α,β)-accurate PAC learner for $Thresh_d$, whose sample complexity is the solution to the equation

$$m = C\frac{\sqrt{m\log(1/\rho)}}{\rho\alpha}(\log^* d \log(m/\rho))^{1.5} \log(1/\beta)$$

This gives us that

$$m = \tilde{O}\left(\frac{\log^4(1/\rho)}{\rho^2\alpha^2}(\log^* d)^3\log^2(1/\beta)\right).$$

$\square$

Finite Binary Hypothesis Classes:

By applying a result of [KLN$^+$11] on privately (agnostically) learning finite classes H, we get a learner with sample complexity that's polynomial in $\log|H|$.

Theorem 5.6.13. *For all sufficiently small $\rho, \alpha, \beta \in (0,1)$, and for all finite hypothesis classes H, there exists a ρ-replicable, (α, β)-accurate agnostic PAC learner for H with sample complexity*

$$m = O\left(\frac{(\log|H| + \log(1/\beta))^2\log(1/\rho)}{\alpha^2\rho^2}\right)$$

Proof. Let ε be set as specified in Corollary 5.3.18. The work of [KLN$^+$11] gives an $(\varepsilon, 0)$-DP agnostic learner for finite classes with sample complexity $m = O\left((\log|H| + \log(1/\beta))\left(\frac{1}{\alpha\varepsilon} + \frac{1}{\alpha^2}\right)\right)$.

We note that correlated sampling does not affect the accuracy guarantees since it maintains the distribution of the differentially private algorithm. Hence,substituting the value of ε and applying Corollary 5.3.18, we get a ρ-replicable (α, β)-accurate agnostic PAC learner for finite class H, whose sample complexity is the solution to the equation

$$m = O\left((\log|H| + \log(1/\beta))\left(\frac{\sqrt{m\log(1/\rho)}}{\alpha\rho} + \frac{1}{\alpha^2}\right)\right),$$

which gives us a quadratic in $\sqrt{m}$. Solving, we get that

$$m = O\left(\frac{(\log|H| + \log(1/\beta))^2\log(1/\rho)}{\alpha^2\rho^2}\right)$$

$\square$

The dependence on the accuracy parameter obtained via directly using our transformation here is suboptimal for realizable learners, and we cannot hope to improve it using our reduction alone, since the private finite class learner described above is optimal for realizable learners as well. See the finite class learner presented in Section 5.5.3 (that achieves the right inverse linear dependence on α) for more discussion. There also isn't a known replicable boosting algorithm with an inverse linear dependence on the accuracy parameter α. It is an interesting open question to investigate whether such a boosting algorithm exists.

We also observe that via our reduction, we obtain ρ-replicable (α, β)-PAC learners with sample complexity scaling as $O_{\rho,\alpha,\beta}(\texttt{PRDim}(H)^2)$ for finite hypothesis classes with finite probabilistic representation dimension (denoted by `PRDim`), and ρ-replicable (α, β)-PAC learners with sample complexity scaling as $\tilde{O}_{\rho,\alpha,\beta}(\texttt{LDim}(H)^{12})$ for finite hypothesis classes with finite Littlestone dimension (denoted by `LDim`) by instantiating our general transformation with private learners due to [BNS13] and [GGKM21] respectively. This improves on the sample complexities obtained in the work by Ghazi, Kumar and Manurangsi [GKM21]. We also give a direct version of the argument in Theorem 5.8.1.

Distribution Estimation Problems

As another illustration of the generality of our reduction, we instantiate it to give the first replicable algorithms for some distribution estimation problems.

Discrete distribution estimation

Consider the set P_k of all distributions over the domain $[k] = \{1, 2, 3, \ldots, k\}$ (where k is a natural number). The problem of discrete distribution estimation involves getting samples from any unknown fixed distribution D from P_k, and having to output a distribution D' that is close in some measure of distance to D (we call the closeness the "accuracy" of the algorithm).

We now describe the problem more formally. An algorithm is said to solve the discrete distribution estimation problem with accuracy α and $m(k)$ samples, if for all $k > 1$, and for all fixed distributions D over $[k]$, there exists an algorithm taking $m(k)$ independently drawn examples from D and outputting a distribution D' such that in expectation over the coins of the algorithm and the randomness of the sample, $d_{TV}(D,D') \le \alpha$.

It is known that the sample complexity of solving this problem with accuracy α (with no stability constraints) is $\Theta(k/\alpha^2)$.

If we add privacy constraints to the picture, it is known that there is an ε-DP algorithm for discrete distribution estimation that requires $O(k/\alpha^2 + k/\alpha\varepsilon)$ examples (see e.g., [ASZ21]). They show that this is tight even for (ε,δ)-DP algorithms, (when $\delta \le \varepsilon$, which is most often the regime of interest).

We can instantiate our reduction with this algorithm to get the first replicable algorithm for discrete distribution estimation.

Theorem 5.6.14. *Fix any $k > 2$. For all sufficiently small $\rho, \alpha \in (0,1)$, there exists an α-accurate, ρ-replicable algorithm that solves the discrete distribution estimation problem with α-accuracy, and m examples, where*

$$m = O\left(\frac{k^2 \log 1/\rho}{\alpha^2\rho^2}\right).$$

Proof. Let ε be set as specified in Corollary 5.3.18 (which gives the parameters for our conversion from differential privacy to replicability). The work of [ASZ21] and [DHS15] give an $(\varepsilon,0)$-DP algorithm $\mathscr{A}$ for discrete distribution estimation that takes in $m = O(k/\alpha^2 + k/\alpha\varepsilon)$ samples.

Now, we post-process this algorithm to get a finite output space to apply our reduction to. For every $i \in [k]$, round every coordinate of the output distribution d to the closest multiple of $\frac{\alpha}{k}$, to get a vector v. Now, apply the procedure given in Corollary 5.3.18 to convert this to a replicable algorithm. Call the output of the replicable algorithm $\hat{v}$. Finally do an ℓ_1 projection from $\hat{v}$ back to the k-simplex to get a new distribution $\hat{d}$.

We now argue that the above transformation preserves accuracy. Let the original distribution be p. Then, by the triangle inequality, we can write that

$$\mathbb{E}[\|\hat{d} - p\|_1] \leq \mathbb{E}[\|\hat{d} - \hat{v}\|_1] + \mathbb{E}[\|\hat{v} - p\|_1].$$

Now, since $\hat{v}$ is identically distributed to v (since the transformation to replicability simply involves correlated sampling), we have that for every such vector, there is a distribution in the k-simplex that is within α of it in ℓ_1 distance (because every output vector v prior to applying correlated sampling was obtained by rounding each coordinate of a distribution in the k-simplex to the closest multiple of $\frac{\alpha}{k}$). Hence, since $\hat{d}$ is the ℓ_1 projection of $\hat{v}$ onto the k-simplex, we have that $\mathbb{E}[\|\hat{d} - \hat{v}\|_1] \leq \alpha$. Hence, we can write that

$$\mathbb{E}[\|\hat{d} - p\|_1] \leq \alpha + \mathbb{E}[\|v - p\|_1],$$

where in the second term on the right hand side, we have used again that $\hat{v}$ and v are identically distributed.

Finally, since $\mathbb{E}[\|v - p\|_1] \leq \mathbb{E}[\|v - d\|_1] + \mathbb{E}[\|d - p\|_1]$ (by triangle inequality), and each of these terms is smaller than α (since v is obtained by discretizing d to a grid of length $\frac{\alpha}{k}$, and the second term can be bounded by the accuracy of algorithm $\mathscr{A}$), we get that

$$\mathbb{E}[\|\hat{d} - p\|_1] \leq \alpha,$$

as required.

Replicability of this transformation follows from the fact that the transformation prior to projection onto the simplex is replicable (by Corollary 5.3.18), and the fact that replicability is preserced under post-processing.

Hence, substituting the value of ε into the number of samples needed for the algorithm $\mathscr{A}$ to be α-accurate, we get a ρ-replicable α-accurate algorithm for discrete distribution estimation, whose sample complexity is the solution to the equation

$$m = O\left(\frac{k}{\alpha^2} + \frac{k\sqrt{m \log 1/\rho}}{\alpha\rho}\right),$$

which gives us a quadratic in $\sqrt{m}$. Solving, we get that

$$m = O\left(\frac{k^2 \log 1/\rho}{\alpha^2\rho^2}\right),$$

completing the proof. □

It is unclear from our results whether there is an algorithm for replicable discrete distribution estimation over $[k]$ that can achieve sample complexity linear in k; we leave this as an open problem.

Gaussian mean estimation

In this section, we give a replicable algorithm for high-dimensional Gaussian mean estimation (in the unknown covariance case).

In Gaussian mean estimation in d dimensions, algorithms are given examples drawn independently from a Gaussian distribution $N(\mu, \Sigma^2)$, where $\mu \in \mathbb{R}^d$ is the unknown mean, and Σ is an unknown $d \times d$ positive definite matrix. The goal is to estimate μ. The metric we will use to evaluate the quality of an estimate is the "Mahalanobis distance", which measures the error scaled according to the covariance matrix of the Gaussian distribution.

That is, with probability at least $1-\beta$ over the examples and the internal randomness of the algorithm, we want the algorithm given sample access to $N(\mu, \Sigma^2)$ to output a value $\hat{\mu}$ such that

$$\|\hat{\mu} - \mu\|_\Sigma = \|\Sigma^{-1/2}(\hat{\mu} - \mu)\|_2 \leq \alpha.$$

Without stability constraints, it is known that this problem can be solved using $m = \Theta(d/\alpha^2)$ examples.

Under the constraints of approximate differential privacy, the picture is more complicated. For a long time, the best dependence on d that was known, was $d^{3/2}$, with the bottleneck being private covariance estimation. However, in recent work, Brown, Gaboardi, Smith, Ullman, and Zakynthinou [BGS$^+$21] gave a sophisticated differentially private algorithm that achieved a linear dependence on d, by avoiding covariance estimation entirely. It is not clear how to make similar techniques work to obtain replicable algorithms via a direct analysis. Our reduction allows us to lift the analysis from [BGS$^+$21] to give a replicable algorithm for this task.

Since correlated sampling is known to only work on finite output spaces, we need to assume that the mean falls in a bounded ℓ_∞ ball (though our accuracy will not depend on the bounds of this ball). Additionally, we will need to discretize the output of the differentially private algorithm. However, discretization in this case is non-trivial, as the measure of accuracy is with respect to the unknown covariance matrix, and hence, we will first have to replicably estimate the minimum eigenvalue of the covariance matrix in order to decide the right level of discretization. For this purpose, we once again use our reduction and apply it to a differentially private algorithm for this task, also [BGS$^+$21]. We will assume that the covariance matrix's minimum eigenvalues are between non-negative numbers k and ℓ (known to the algorithm),[19] to guarantee finiteness of the output space for this algorithm. Again, our sample complexity is independent of these parameters.

Theorem 5.6.15. *Fix $R > 0$, $0 < k < \ell$, and sufficiently small $\rho > 0$. Fix a distribution $D = N(\mu, \Sigma)$, where $\|\mu\|_\infty \leq R$, and the minimum eigenvalue of Σ is between k and ℓ. Then, there is ρ-replicable algorithm that outputs an α-accurate estimate of the mean μ (in Mahalanobis distance) with probability at least $1 - \beta$, when given m independently drawn samples from D, where*

$$m = \tilde{O}\left(\frac{(d \log(1/\alpha) + \log(1/\beta))^2 \log^3(1/\rho)}{\alpha^2 \rho^2}\right).$$

[19]Note that this assumption can be relaxed by directly estimating the minimum eigenvalue using replicable heavy hitters instead of reducing to the DP algorithm

Proof. First, consider Lemma C.2 in [BGS$^+$21]. This gives a general way to privately estimate the minimum eigenvalue of Σ. We first discretize and truncate the output space of this algorithm as follows (and round outputs to their closest point in the corresponding grid). The discretization length will be $k/8$, and the upper bound will be 4ℓ. Since the algorithm in their paper guarantees a 4-approximation of the minimum eigenvalue with probability at least $1-\beta$, and takes $O(\frac{d\log(1/\beta\delta)}{\varepsilon})$ examples, by the assumed bounds on the covariance matrix, the discretized version guarantees an 8-approximation.

Now, since the output space of the eigenvalue estimation algorithm has been made finite, we apply the transformation in Corollary 5.3.18 with ε, δ set accordingly (with ρ being $\rho/2$). This gives a $\rho/2$-replicable algorithm that gives an 8-approximation to the minimum eigenvalue of the covariance matrix with sample complexity that is the solution to the equation $m_1 = O\left(\frac{d\sqrt{m_1\log(1/\rho)}\log(m_1/\beta\rho)}{\rho}\right)$, which gives us that

$$m_1 = \tilde{O}\left(\frac{d^2\log^3(1/\rho)\log^2(1/\beta)}{\rho^2}\right).$$

Let the output of this algorithm $\mathscr{A}_1$ be $\hat{\lambda}$.

Now, we are ready to use the mean estimator described in Theorem 2 from [BGS$^+$21] (on a fresh set of samples). We will assume that hardcoded into this algorithm is an eigenvalue $\hat{\lambda}$, which is an 8-approximation to the minimum eigenvalue λ_d of the covariance matrix. Consider a postprocessing of the output $\hat{\mu}$ of the algorithm described in that theorem such that the value of each coordinate is truncated to have ℓ_∞ norm at most R, and has been projected to an α'-grid, where $\alpha' = \min(\hat{\lambda}^{1/2}, 1)\frac{\alpha}{\sqrt{d}}$. Let the post-processed mean be μ_{disc}. Then, by the guarantee of Theorem 2 in their paper, we have that with probability at least $1-3\beta$,

$$\begin{aligned}
\|\mu_{disc}-\mu\|_\Sigma &\le \|\mu_{disc}-\hat{\mu}\|_\Sigma + \|\hat{\mu}-\mu\|_\Sigma \\
&\le \|\Sigma^{-1/2}[\mu_{disc}-\hat{\mu}]\|_2+\alpha \\
&\le \frac{1}{\sqrt{\lambda_d}}\|\mu_{disc}-\hat{\mu}]\|_2+\alpha \\
&\le \frac{\alpha'}{\sqrt{\lambda_d}}\sqrt{d}+\alpha \\
&\le \frac{\alpha\sqrt{\lambda}}{\sqrt{d}\sqrt{\lambda_d}}\sqrt{d}+\alpha \\
&\le \frac{\alpha\sqrt{8\lambda_d}}{\sqrt{d}\sqrt{\lambda_d}}\sqrt{d}+\alpha = O(\alpha)
\end{aligned}$$

Note that the sample complexity of this (ε,δ)-DP algorithm (where privacy is wrt the fresh sample) is

$$m_2 = O\left(\frac{d+\log(1/\beta)}{\alpha^2}+\frac{\log(1/\delta)}{\varepsilon}+\frac{d\log(1/\alpha)+\log(1/\beta)}{\alpha\varepsilon}\right).$$

Now, we are ready to apply our transformation (recall that it preserves accuracy, since the distribution is unchanged by correlated sampling). Setting ε,δ as in Corollary 5.3.18 (with ρ set to $\rho/2$), and applying our transformation we can convert this to a $\rho/2$-replicable algorithm $\mathscr{A}_2$, with sample complexity that is the solution to the equation

$$m_2 = O\left(\frac{d+\log(1/\beta)}{\alpha^2}+\log\frac{m_2}{\rho}\frac{\sqrt{m_2\log(1/\rho)}}{\rho}+\frac{(d\log(1/\alpha)+\log(1/\beta))\sqrt{m_2\log(1/\rho)}}{\alpha\rho}\right).$$

Solving this equation gives that

$$m_2 = \tilde{O}\left(\frac{(d\log(1/\alpha)+\log(1/\beta))^2\log^3(1/\rho)}{\alpha^2\rho^2}\right).$$

Now, note first that by adaptive composition, running $\mathscr{A}_1$ and then using its estimate in the algorithm

$\mathscr{A}_2$ together gives a ρ-replicable algorithm (since each is individually $\rho/2$-replicable). Additionally, the composed algorithm is α-accurate with probability $1-4\beta$ (taking a union bound of the failure probabilities of $\mathscr{A}_1$ and $\mathscr{A}_2$). Note that m_2 asymptotically dominates m_1, so the sample complexity of this entire procedure is

$$m = \tilde{O}\left(\frac{(d\log(1/\alpha)+\log(1/\beta))^2\log^3(1/\rho)}{\alpha^2\rho^2}\right).$$

□

Gaussian Identity Testing

As a final example of the generality of our reduction, we consider the problem of identity testing of multivariate Gaussian distributions. In this problem, we are given samples from either a fixed Gaussian distribution D with known mean and covariance, or from a Gaussian distribution that is α-far in Mahalanobis distance from D. The goal is to correctly guess which case we're in with probability at least $2/3$ (we will say the algorithm successfully distinguishes the two cases if this is satisfied). Without stability constraints, this problem can be solved with $O(\frac{\sqrt{d}}{\alpha^2})$ samples.

This problem was studied subject to privacy constraints in [CKM$^+$20], and their results were then improved in [Nar22]. We apply results of the latter to get a replicable algorithm for Gaussian identity testing.

Theorem 5.6.16. *Fix $d \in \mathbb{N}^+$ and sufficiently small $\rho, \alpha > 0$. Fix known $\mu \in \mathbb{R}^d$ and known covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. Then, there is a ρ-replicable algorithm $\mathscr{A}$ that can succesfully distinguish between Case H_0, where $\mathscr{A}$ receives m samples from $N(\mu,\Sigma)$ and Case H_1 where $\mathscr{A}$ receives m samples from any distribution $N(\mu',\Sigma)$, such that $\|\mu'-\mu\|_\Sigma \geq \alpha$, as long as*

$$m = \tilde{O}\left(\frac{d^{1/2}}{\alpha^2\rho^2}\right).$$

Proof. Note that Theorem 1.7 of [Nar22] gives an $(\varepsilon, 0)$-algorithm for this task that achieves sample complexity $m = \tilde{O}\left(\frac{d^{1/2}}{\alpha^2} + \frac{d^{1/4}}{\alpha\varepsilon}\right)$. We directly instantiate this algorithm with our reduction in order to get the result (since the output space is finite (just a single bit), we can do this without modifying the algorithm).

Set ε as in Corollary 5.3.18. Note that since our reduction maintains the same distribution as the differentially private algorithm, the accuracy guarantees are the same. Hence, there is a ρ-replicable algorithm for this task with sample complexity that is the solution to the equation

$$m = \tilde{O}\left(\frac{d^{1/2}}{\alpha^2} + \frac{d^{1/4}\sqrt{m \log 1/\rho}}{\alpha\rho}\right).$$

Then, solving this equation, we get that

$$m = \tilde{O}\left(\frac{d^{1/2}}{\alpha^2\rho^2}\right).$$

□

5.7 Appendix: Estimating OPT

In this section, we give an algorithm for replicably estimating the minimum error hypothesis in a class (X,H) over an arbitrary joint distribution D over $X \times \{0,1\}$.

Algorithm 36. Replicably estimate OPT

Result: Outputs $v \in [OPT, OPT + \alpha/2]$

Input: Finite Class H, Joint Distribution D over $X \times \{0,1\}$ (Sample Access)

Parameters:

- Replicability, Accuracy, Confidence $\rho, \alpha, \beta > 0$
- Sample Complexity $m = m(\rho,\alpha,\beta) \leq O\left(\frac{\log(|H|/\beta\rho)}{\alpha^2\rho^2}\right)$

Algorithm:

```
1. Draw a labeled sample $S \sim D^m$ and compute $err_S(f)$ for every $f \in H$.
2. $a \leftarrow_r [0, \alpha/16]$
3. $B_i = [i\alpha + a, (i+1)\alpha + a)$
return $\frac{j}{8}\alpha + a$, where $OPT_S + \alpha/4 \in B_j$
```

Lemma 5.7.1. *Let $\mathscr{D}$ be a joint distribution over $X \times \{0,1\}$ and H a concept class over X. Then for any $\alpha, \beta, \rho > 0$, Algorithm 36 is a ρ-replicable algorithm over $O\left(\frac{\log(\frac{|H|}{\rho\beta})}{\rho^2\alpha^2}\right)$ samples that outputs a good estimate of OPT with high probability:*

$$\mathbf{Pr}_{r,S}\big[\mathscr{A}(S) \in [OPT, OPT + \alpha/2]\big] \geq 1 - \beta.$$

Proof. The proof uses an argument similar to the randomized rounding trick introduced in [ILPS22] for replicable statistical queries. Assume for simplicity that $\frac{1}{8\alpha}$ is integer (the argument is essentially no different otherwise), and break the interval $[0,1]$ into $\frac{\alpha}{8}$-sized buckets:

$$B_1 = \left[0, \frac{\alpha}{8}\right), \ \dots, B_{\frac{1}{\alpha}} = \left[1 - \frac{\alpha}{8}, 1\right].$$

Consider the rounding scheme ROUND that maps OPT_S to the upper limit of its corresponding bucket. Notice that as long as OPT is not within $\frac{\rho\alpha}{64}$ of the threshold value between two buckets, uniform convergence promises that OPT_{S_1} and OPT_{S_2} will lie in the same bucket with probability at least $1 - \frac{\rho}{2}$. As such the problem only occurs at the boundaries, which can be fixed by randomly shifting the thresholds between each bucket by $a \in [0, \frac{\alpha}{16}]$. Then for any fixed value of OPT, the probability it lies within $\frac{\rho\alpha}{64}$ of a shifted boundary is at most $\frac{\rho}{2}$, which combined with the previous observation proves the algorithm ρ-replicable.

Towards correctness, observe that uniform convergence of finite classes promises that the empirical optimum OPT_S is within $\frac{\alpha}{16}$ of the true optimum with probability at least $1 - \beta$. Furthermore, rounding shifts any value by at most $\frac{3\alpha}{16}$. Thus $\text{ROUND}(OPT_S + \alpha/4) \in [OPT, OPT + \frac{\alpha}{2}]$ with high probability as desired. □

5.8 Appendix: Learning Finite Littlestone Classes

One immediate application of list heavy-hitters is a sample-efficient replicable algorithm for classes with finite Littlestone dimension, as in [GGKM21], leading to a modest improvement in sample complexity over the best known bound of $\tilde{O}(d^{14})$.

Theorem 5.8.1. *Let (X,H) be a class with Littlestone dimension d. Then the sample complexity of realizably replicably learning (X,H) is at most:*

$$n(\rho, \alpha, \beta) \leq \tilde{O}\left(\frac{d^{12} \log^3(1/\beta)}{\alpha^2 \rho^2}\right)$$

Proof. In their work on user level privacy, Ghazi, Kumar, and Manurangsi [GKM21] build on the work of [GGKM21] to show the existence of an algorithm outputting lists of hypotheses satisfying the following guarantees:

1. *Optimality*: With probability at least $1-\beta/2$, all hypotheses output by $\mathscr{L}$ have risk at most $\alpha/2$
2. *Heavy Hitter*: There exists $h \in H$ output with probability $\Omega(1/d)$
3. *Size*: $\mathscr{L}$ outputs at most $\exp(d^2 + d\log\frac{d}{\alpha\beta})$ hypotheses
4. *Sample Complexity*: $\mathscr{L}$ uses at most $\tilde{O}(\frac{d^6\log^2\frac{1}{\beta}}{\alpha^2})$ samples.

Note that for $\beta \leq O(1/d)$, any heavy hitter of this distribution is a good hypothesis, so it is enough to replicably output such a heavy hitter. Applying Theorem 5.6.7, this can be done ρ-replicably and with probability at least $1-\beta$ using

$$O\left(\frac{\log^2\frac{|\mathscr{D}|\log\frac{1}{\rho\beta}}{\eta}\log\frac{1}{\rho}+\log\frac{1}{\rho\beta}}{\eta^2\rho^2}\log^3\frac{1}{\rho}\right)=\tilde{O}\left(\frac{d^6\log\frac{1}{\rho}+d^2\log\frac{1}{\rho\beta}}{\rho^2}\log^3\frac{1}{\rho}\right)$$

i.i.d outputs of the list algorithm. Each output itself costs $\tilde{O}(\frac{d^6\log^2\frac{1}{\beta}}{\alpha^2})$ samples to generate, leading to the stated sample complexity. $\square$

5.9 Appendix: Additional Properties of Replicability

5.9.1 Randomness Management

Often, we design replicable algorithms which use randomness for multiple purposes. How do we ensure that they use the same sections of random string r for the same subroutines? What if the number of bits used for each purpose varies between runs of the algorithm? The following arguments show that we can typically guarantee that the same sections of r are used for the same purposes across both runs.

Lemma 5.9.1. *Say an algorithm $\mathscr{A}$ makes at most k calls to its randomness oracle, using at most $b_1, \ldots, b_k$ bits of randomness for each call respectively. Then there is an algorithm $\mathscr{A}'$ that replicably uses at most $k \cdot \max_{i\in[k]}\{b_i\}$ bits of randomness.*

Here, by "replicably uses" we mean that algorithm $\mathscr{A}'$ uses the same positions in the random string for each subroutine in every run of the algorithm.

Proof. Have $\mathscr{A}'$ interpret its random string as follows: rather than using randomness sequentially for each of k purposes (non-replicable if the required number of bits changes), portion the random string into k pieces in a modular way. In other words, use the bits of r in positions $i \mod k$ solely for the i'th call to the randomness oracle by algorithm $\mathscr{A}$. At most $k \cdot \max_{i\in[k]}\{b_i\}$ bits of randomness are used. □

Note that the algorithm itself does not need to know how much randomness it will use a priori to use this method.

What if the algorithm does not have a fixed number of calls to the randomness oracle? As long as the randomness calls occur sequentially, one can assign consistent subsections of the random string to each possible call. To do so, we use the same snake-path trick (i.e., the Cantor pairing function) often used to equate the cardinality of the natural numbers and rational numbers.

Lemma 5.9.2. *Say an algorithm $\mathscr{A}$ makes at most k calls to its randomness oracle, using at most $b_1, \ldots, b_k$ bits of randomness for each call respectively. Then there is an algorithm $\mathscr{A}'$ that replicably uses at most $(k+\max_{i\in[k]}\{b_i\})^2/2+(k+\max_{i\in[k]}\{b_i\})/2$ bits of randomness.*

Proof. We allocate bits from our randomness oracle to different (unknown bit-length) calls using the Cantor pairing function. The maximum overhead in bit complexity of the randomness occurs when the k'th randomness call uses the most bits. In this case, roughly half of $(k+\max_{i\in[k]}\{b_i\})^2$ random bits must be drawn. □

Again, the algorithm itself does not need to know how much randomness it will use a priori to use this method. It also does not need to know how many different calls k to the randomness oracle will be performed, so long as these calls can be ordered sequentially in some canonical way. For example, if the algorithm operates in rounds with a finite number of (conditional) random calls in each round, then the algorithm can reserve specific sections of its random tape for each of the possible calls without requiring infinite randomness.

More generally, if a replicable algorithm uses the Cantor pairing function to allocate portions of its randomness, then the Cantor pairing function can be replaced by any deterministic pairing function $f : \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{Z}^+$ (and the ensuing algorithm will still be replicable). However, different or more situational pairing functions may give a specific replicable algorithm $\mathscr{A}$ improvements in complexity parameters such as amount of random bits used, time complexity, and space complexity. When designing replicable algorithms with very small parameters, one may have to be careful to ensure that the randomness management can also be done within these constraints.

5.9.2 Replicability across Two Close Distributions

Next, we prove a simple Lemma bounding the effect of distributional shift on replicability.

Lemma 5.9.3 (Replicability under Distributional Shift)**.** *Let D_1 and D_2 be two distributions over $\mathscr{X}$ with total variational distance $d_{TV}(D_1, D_2) = \delta$. Let $\rho \geq 0$, and let $\mathscr{A}$ be a ρ-replicable algorithm that uses a sample of size exactly m. Then*

$$\mathbf{Pr}_{S_1 \sim D_1^m, S_2 \sim D_2^m, r}[\mathscr{A}(S_1; r) = \mathscr{A}(S_2; r)] \geq (1-\delta)^{2m}\rho.$$

Proof. Since $d_{TV}(D_1, D_2) = \delta$, there exist distributions D, D', and D'' such that $D_1 = (1-\delta)D +$

$\delta D'$ and $D_2 = (1-\delta)D + \delta D''$. Thus,

$$\begin{aligned}
\mathbf{Pr}_{S_1 \sim D_1^m, S_2 \sim D_2^m, r}[\mathscr{A}(S_1; r) = \mathscr{A}(S_2; r)] &= (1-\delta)^{2m} \mathbf{Pr}_{S_1 \sim D^m, S_2 \sim D^m, r}[\mathscr{A}(S_1; r) = \mathscr{A}(S_2; r)] \\
&\quad + \dots \\
&\quad + \delta^{2m} \mathbf{Pr}_{S_1 \sim D'^m, S_2 \sim D''^m, r}[\mathscr{A}(S_1; r) = \mathscr{A}(S_2; r)] \\
&\geq (1-\delta)^{2m} \mathbf{Pr}_{S_1 \sim D^m, S_2 \sim D^m, r}[\mathscr{A}(S_1; r) = \mathscr{A}(S_2; r)] \\
&= (1-\delta)^{2m} \rho.
\end{aligned}$$

□

However, Lemma 5.9.3 may not be tight for specific class of algorithms, functions, or distributions.

5.10 Glossary

ρ	replicability parameter
(η,ν)	2-parameter definition of replicability
(ε,δ)	differential privacy parameters
H	hypothesis classes
f,g,h	target functions and functions from hypothesis classes
$\mathscr{X}$	input spaces
$\mathscr{Y}$	output spaces
(α,β)	accuracy parameters (failure probability is β).
m	sample complexity, size of datasets
D	distributions (subscripts for multiple distributions)
P	family of distributions
err	learning error (subscripts for sample and distribution)
S	input datasets
x,y	single data point and single output respectively
r	internal coins
d	(notions of) dimension
Δ	symmetric difference

d_{TV}	total variation distance
$\approx_{\varepsilon,\delta}$	approximate max-KL indistinguishable
p	Bernoulli biases
b	bit used for message
c	ciphertext
i, j	subscripts for (dual) indexing
O	set of outputs
$\mathscr{A}, \mathscr{B}$	algorithms
I	interval
v	threshold values
$\|\|$	concatenation of strings

Acknowledgements

We thank Adam Smith for helpful discussions on max information, Zhiwei Wu for pointing us to perfect generalization, Shay Moran for helpful discussions regarding the relation of our work to [KKMV23], and Christopher Ye for helpful comments on a prior version of this manuscript. The views expressed in this paper are those of the authors and not those of the U.S. Census Bureau or any other sponsor.

Chapter 5, in full, is a reprint of the material as it appeared on Arxiv at https://arxiv.org/abs/2303.12921v2. This material is the full version of the paper that appeared in the Proceedings of the 55th Annual ACM SIGACT Symposium on Theory of Computing 2023. Bun, Mark; Gaboardi, Marco; Hopkins, Max; Impagliazzo, Russell; Lei, Rex; Pitassi, Toniann; Sivakumar, Satchit;

Sorrell, Jessica. “Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization”. For this dissertation, minor formatting edits were made to improve readability. The dissertation author was the primary investigator and author of this paper.

Bibliography

[ABHU15] P. Awasthi, M. F. Balcan, N. Haghtalab, and R. Urner. Efficient learning of linear separators under bounded noise. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015*, pages 167–190, 2015.

[ABHZ16] P. Awasthi, M. F. Balcan, N. Haghtalab, and H. Zhang. Learning and 1-bit compressed sensing under asymmetric noise. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016*, pages 152–192, 2016.

[ABMS20] Noga Alon, Amos Beimel, Shay Moran, and Uri Stemmer. Closure properties for private classification and online prediction. In *Conference on Learning Theory*, pages 119–152. PMLR, 2020.

[AH18] Aws Albarghouthi and Justin Hsu. Synthesizing coupling proofs of differential privacy. *Proc. ACM Program. Lang.*, 2(POPL):58:1–58:30, 2018.

[AHHM21] Noga Alon, Steve Hanneke, Ron Holzman, and Shay Moran. A theory of PAC learnability of partial concept classes. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 658–671. IEEE, 2021.

[AL88] D. Angluin and P. Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, 1988.

[ALMM19] Noga Alon, Roi Livni, Maryanthe Malliaris, and Shay Moran. Private pac learning implies finite littlestone dimension. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 852–860, 2019.

[AS19] Omer Angel and Yinon Spinka. Pairwise optimal coupling of multiple random variables, 2019.

[Ass20] Association for Computing Machinery. Artifact review and badging - current. version 1.1. 2020.

[ASZ21] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. Differentially private assouad, fano, and le cam. In Vitaly Feldman, Katrina Ligett, and Sivan Sabato, editors, *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, volume 132 of *Proceedings of Machine Learning Research*, pages 48–78. PMLR, 2021.

[Bak16] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature News*, 533(7604):452, May 2016.

[BBG18] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 62806290, Red Hook, NY, USA, 2018. Curran Associates Inc.

[BCK$^+$21] Gilles Barthe, Rohit Chadha, Paul Krogmeier, A. Prasad Sistla, and Mahesh Viswanathan. Deciding accuracy of differential privacy schemes. *Proc. ACM Program. Lang.*, 5(POPL):1–30, 2021.

[BCS20] Mark Bun, Marco Leandro Carmosino, and Jessica Sorrell. Efficient, noise-tolerant, and private learning via boosting. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, Proceedings of Machine Learning Research. PMLR, 2020.

[BDLM01] S. Ben-David, P. Long, and Y. Mansour. Agnostic boosting. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 507–516, 2001.

[BDPSS09] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT*, volume 3, page 1, 2009.

[BDRS18] Mark Bun, Cynthia Dwork, Guy N. Rothblum, and Thomas Steinke. Composable and versatile privacy via truncated CDP. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 74–86. ACM, 2018.

[BE02] Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

[BE12] C. Glenn Begley and Lee M Ellis. Raise standards for preclinical cancer research. *Nature (London)*, 483(7391):531–533, 2012.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth.

Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[BF16] Raef Bassily and Yoav Freund. Typicality-based stability and privacy. *CoRR*, abs/1604.03336, 2016.

[BGA$^+$15] Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. Higher-order approximate relational refinement types for mechanism design and differential privacy. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 55–68. ACM, 2015.

[BGH$^+$16] Mohammad Bavarian, Badih Ghazi, Elad Haramaty, Pritish Kamath, Ronald L. Rivest, and Madhu Sudan. The optimality of correlated sampling. *CoRR*, abs/1612.01041, 2016.

[BGS$^+$21] Gavin Brown, Marco Gaboardi, Adam D. Smith, Jonathan R. Ullman, and Lydia Zakynthinou. Covariance-aware private mean estimation without private covariance estimation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 7950–7964, 2021.

[BH07] P. Bühlmann and T. Hothorn. Boosting algorithms: Regularization, prediction and model fitting. *Statist. Sci.*, 22(4):477–505, 11 2007.

[BHK09] Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 1193–1200. SIAM, 2009.

[BI91] Gyora M Benedek and Alon Itai. Learnability with respect to fixed distributions. *Theoretical Computer Science*, 86(2):377–389, 1991.

[BK09] E. Beigman and B. B. Klebanov. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 280–287, 2009.

[BLM20] Mark Bun, Roi Livni, and Shay Moran. An equivalence between private classification and online prediction. *2020 IEEE 61st Annual Symposium on Foundations of*

Computer Science (FOCS), pages 389–402, 2020.

[Blu03] A. Blum. Machine learning: My favorite results, directions, and open problems. In *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 11–14, 2003.

[BLW96] Peter L Bartlett, Philip M Long, and Robert C Williamson. Fat-shattering and the learnability of real-valued functions. *journal of computer and system sciences*, 52(3):434–452, 1996.

[BMA19] Raef Bassily, Shay Moran, and Noga Alon. Limits of private learning with access to public data. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 10342–10352, 2019.

[BMN^{+}18] Raef Bassily, Shay Moran, Ido Nachum, Jonathan Shafer, and Amir Yehudayoff. Learners that use little information. In Firdaus Janoos, Mehryar Mohri, and Karthik Sridharan, editors, *Algorithmic Learning Theory, ALT 2018, 7-9 April 2018, Lanzarote, Canary Islands, Spain*, volume 83 of *Proceedings of Machine Learning Research*, pages 25–55. PMLR, 2018.

[BMNS19] Amos Beimel, Shay Moran, Kobbi Nissim, and Uri Stemmer. Private center points and learning of halfspaces. In *Conference on Learning Theory*, pages 269–282. PMLR, 2019.

[BNS13] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of private learners. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 97–110. ACM, 2013.

[BNS^{+}16a] Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC 2016. Association for Computing Machinery, 2016.

[BNS16b] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. *Theory Comput.*, 12(1):1–61, 2016.

[BNS16c] Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In *Proceedings of the 2016 ACM Conference on Innovations in*

Theoretical Computer Science, pages 369–380, 2016.

[BNS$^+$21] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan R. Ullman. Algorithmic stability for adaptive data analysis. *SIAM J. Comput.*, 50(3), 2021.

[BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 634–649, 2015.

[Bro97] Andrei Z. Broder. On the resemblance and containment of documents. In Bruno Carpentieri, Alfredo De Santis, Ugo Vaccaro, and James A. Storer, editors, *Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings*, pages 21–29. IEEE, 1997.

[BS07] J. Bradley and R. Schapire. Filterboost: Regression and classification on large datasets. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

[BSU19] Mark Bun, Thomas Steinke, and Jonathan R. Ullman. Make up your mind: The price of online queries in differential privacy. *J. Priv. Confidentiality*, 9(1), 2019.

[BUV18] Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *SIAM J. Comput.*, 47(5):1888–1938, 2018.

[CCMY24] Zachary Chase, Bogdan Chornomaz, Shay Moran, and Amir Yehudayoff. Local borsuk-ulam, stability, and replicability. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 17691780, New York, NY, USA, 2024. Association for Computing Machinery.

[CKM$^+$20] Clément L. Canonne, Gautam Kamath, Audra McMillan, Jonathan R. Ullman, and Lydia Zakynthinou. Private identity testing for high-dimensional distributions. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[CKMY20] S. Chen, F. Koehler, A. Moitra, and M. Yau. Classification under misspecification: Halfspaces, generalized linear models, and connections to evolvability. *CoRR*, abs/2006.04787, 2020.

[CLN$^+$16] Rachel Cummings, Katrina Ligett, Kobbi Nissim, Aaron Roth, and Zhiwei Steven

Wu. Adaptive learning with robust generalization guarantees. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 772–814. JMLR.org, 2016.

[CMY23] Zachary Chase, Shay Moran, and Amir Yehudayoff. Replicability and stability in learning, 2023.

[Coh97] E. Cohen. Learning noisy perceptrons by a perceptron in polynomial time. In *Proceedings of the Thirty-Eighth Symposium on Foundations of Computer Science*, pages 514–521, 1997.

[Dan16] A. Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the 48th Annual Symposium on Theory of Computing, STOC 2016*, pages 105–117, 2016.

[DFH$^+$15a] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2350–2358, 2015.

[DFH$^+$15b] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 117–126. ACM, 2015.

[DGT19] I. Diakonikolas, T. Gouleakis, and C. Tzamos. Distribution-independent PAC learning of halfspaces with massart noise. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 4751–4762, 2019.

[DHS15] Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt. Differentially private learning of structured discrete distributions. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2566–2574, 2015.

[DK20] I. Diakonikolas and D. M. Kane. Hardness of learning halfspaces with massart noise. *CoRR*, abs/2012.09720, 2020.

[DKK$^+$20] I. Diakonikolas, D. Kane, V. Kontonis, C. Tzamos, and N. Zarifis. A polynomial time algorithm for learning halfspaces with tsybakov noise. *CoRR*, abs/2010.01705, 2020.

[DKTZ20a] I. Diakonikolas, V. Kontonis, C. Tzamos, and N. Zarifis. Learning halfspaces with massart noise under structured distributions. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020*, volume 125 of *Proceedings of Machine Learning Research*, pages 1486–1513. PMLR, 2020.

[DKTZ20b] I. Diakonikolas, V. Kontonis, C. Tzamos, and N. Zarifis. Learning halfspaces with tsybakov noise. *CoRR*, abs/2006.06467, 2020.

[DMNS16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016.

[DMY16] Ofir David, Shay Moran, and Amir Yehudayoff. Supervised learning through the lens of compression. *Advances in Neural Information Processing Systems*, 29:2784–2792, 2016.

[DPVWV23] Peter Dixon, A. Pavan, Jason Vander Woude, and N. V. Vinodchandran. List and certificate complexities in replicable learning, 2023.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 51–60. IEEE Computer Society, 2010.

[DSS$^+$15] Cynthia Dwork, Adam D. Smith, Thomas Steinke, Jonathan R. Ullman, and Salil P. Vadhan. Robust traceability from trace amounts. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 650–669. IEEE Computer Society, 2015.

[DW79a] L. Devroye and T. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979.

[DW79b] L. Devroye and T. J. Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Trans. Inform. Theory*, 25, 1979.

[EHKS23] Eric Eaton, Marcel Hussing, Michael Kearns, and Jessica Sorrell. Replicable reinforcement learning, 2023.

[EKK$^+$23] Hossein Esfandiari, Alkis Kalavasis, Amin Karbasi, Andreas Krause, Vahab Mirrokni, and Grigoris Velegkas. Replicable bandits. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[EKM$^+$23] Hossein Esfandiari, Amin Karbasi, Vahab Mirrokni, Grigoris Velegkas, and Felix Zhou. Replicable clustering, 2023.

[Fel10] V. Feldman. Distribution-specific agnostic boosting. In *Proceedings of Innovations in Computer Science*, pages 241–250, 2010.

[Fel16] V. Feldman. A general characterization of the statistical query complexity. *CoRR*, abs/1608.02198, 2016.

[FGRW09] V. Feldman, V. Guruswami, P. Raghavendra, and Y. Wu. Agnostic learning of monomials by halfspaces is hard. In *FOCS*, pages 385–394, 2009.

[FJ14] Matthew Fredrikson and Somesh Jha. Satisfiability modulo counting: a new approach for analyzing privacy properties. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 42:1–42:10. ACM, 2014.

[Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[FS97] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[FV13] B. Frénay and M. Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[FX14] Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. In Maria-Florina Balcan, Vitaly Feldman, and Csaba Szepesvári, editors, *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, volume 35 of *JMLR Workshop and Conference Proceedings*, pages 1000–1019. JMLR.org, 2014.

[Gav03] D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *JMLR*, 4:101–117, 2003.

[GG11] Erann Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, 18:136, 2011.

[GG17] Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic nc. In *ICALP*, 2017.

[GGH18] Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-Deterministic Proofs. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[GGKM21] Badih Ghazi, Noah Golowich, Ravi Kumar, and Pasin Manurangsi. Sample-efficient proper pac learning with approximate differential privacy. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 183–196, 2021.

[GGMW19] Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-deterministic streaming. *Electron. Colloquium Comput. Complex.*, 26:177, 2019.

[GGR13] Oded Goldreich, Shafi Goldwasser, and Dana Ron. On the possibilities and limitations of pseudodeterministic algorithms. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, page 127138, New York, NY, USA, 2013. Association for Computing Machinery.

[GHH$^+$13] Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 357–370. ACM, 2013.

[GKM21] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. User-Level Differentially Private Learning via Correlated Sampling . In *Proc. 35th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[GL19] Ofer Grossman and Yang P. Liu. Reproducibility and pseudo-determinism in log-space. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 606–620. SIAM, 2019.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 365377, New York, NY, USA, 1982. Association for Computing Machinery.

[GM18] Anna C. Gilbert and Audra McMillan. Property testing for differential privacy. *CoRR*, abs/1806.06427, 2018.

[GNP20] Marco Gaboardi, Kobbi Nissim, and David Purser. The complexity of verifying loop-free programs as differentially private. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 129:1–129:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Gol19] Oded Goldreich. Multi-pseudodeterministic algorithms. *Electron. Colloquium Comput. Complex.*, 26:12, 2019.

[Gol21] Noah Golowich. Differentially private nonparametric regression under a growth condition. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 2149–2192. PMLR, 2021.

[Gow10] W. T. Gowers. Decompositions, approximate structure, transference, and the hahn-banach theorem. *Bulletin of the London Mathematical Society*, 42(4), Jun 2010.

[GT08] B. Green and T. Tao. The primes contain arbitrarily long arithmetic progressions. *Annals of Mathematics*, 167(2):481–547, 2008.

[Hau92] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992.

[HIB$^+$17] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters, 2017. cite arxiv:1709.06560Comment: Accepted to the Thirthy-Second AAAI Conference On Artificial Intelligence (AAAI), 2018.

[HIK$^+$24] Max Hopkins, Russell Impagliazzo, Daniel Kane, Sihan Liu, and Christopher Ye. Replicability in high dimensional statistics, 2024.

[HKLM22] Max Hopkins, Daniel M. Kane, Shachar Lovett, and Gaurav Mahajan. Realizable learning is all you need. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, pages 3015–3069. PMLR, 2022.

[Hol05] T. Holenstein. Key agreement from weak bit agreement. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 664–673. ACM, 2005.

[Hol09] Thomas Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory Comput.*, 5(1):141–172, 2009.

[IHGP17] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *CoRR*, abs/1708.04133, 2017.

[ILPS22] Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell. Reproducibility in learning. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 818–831. ACM, 2022.

[ILS21] Russell Impagliazzo, Rex Lei, and Jessica Sorrell. Reproducibility in Learning. In *Theory and Practice of Differential Privacy*, 2021.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[Ioa05] John P. A. Ioannidis. Why most published research findings are false. *PLoS Med 2(8): e124*, 2005.

[JKT20] Young Hun Jung, Baekjin Kim, and Ambuj Tewari. On the equivalence between online and private learnability beyond binary classification. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[JLN$^+$20] Christopher Jung, Katrina Ligett, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Moshe Shenfeld. A new analysis of differential privacy's generalization guarantees. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science*

Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA, volume 151 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Kea98] M. J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

[KK09] A. Kalai and V. Kanade. Potential-based agnostic boosting. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009*, pages 880–888, 2009.

[KKL$^+$24] Alkis Kalavasis, Amin Karbasi, Kasper Green Larsen, Grigoris Velegkas, and Felix Zhou. Replicable learning of large-margin halfspaces, 2024.

[KKMN09] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *Proceedings of the 18th international conference on World wide web*, pages 171–180, 2009.

[KKMV23] Alkis Kalavasis, Amin Karbasi, Shay Moran, and Grigoris Velegkas. Statistical indistinguishability of learning algorithms. Personal communication, January 2023.

[KKVZ24] Alkis Kalavasis, Amin Karbasi, Grigoris Velegkas, and Felix Zhou. On the computational landscape of replicable learning, 2024.

[KLM$^+$20] Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 2263–2285. PMLR, 2020.

[KLN$^+$11] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[KMV08] A. Kalai, Y. Mansour, and E. Verbin. On agnostic boosting and parity learning. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 629–638, 2008.

[KORW12] Guy Kindler, Ryan O'Donnell, Anup Rao, and Avi Wigderson. Spherical cubes: Optimal foams from computational hardness amplification. *Communications of the ACM*, 55, 10 2012.

[KR99] M. J. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Comput.*, 11, 1999.

[KS99] A. R. Klivans and R. A. Servedio. Boosting and hard-core sets. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99*, pages 624–633. IEEE Computer Society, 1999.

[KS03] A. Kalai and R. Servedio. Boosting in the presence of noise. In *Proceedings of the 35th Annual Symposium on Theory of Computing (STOC)*, pages 196–205, 2003.

[KS14] Shiva P. Kasiviswanathan and Adam Smith. On the 'semantics' of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality*, 6(1), Jun. 2014.

[KSS94] M. Kearns, R. Schapire, and L. Sellie. Toward Efficient Agnostic Learning. *Machine Learning*, 17(2/3):115–141, 1994.

[KT02] Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002.

[KVYZ23] Amin Karbasi, Grigoris Velegkas, Lin F. Yang, and Felix Zhou. Replicability in reinforcement learning, 2023.

[LKM$^+$18] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Lon01] Philip M Long. On agnostic learning with $\{0,*, 1\}$-valued and real-valued hypotheses. In *International Conference on Computational Learning Theory*, pages 289–302. Springer, 2001.

[LS05] P. Long and R. Servedio. Martingale boosting. In *Proc. 18th Annual Conference on Learning Theory (COLT)*, pages 79–94, 2005.

[LS08] P. Long and R. Servedio. Adaptive martingale boosting. In *Proc. 22nd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 977–984, 2008.

[LS10] P. M. Long and R. A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, 2010.

[LS19] Katrina Ligett and Moshe Shenfeld. A necessary and sufficient stability notion for adaptive generalization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11481–11490, 2019.

[MHS19] Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. In *Conference on Learning Theory*, pages 2512–2530. PMLR, 2019.

[MM02] Y. Mansour and D. McAllester. Boosting using branching programs. *Journal of Computer & System Sciences*, 64(1):103–112, 2002.

[MN06] P. Massart and E. Nedelec. Risk bounds for statistical learning. *Ann. Statist.*, 34(5):2326–2366, 10 2006.

[Mou] Nima Mousavi. How tight is chernoff bound? Unpublished manuscript.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[MSS23] Shay Moran, Hilla Schefler, and Jonathan Shafer. The bayesian stability zoo. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 61725–61746. Curran Associates, Inc., 2023.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, page 94103, USA, 2007. IEEE Computer Society.

[Nar22] Shyam Narayanan. Private high-dimensional hypothesis testing. In Po-Ling Loh and Maxim Raginsky, editors, *Conference on Learning Theory, 2-5 July 2022, London, UK*, volume 178 of *Proceedings of Machine Learning Research*, pages 3979–4027. PMLR, 2022.

[Nat19] National Academies of Sciences, Engineering, and Medicine. *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC, 2019.

[NFPH15] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. Verifiable differential privacy. In Laurent Réveillère, Tim Harris, and Maurice Herlihy, editors, *Proceedings of the Tenth European Conference on Computer Systems, EuroSys 2015, Bordeaux, France, April 21-24, 2015*, pages 28:1–28:14. ACM, 2015.

[NK19] Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

[NSS$^+$18] Kobbi Nissim, Adam Smith, Uri Stemmer, Thomas Steinke, and Jonathan Ullman. The limits of post-selection generalization. *Advances in Neural Information Processing Systems*, 31, 2018.

[PVLS$^+$20] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:2003.12206*, 2020.

[RP10] Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In Paul Hudak and Stephanie Weirich, editors, *Proceeding of the 15th ACM SIGPLAN ICFP 2010, Baltimore, Maryland, USA, September 27-29, 2010*, pages 157–168. ACM, 2010.

[RRS$^+$20] Ryan Rogers, Aaron Roth, Adam D. Smith, Nathan Srebro, Om Thakkar, and Blake E. Woodworth. Guaranteed validity for empirical approaches to adaptive data analysis. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2830–2840. PMLR, 2020.

[RRST16] Ryan M. Rogers, Aaron Roth, Adam D. Smith, and Om Thakkar. Max-information, differential privacy, and post-selection hypothesis testing. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 487–494. IEEE Computer Society, 2016.

[RRT$^+$16] Maxim Raginsky, Alexander Rakhlin, Matthew Tsao, Yihong Wu, and Aolin Xu. Information-theoretic analysis of stability and bias of learning algorithms. In *2016 IEEE Information Theory Workshop, ITW 2016, Cambridge, United Kingdom, September 11-14, 2016*, pages 26–30. IEEE, 2016.

[RS94] R. Rivest and R. Sloan. A formal model of hierarchical concept learning. *Information and Computation*, 114(1):88–114, 1994.

[RTTV08] O. Reingold, L. Trevisan, M. Tulsiani, and S. Vadhan. Dense subsets of pseudorandom sets. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, page 7685, USA, 2008. IEEE Computer Society.

[RW78] W. H. Rogers and T. J. Wagner. A Finite Sample Distribution-Free Performance Bound for Local Discrimination Rules. *The Annals of Statistics*, 6(3):506 – 514, 1978.

[RZ16] Daniel Russo and James Zou. Controlling bias in adaptive data analysis using information theory. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 1232–1240. JMLR.org, 2016.

[SBG21] Satchit Sivakumar, Mark Bun, and Marco Gaboardi. Multiclass versus binary differentially private pac learning. *Advances in Neural Information Processing Systems*, 34:22943–22954, 2021.

[Sch90] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Sch03] R. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

[Ser02] Rocco A. Servedio. Pac analogues of perceptron and winnow via boosting the margin. *Machine Learning*, 47:133–151, 2002.

[Ser03] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

[SF12] R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[Slo88] R. H. Sloan. Types of noise in data for concept learning. In *Proceedings of the First Annual Workshop on Computational Learning Theory*, COLT '88, pages 91–96, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[Slo92] R. H. Sloan. Corrigendum to types of noise in data for concept learning. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992*, page 450, 1992.

[Slo96] R. H. Sloan. *Pac Learning, Noise, and Geometry*, pages 21–41. Birkhäuser Boston, Boston, MA, 1996.

[SSSSS10] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research*, 11:2635–2670, 2010.

[SZ20] Thomas Steinke and Lydia Zakynthinou. Reasoning about generalization via conditional mutual information. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 3437–3452. PMLR, 2020.

[TTV09] Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *IEEE Conference on Computational Complexity*, pages 126–136, 2009.

[Val84] L. G. Valiant. A theory of the learnable. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 436–445. ACM Press, 1984.

[Vap82] V. Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics*. Springer-Verlag, Berlin, Heidelberg, 1982.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16, 1971.

[VC74] Vladimir Vapnik and Alexey Chervonenkis. Theory of pattern recognition, 1974.

[vL10] Ulrike von Luxburg. Clustering stability: An overview. *Foundations and TrendsÂ® in Machine Learning*, 2(3):235–274, 2010.

[XR17] Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2524–2533, 2017.

[ZH19] Tijana Zrnic and Moritz Hardt. Natural analysts in adaptive data analysis. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7703–7711. PMLR, 2019.

[ZK17] Danfeng Zhang and Daniel Kifer. Lightdp: towards automating differential privacy proofs. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 888–901. ACM, 2017.

[ZLC17] Y. Zhang, P. Liang, and M. Charikar. A hitting time analysis of stochastic gradient langevin dynamics. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017*, pages 1980–2022, 2017.

[ZSA20] C. Zhang, J. Shen, and P. Awasthi. Efficient active learning of sparse halfspaces with arbitrary bounded noise. *CoRR*, abs/2002.04840, 2020.