Stability is Stable: Connections between Replicability, Privacy, and Adaptive Generalization

Mark Bun[*]   Marco Gaboardi [†]   Max Hopkins[‡]   Russell Impagliazzo[§]   Rex Lei[¶]

Toniann Pitassi[‖]   Satchit Sivakumar[**]   Jessica Sorrell[††]

March 28, 2023

Abstract

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.

Contents

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 1.1. *A randomized algorithm* $A : \mathcal{X}^n \to \mathcal{Y}$ *is ρ-replicable if for every distribution D over $\mathcal{X}$, we have*

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

where $S_1, S_2 \in \mathcal{X}^n$ *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 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 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

---

[1][ILPS22] stated this definition under the name "reproducibility." See Section 2.6 for a discussion of why we refer to it as "replicability" instead.

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+16a, 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 1.2. *An algorithm $A : \mathcal{X}^n \to \mathcal{Y}$ is $(\beta, \varepsilon, \delta)$-perfectly generalizing if, for every distribution D over $\mathcal{X}$, 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 $\mathcal{O} \subseteq \mathcal{Y}$,*

$$e^{-\varepsilon}(\mathbf{Pr}_{Sim_D}[\mathcal{O}] - \delta) \leqslant \mathbf{Pr}[A(S) \in \mathcal{O}] \leqslant e^{\varepsilon}\mathbf{Pr}_{Sim_D}[\mathcal{O}] + \delta. \tag{1}$$

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

Definition 1.3. *An algorithm $A : \mathcal{X}^n \to \mathcal{Y}$ has (ε, δ)-max-information with respect to product distributions if for every set of outcomes $\mathcal{O} \subseteq (\mathcal{Y} \times \mathcal{X}^n)$ we have*

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

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

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.

1.1 Our Main Results

1.1.1 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 $(x, f(x)) \in \mathcal{X} \times \{0, 1\}$ where x is drawn from an arbitrary marginal distribution over $\mathcal{X}$, and f is an arbitrary function from a fixed concept class H. A "good" outcome for such a distribution D is a hypothesis $h : \mathcal{X} \to \{0, 1\}$ 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 6.4 for other examples.)

Figure 1 illustrates the known relationships between the various stability notions that hold with respect to any statistical task.
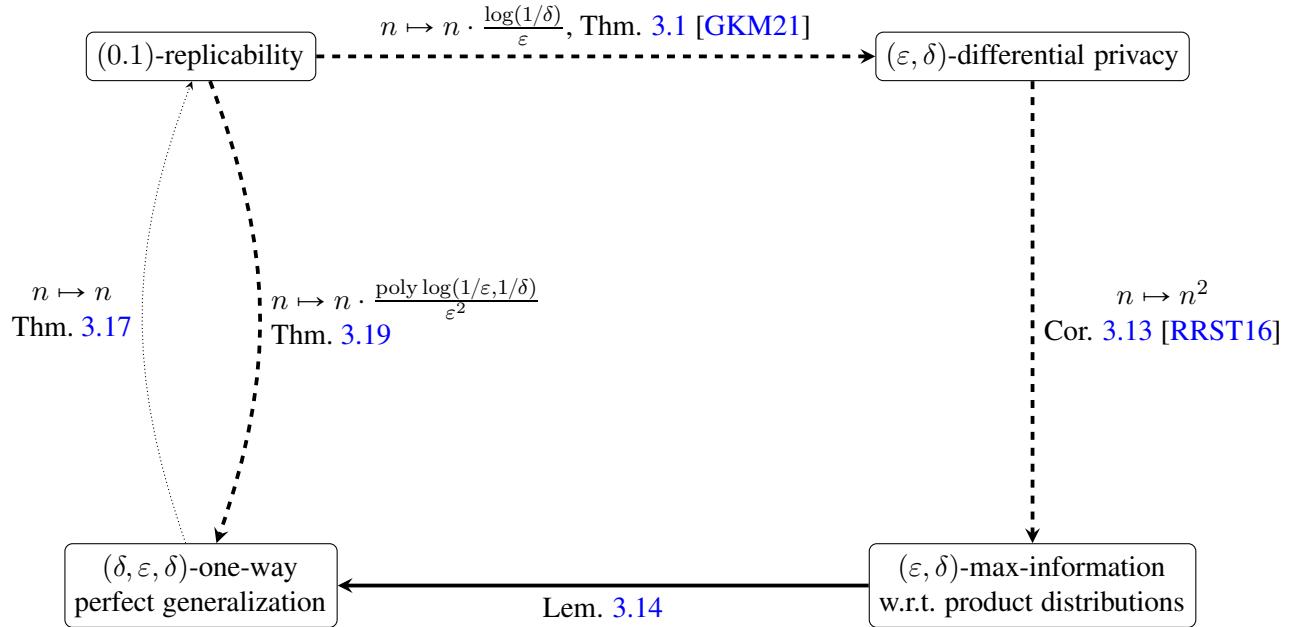
Figure 1: The solid arrow from A to B means that every algorithm satisfying A also satisfies B. A dashed arrow means that for every statistical task, a solution satisfying A can be computationally efficiently transformed into a solution satisfying B with the stated blowup in sample complexity. The thin 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.

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

6

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 6.4.

Equivalence between perfect generalization and differential privacy. For simplicity, the relationships summarized in Figure 1 are stated in terms of a one-way variant of perfect generalization, where only the inequality on the right of (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 3.17, and Theorem 3.19 actually yields the stronger conversion back to a two-way perfectly generalizing algorithm (See Theorem 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 $n \geqslant O(\log(1/\delta)/\varepsilon)$ 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 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 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 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.

1.1.2   Separating Stability: Computational Barriers and the Complexity of Correlated Sampling

All of the transformations appearing in Figure 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 1.2 and 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 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 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 2.5 for more discussion.)

1.1.3   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

---

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

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 5.1).

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 5.2).

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 5.3). 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 $\tilde{O}_H(1/\alpha^2)$.

1.2 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 $\mathcal{Y}$ is finite (Theorem 3.17). This can be done via a primitive called *correlated sampling* (See Section 2.5). A correlated sampling algorithm for a class of distributions $\mathcal{P} = \{P\}$ 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 $P, Q \in \mathcal{P}$ satisfy $d_{TV}(P, Q) \leqslant \eta$, we have $\mathbf{Pr}[CS(P, r) = CS(Q, r)] \geqslant 1 - O(\eta)$. 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 1 holds) are replicable via correlated sampling.

Conversely, we show how to convert replicable algorithms to perfectly generalizing ones (Theorem 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). 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 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 3.17 and 3.19, it follows that bounded max-information also implies perfect generalization for finite outcome spaces (Theorem 6.3).

Replicability implies differential privacy. In Theorem 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 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 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 3.12 and 3.13).

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 in learning has a long history as a tool for ensuring generalization. Early work [RW78, DW79, 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+16a, 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.

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 from Lemma 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 1.4. *Fix* $n \in \mathbb{N}$, $\beta, \varepsilon, \delta \in (0,1]$. *Let* $\mathcal{X}$ *be a countable domain and* $A : \mathcal{X}^n \to \mathcal{Y}$ *be a* $(\beta, \varepsilon, \delta)$-*one-way perfectly generalizing algorithm for a statistical task. Then there exists an algorithm* $A' : \mathcal{X}^n \to \mathcal{Y}$ *that is* $\left(\frac{2\rho}{1+\rho}\right)$-*replicable for* $\rho = 4\varepsilon + 2\delta + 2\beta$, *and for all* $S \in \mathcal{X}^n$, $A(S) = A'(S)$.

---

[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].

Whether a transformation exists for general measure spaces remains open.[4] In Section 6.3.1 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 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 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 5.3, 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 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 C, 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?

2 Preliminaries

We start by formally defining a statistical task.

Definition 2.1. *A statistical task* with data domain $\mathcal{X}$ and output space $\mathcal{Y}$ is a set of pairs $\mathcal{T} = \{(D, G_D)\}$, *where D is a distribution over $\mathcal{X}$ and $G_D \subseteq \mathcal{Y}$ is a "good" set of outputs for distribution D. A randomized*

---

[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.

algorithm $\mathcal{A}$ solves statistical task $\mathcal{T}$ using m samples and with failure probability β if for every $(D, G_D) \in$ $\mathcal{T}$,

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

2.1 Notions of Distributional Closeness

We recall the definition of total variation distance, that will be crucial in this work.

Definition 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 2.3 ((ε, δ)-indistinguishability)**.** *Let P and Q be probability distributions over some domain $\mathcal{Y}$. Then, we say that P is (ϵ, δ)-indistinguishable from Q (denoted as $P \approx_{\varepsilon, \delta} Q$) if for all $O \subseteq \mathcal{Y}$,*

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

We will frequently talk about random variables being (ε, δ)-indistinguishable, which means that their distributions are (ε, δ)-indistinguishable.

2.2 Differential Privacy

We say that two datasets $S, S' \in \mathcal{X}^n$ 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 2.4 (Differential Privacy [DMNS16])**.** *A randomized algorithm $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ is said to be (ε, δ)-differentially private if for every pair of neighboring datasets $S, S' \in \mathcal{X}^n$, we have that for all subsets $O \subseteq \mathcal{Y}$,*

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

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

One important property of differential privacy is that it is closed under post-processing by arbitrary functions.

Lemma 2.5 (Post-Processing [DMNS16])**.** *If $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ is (ε, δ)-differentially private, and $\mathcal{B} : \mathcal{Y} \to \mathcal{Z}$ is any randomized function, then the algorithm $\mathcal{B} \circ \mathcal{A}$ 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 2.6 (ℓ_1-Sensitivity)**.** *Let $f : \mathcal{X}^n \to \mathbb{R}^d$ be a function. Its ℓ_1-sensitivity is*

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

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 2.7 (Exponential Mechanism [MT07])**.** *Let L be a set of outputs and $g : L \times \mathcal{X}^n \to \mathbb{R}$ 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 $S \in \mathcal{X}^n$, outputs an element $m \in L$ such that, for all $a > 0$, we have*

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

Standard (ε, δ)-differential privacy automatically protects the privacy of groups of individuals.

Lemma 2.8 (Group Privacy [DMNS16])**.** *Let $k \in \mathbb{N}^+$ and let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an (ε, δ)-DP algorithm. Then for all datasets $S, S' \in \mathcal{X}^m$ such that $\|S - S'\|_0 \leqslant k$,*

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

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 2.9 (Secrecy of the sample, [KLN$^+$11, BBG18])**.** *Let $A : \mathcal{X}^n \to \mathcal{Y}$ be an (ε, δ)-differentially private algorithm. Consider the algorithm $A' : \mathcal{X}^m \to \mathcal{Y}$ 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*

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

2.3 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 2.10 ([ILPS22])**.** *Let D be a distribution over domain $\mathcal{X}$. Let $\mathcal{A}$ be a randomized algorithm that takes as input samples from D. We say that $\mathcal{A}$ is ρ-reproducible if*

$$\mathbf{Pr}_{S, S', r}[\mathcal{A}(S; r) = \mathcal{A}(S'; r)] \geqslant 1 - \rho,$$

where S, S' are sets of samples drawn i.i.d. from D and r represents the internal randomness of $\mathcal{A}$.

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 2.11 ([ILPS22])**.** *Let $A(S; r)$ be an algorithm operating on a sample set $S \in \mathcal{X}^n$ 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 $\mathbf{Pr}_{S \sim D^n}[A(S; r) = z_r] \geqslant 1 - \eta$. 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 1.1) are essentially equivalent:

Claim 2.12 ([ILPS22])**.** *For every $0 \leqslant \rho \leqslant v \leqslant 1$,*

---

[5]See Section 2.5 for an explanation of our renaming of this definition to "replicable."

1. *Every ρ-replicable algorithm is also $(\rho/v, v)$-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 2.13 (Amplification of Replicability, Theorem A.3, [ILPS22]). *Let $0 < \eta, \nu, \beta < \frac{1}{2}$ and $m > 0$. Let $\mathcal{A}$ 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 $\mathcal{A}'$ for D with sample complexity $m' = \tilde{O}(m(\log 1/\beta)^3/\rho^2(1/2 - \eta)^2)$ and failure probability at most $4(\beta + \rho)$.*

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 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 $\mathcal{X}$, consider m independent draws $x_1, x_2, \cdots x_m$ from distribution P. A labeled sample of size m is the set $\{(x_1, f(x_1)), (x_2, f(x_2)), \cdots, (x_m, f(x_m))\}$ where $f \in H$. 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,*

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

We also consider a variant called agnostic PAC learning, where the labels of the input dataset can be noisy.

Definition 2.15 (Agnostic PAC learning, [Hau92, VC74]). *A learning problem is defined by a hypothesis class H. We say an algorithm $\mathcal{A}$ is an (α, β)-accurate PAC learner for the hypothesis class H if for all distributions D over input, output pairs, $\mathcal{A}$ 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,*

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

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

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

2.5 Correlated Sampling

In the correlated sampling problem, two (or more) players are given probability distributions $\mathcal{P}, \mathcal{Q}$ 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 $\mathcal{P}, \mathcal{Q}$ over Ω, let $d_{TV}(\mathcal{P}, \mathcal{Q})$ denote the total variational distance between $\mathcal{P}$ and $\mathcal{Q}$.

Definition 2.17. *(Correlated Sampling) A correlated sampling strategy for a finite set Ω with error ε : $[0, 1] \to [0, 1]$ is an algorithm $CS : \Delta_{\Omega} \times \mathcal{R}'$ and a distribution $\mathcal{R}'$ on random strings such that:*

- *(Marginal Correctness) For all $\mathcal{P} \in \Delta_{\Omega}$ and $w \in \Omega$, $\mathbf{Pr}_{r' \sim \mathcal{R}'}[CS(\mathcal{P}, r') = w] = \mathcal{P}(w)$.*

- *(Error Guarantee) For all $\mathcal{P}, \mathcal{Q} \in \Delta_{\Omega}$, $\mathbf{Pr}_{r' \sim \mathcal{R}'}[CS(\mathcal{P}, r') \neq CS(\mathcal{Q}, r')] \leqslant \varepsilon(d_{TV}(\mathcal{P}, \mathcal{Q}))$*

Several independent papers [KT02, Hol09, Bro97] give correlated sampling strategies over finite sets, with $\epsilon(\delta) = \frac{2\delta}{1+\delta}$. 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 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 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 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.

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.

3 Equating Stability: Differential Privacy, Perfect Generalization, and Replicability

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 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 3.1 (Replicability → DP). *Let $\mathcal{T}$ be a statistical problem. For all $\beta > 0$, if there is a 0.01-replicable algorithm solving $\mathcal{T}$ using n_R samples and with failure probability β, then for any $0 < \varepsilon, \delta \leqslant 1$ there is an (ε, δ)-DP algorithm for $\mathcal{T}$ using n_{DP} samples with failure probability $O\left(\beta \log \frac{1}{\beta}\right)$, where*

$$n_{DP}(\epsilon, \delta, \beta) \leqslant n_R \cdot O\left(\frac{\log \delta^{-1} \log \beta^{-1}}{\varepsilon} + \log^2 \beta^{-1}\right)$$

This conversion relies on the following private algorithm for selecting an approximate mode.

Theorem 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 $S \in \mathcal{X}^m$, 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 $poly(m, \log(|\mathcal{X}|))$ 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 $\mathcal{A}$ on n samples, consider the following simple procedure adapted from [GKM21]: partition a larger data set, run $\mathcal{A}$ on each part, and

---

[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.

privately output a commonly repeated element.

Algorithm 1: DP-to-Replicability Reduction

Result: Privately ouputs solution to $(\mathcal{X}, R)$

Input: Statistical Problem $(\mathcal{X}, R)$. Distribution D over $\mathcal{X}$, Replicable algorithm $\mathcal{A}$ 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 $S_{i,j} \sim \mathcal{X}^n$

2. Sample k_1 random strings $\{r_j\}$.

3. Let $y_{i,j} = \mathcal{A}(S_{i,j}; r_j)$.

4. Run (ε, δ)-DP Selection on $\{y_{i,j}\}$ and denote the output by y^*.

return y^*

Recalling the two-parameter definition of replicability (2.11 and 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 3.1 is an immediate consequence of the following proposition.

Proposition 3.3. *For all sufficiently small $\beta, \varepsilon, \delta > 0$, if $\mathcal{A}$ is $(0.1, 0.1)$-replicable and has failure probability β, then Algorithm 1 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 $\mathcal{A}$ 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 $\mathcal{A}$ 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.

$\square$

3.2 Approximate-DP Implies One-Way Perfect Generalization

3.2.1 Preliminaries about Perfect Generalization

Perfect generalization (Definition 1.2) is a notion of stability that captures the idea (like differential privacy) that an algorithm $\mathcal{A}$ 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 3.4. *An algorithm $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ is said to be $(\beta, \varepsilon, \delta)$-sample perfectly generalizing, if for every distribution D over $\mathcal{X}$, with probability at least $1 - \beta$ over the draw of two i.i.d. samples $S_1, S_2 \sim D^m$, $\mathcal{A}(S_1) \approx_{\varepsilon, \delta} \mathcal{A}(S_2)$.*

Cummings et al. [CLN+16] prove the following lemma relating perfect generalization to sample perfect generalization.

Lemma 3.5 ([CLN+16]). *If algorithm $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ 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 3.6. *Fix $\beta, \varepsilon, \delta \in (0, 1]$. If algorithm $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ is $(\beta, \varepsilon, \delta)$-sample perfectly generalizing, then it is also $(\sqrt{\beta}, \varepsilon, \delta + \sqrt{\beta})$-perfectly generalizing.*

Proof. Since $\mathcal{A}$ is $(\beta, \varepsilon, \delta)$-sample PG, we have that for any distribution D over $\mathcal{X}$, 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 $\mathcal{A}(S_1) \approx_{\varepsilon, \delta} \mathcal{A}(S_2)$. This guarantees by the reverse Markov inequality that for $d = 1 - \sqrt{\beta}$,

$$\mathbb{E}_{S_1}[\mathbf{Pr}_{S_2}(\mathcal{A}(S_1) \approx_{\varepsilon, \delta} \mathcal{A}(S_2))] \geqslant 1 - \beta \implies$$
$$\mathbf{Pr}_{S_1}[\mathbf{Pr}_{S_2}(\mathcal{A}(S_1) \approx_{\varepsilon, \delta} \mathcal{A}(S_2)) \geqslant d] \geqslant \frac{1 - \beta - d}{1 - d} \implies$$
$$\mathbf{Pr}_{S_1}[\mathbf{Pr}_{S_2}(\mathcal{A}(S_1) \approx_{\varepsilon, \delta} \mathcal{A}(S_2)) \geqslant 1 - \sqrt{\beta}] \geqslant 1 - \sqrt{\beta}.$$

Now, set Sim_D to be the distribution of $\mathcal{A}(S_2)$ 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 $\mathcal{A}(S_1) \approx_{\varepsilon, \delta} \mathcal{A}(S_2)$.

Then, we get that with probability at least $1 - \sqrt{\beta}$ over the draw of S_1, for any $O \subseteq Y$,

$$\begin{aligned}
\mathbf{Pr}_{Sim_D}[O] &= \mathbf{Pr}_{S_2, \mathcal{A}}[\mathcal{A}(S_2) \in O] \\
&= \mathbf{Pr}_{S_2, \mathcal{A}}[\mathcal{A}(S_2) \in O \mid S_2 \in G_{S_1}]\mathbf{Pr}[S_2 \in G_{S_1}] + \mathbf{Pr}_{S_2, \mathcal{A}}[\mathcal{A}(S_2) \in O \mid S_2 \notin G_{S_1}]\mathbf{Pr}[S_2 \notin G_{S_1}] \\
&\leqslant \mathbf{Pr}_{S_2, \mathcal{A}}[\mathcal{A}(S_2) \in O \mid S_2 \in G_{S_1}] + \sqrt{\beta} \\
&\leqslant e^{\varepsilon}\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S_1) \in O] + \delta + \sqrt{\beta}.
\end{aligned}$$

---

[7]Note this choice can be taken universally with respect to all parameters and the statistical problem itself.

Similarly, we can also argue that

$$
\begin{aligned}
\mathbf{Pr}_{Sim_D}[O] &= \mathbf{Pr}_{S_2,\mathcal{A}}[\mathcal{A}(S_2) \in O] \\
&= \mathbf{Pr}_{S_2,\mathcal{A}}[\mathcal{A}(S_2) \in O \mid S_2 \in G_{S_1}]\mathbf{Pr}[S_2 \in G_{S_1}] + \mathbf{Pr}_{S_2,\mathcal{A}}[\mathcal{A}(S_2) \in O \mid S_2 \notin G_{S_1}]\mathbf{Pr}[S_2 \notin G_{S_1}] \\
&=\geqslant \mathbf{Pr}_{S_2,\mathcal{A}}[\mathcal{A}(S_2)] \in O \mid S_2 \in G_{S_1})(1 - \sqrt{\beta}) \\
&\geqslant e^{-\varepsilon}\left(\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S_1) \in O] - \delta\right)(1 - \sqrt{\beta}) \\
&\geqslant e^{-\varepsilon}\left(\mathbf{Pr}_{\mathcal{A}}(\mathcal{A}(S_1) \in O] - \delta - \sqrt{\beta}\right).
\end{aligned}
$$

Hence, with probability at least $1 - \sqrt{\beta}$ over the draw of S_1,

$$
Sim_D \approx_{\varepsilon,\delta} \mathcal{A}(S_1).
$$

$\square$

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 3.7 (One-way perfect generalization). *An algorithm $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ is said to be $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing if for every distribution D over $\mathcal{X}$, 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 $O \subseteq \mathcal{Y}$ we have that*

$$
\mathbf{Pr}[\mathcal{A}(S) \in O] \leqslant e^{\varepsilon}\mathbf{Pr}_{Sim_D}[O] + \delta.
$$

Next, we prove a simple lemma relating the parameters achievable with perfect generalization.

Lemma 3.8. *Fix $m \in \mathbb{N}$, $\beta, \varepsilon, \delta \in (0,1]$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be a $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing algorithm. Then, $\mathcal{A}$ 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 $\mathcal{X}$, there exists a distribution Sim_D, such that with probability $1 - \beta$ over the draw of S, for all $O \subseteq \mathcal{Y}$,

$$
\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O] \leqslant e^{\varepsilon}\mathbf{Pr}_{Sim_D}[O] + \delta.
$$

Using the fact that for $\varepsilon \leqslant 1$, $e^{\varepsilon} \leqslant 1 + 2\varepsilon$, we get that

$$
\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O] \leqslant \mathbf{Pr}_{Sim_D}[O](1 + 2\varepsilon) + \delta,
$$

which gives us that

$$
\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O] \leqslant \mathbf{Pr}_{Sim_D}[O] + 2\varepsilon + \delta.
$$

Now, since this works for any set O, consider applying it to O^c. Then, we get that

$$
\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O^c] \leqslant \mathbf{Pr}_{Sim_D}[O^c] + 2\varepsilon + \delta,
$$

which implies (by writing $\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O^c] = 1 - \mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O]$ and likewise for Sim_D and doing some algebraic manipulation) that

$$
\mathbf{Pr}_{Sim_D}[O] \leqslant \mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O] + 2\varepsilon + \delta.
$$

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 3.9 ([CLN$^+$16]). *If algorithm $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ is $(\varepsilon, 0)$-differentially private, then for all $\beta > 0$, it is $(\beta, \varepsilon\sqrt{2m \ln(2|\mathcal{Y}|/\beta)}, 0)$-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.

3.2.2 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 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*

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

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.

3.2.3 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 3.11 ([RRST16]). *Fix $m \in \mathbb{N}$, $\varepsilon \in (0, 1/2]$ and $\delta \in [0, \varepsilon/15)$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an (ε, δ)-DP algorithm. Then, for any distribution D over $\mathcal{X}$, if $S \sim D^m$, for all $t > 0$, $I_\infty^\beta(S; \mathcal{A}(S)) \leqslant m\nu + 6t\varepsilon\sqrt{m}$, 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 3.12. *Fix $m \in \mathbb{N}$, sufficiently small $\rho \in (0, 1)$. Let $\varepsilon = \frac{\rho}{\sqrt{8m \log(1/\rho)}}$, $\delta \leqslant \frac{\varepsilon\rho^6}{m^2}$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an (ε, δ)-DP algorithm. Then, for any distribution D over $\mathcal{X}$, if $S \sim D^m$, $I_\infty^{\rho^3}(S; \mathcal{A}(S)) \leqslant O(\rho)$.*

Proof. Substituting the value of ε, δ, and setting $t = \sqrt{8\log(1/\rho)}$ in the expression for β in Lemma 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 3.11 gives

$$\nu = C\left(\varepsilon^2 + \sqrt{\frac{\delta}{\varepsilon}}\right) \leqslant C\left(\frac{\rho^2}{m} + \frac{\rho^3}{m}\right) = O\left(\frac{\rho^2}{m}\right).$$

Substituting the values of ν, t, and ε in the upper bound for max information gives

$$I_\infty^{\rho^3}(S; \mathcal{A}(S)) \leqslant m\nu + 6t\varepsilon\sqrt{m} = \rho^2 + 6\rho = O(\rho).$$

$\square$

In general, one can convert an (ε, δ)-differentially private algorithm with $\varepsilon = O(1)$ and $\delta = 1/\operatorname{poly}(n)$ into a bounded max-information algorithm by first amplifying the privacy parameters:

Corollary 3.13. *There are constants $c, C > 0$ such that the following holds. Let $\gamma > 0$. Suppose $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ is an (ε, δ)-differentially private algorithm solving a statistical task up to some failure probability β such that $\varepsilon \in (0, 1)$ and $\delta \leqslant \min\{\gamma^2/c^2, \rho^2/C^2\}\rho^4/64\varepsilon^3(2C\rho + \sqrt{72\log(2/\gamma)})^2 n^4$. Then there is an algorithm $\mathcal{A}' : \mathcal{X}^m \to \mathcal{Y}$ 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 $\mathcal{X}$, if $S \sim D^m$ we have $I_\infty^\gamma(S; \mathcal{A}(S)) \leqslant \rho$. Moreover, the conversion from $\mathcal{A}$ to $\mathcal{A}'$ preserves computational efficiency.

Proof. The algorithm $\mathcal{A}'$ simply samples n items without replacement from S and runs $\mathcal{A}$ on the result. This perfectly preserves correctness with respect to any statistical task. By Lemma 2.9, we have that $\mathcal{A}'$ is (ε', δ')-differentially private for

$$\varepsilon' = \frac{2n}{m} \leqslant \frac{1}{\sqrt{m}} \min\left\{\sqrt{\frac{\rho}{2C}}, \frac{\rho}{\sqrt{72\log(2/\gamma)}}\right\}, \qquad \delta' = \frac{n}{m} \cdot \delta.$$

Note that these parameters ensure that

$$\frac{\delta'}{\varepsilon'} = \frac{\delta}{e^\varepsilon - 1} \leqslant \frac{\delta}{\varepsilon} \leqslant \min\left\{\frac{\gamma^2}{4c^2m^2}, \frac{\rho^2}{4C^2m^2}\right\}.$$

Now set $t = \sqrt{2\log(2/\gamma)}$ in the statement of Lemma 3.11, which ensures that

$$e^{-t^2/2} + cm\sqrt{\frac{\delta'}{\varepsilon'}} \leqslant \frac{\gamma}{2} + cm\sqrt{\frac{\gamma^2}{4c^2m^2}} \leqslant \gamma.$$

Then the lemma implies that

$$\begin{aligned}
I_\infty^\gamma(S; \mathcal{A}(S)) &\leqslant Cm\left((\varepsilon')^2 + \sqrt{\frac{\delta'}{\varepsilon'}}\right) + 6t\varepsilon'\sqrt{m} \\
&\leqslant Cm\left(\frac{\rho}{2Cm} + \frac{\rho}{2Cm}\right) + 6\sqrt{2\log(2/\gamma)} \cdot \frac{\rho}{\sqrt{72\log(2/\gamma)}} \\
&\leqslant \rho.
\end{aligned}$$

$\square$

3.2.4 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 3.14. *Fix $m \in \mathbb{N}$, $k > 0$, $\beta \in (0, 1)$ and $\hat{\beta} = \sqrt{\frac{\beta}{1 - 2^{-k}}}$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an algorithm. Then, for every distribution D over $\mathcal{X}$ and $S \sim D^n$, if $I_\infty^\beta(S; \mathcal{A}(S)) \leqslant k$, then $\mathcal{A}$ is $(\hat{\beta}, 2k, \hat{\beta})$-one-way perfectly generalizing.*

Proof. The canonical distribution Sim_D we will consider is the distribution of $\mathcal{A}(S')$, where the randomness is over both $S' \sim D^m$, and the internal randomness of $\mathcal{A}$.

We start by defining a set of 'bad' outputs for each fixed dataset S, i.e. outputs on which the probability mass of $\mathcal{A}(S)$ is substantially larger than that of the canonical distribution Sim_D. Formally, for each dataset S, let

$$B_S = \{y \in \mathcal{Y} : \mathbf{Pr}_\mathcal{A}[\mathcal{A}(S) = y] \geqslant 2^{2k}\mathbf{Pr}_{S' \sim D^m, \mathcal{A}}[\mathcal{A}(S') = y]\}.$$

Next, we define a set of ordered pairs consisting of datasets and their corresponding 'bad' outputs. Formally, let

$$\Theta = \{(S, y) : S \in \mathtt{supp}(D^m), y \in B_S\}.$$

Our goal will be to prove that with high probability over a draw of a dataset S, $\mathcal{A}(S)$ 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 $\mathbb{E}_S[\mathbf{Pr}_\mathcal{A}[\mathcal{A}(S) \in B_S]] = \mathbb{E}_S[\mathbf{Pr}_\mathcal{A}[(S, \mathcal{A}(S)) \in \Theta]]$. By the law of total probability, this is equal to $\mathbf{Pr}_{S \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S)) \in \Theta]$. Using the definition of max-information, we get that

$$\mathbf{Pr}_{S \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S)) \in \Theta] \leqslant 2^k \mathbf{Pr}_{S, S' \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S')) \in \Theta] + \beta. \tag{3}$$

Now, analyzing the term $\mathbf{Pr}_{S, S' \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S')) \in \Theta]$, we get

$$\begin{aligned}
\mathbf{Pr}_{S, S' \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S')) \in \Theta] &= \sum_{T \in \mathtt{supp}(D^m)} \mathbf{Pr}[S = T]\mathbf{Pr}_{S' \sim D^m, \mathcal{A}}[\mathcal{A}(S') \in B_T] \\
&\leqslant \sum_{T \in \mathtt{supp}(D^m)} \mathbf{Pr}[S = T]2^{-2k}\mathbf{Pr}_\mathcal{A}[\mathcal{A}(S) \in B_T] \\
&= 2^{-2k} \sum_{T \in \mathtt{supp}(D^m)} \mathbf{Pr}[S = T]\mathbf{Pr}_{\mathcal{A}, S \sim D^m}[A(S) \in B_S \mid S = T] \\
&= 2^{-2k}\mathbf{Pr}_{S \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S)) \in \Theta],
\end{aligned}$$

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 (3), we get that

$$\mathbf{Pr}_{S \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S)) \in \Theta] \leqslant 2^{-k}\mathbf{Pr}_{S \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S) \in \Theta] + \beta.$$

Rearranging, this gives us that

$$\mathbb{E}_S[\mathbf{Pr}_\mathcal{A}[\mathcal{A}(S) \in B_S]] = \mathbf{Pr}_{S \sim D^m, \mathcal{A}}[(S, \mathcal{A}(S)) \in \Theta] \leqslant \frac{\beta}{1 - 2^{-k}} = \hat{\beta}^2. \tag{4}$$

Finally, by Markov's inequality and the above equation, we can write the following.

$$\mathbf{Pr}_S[\mathbf{Pr}_\mathcal{A}[\mathcal{A}(S) \in B_S] > \hat{\beta}] \leqslant \frac{\mathbb{E}_S[\mathbf{Pr}_\mathcal{A}[\mathcal{A}(S) \in B_S]]}{\hat{\beta}} \leqslant \hat{\beta}. \tag{5}$$

This implies that with probability $1 - \hat{\beta}$ over $S \sim D^m$, $\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \notin B_S] > 1 - \hat{\beta}$. Finally, we can write with probability $1 - \hat{\beta}$ over $S \sim D^m$, for every $O \subseteq \mathcal{Y}$,

$$
\begin{aligned}
\mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O] &= \mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O \mid \mathcal{A}(S) \in B_S]\mathbf{Pr}[\mathcal{A}(S) \in B_S] \\
&\quad + \mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O, \mathcal{A}(S) \notin B_S] \\
&\leqslant 1 \cdot \hat{\beta} + \mathbf{Pr}_{\mathcal{A}}[\mathcal{A}(S) \in O, \mathcal{A}(S) \notin B_S] \\
&\leqslant \hat{\beta} + 2^{2k}\mathbf{Pr}_{S' \sim D^m, \mathcal{A}}[\mathcal{A}(S') \in O, \mathcal{A}(S') \notin B_{S'}] \\
&\leqslant \hat{\beta} + 2^{2k}\mathbf{Pr}_{S' \sim D^m, \mathcal{A}}[\mathcal{A}(S') \in O].
\end{aligned}
$$

This completes the proof. $\qquad\square$

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. 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 3.11 and Lemma 3.14 above gives the following connection between differential privacy and one-way perfect generalization.

Corollary 3.15. *Fix $m \in \mathbb{N}$, $\varepsilon \in (0, 1/2]$ and $\delta \in [0, \varepsilon/15)$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an (ε, δ)-DP algorithm. Then, for sufficiently large constants c, C, for all $t > 0$, $\mathcal{A}$ 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 3.12).

Corollary 3.16. *Fix $m \in \mathbb{N}$, sufficiently small $\rho \in (0, 1]$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an (ε, δ)-DP algorithm, where $\varepsilon = \frac{\rho}{\sqrt{8m\log(1/\rho)}}$, $\delta \leqslant \frac{\varepsilon\rho^6}{m^2}$. Then, $\mathcal{A}$ is $(O(\rho), O(\rho), O(\rho))$-one-way perfectly generalizing.*

Proof. From Corollary 3.12, we get that $I_\infty^{\rho^3}(S; \mathcal{A}(S)) \leqslant O(\rho)$. Substituting $k = \rho$ and $\beta = \rho^3$ in Lemma 3.14, we get that $\hat{\beta} = \sqrt{\frac{c\rho^3}{1 - 2^{-O(\rho)}}} \leqslant O(\sqrt{\rho^2}) = O(\rho)$ (where the first inequality is since $\frac{1}{1 - 2^{-O(\rho)}} = \frac{2^{O(\rho)}}{2^{O(\rho)} - 1} \leqslant \frac{2}{2^{O(\rho)} - 1} \leqslant \frac{C}{\rho}$ for some constant C, since $2^{c\rho} = e^{c\rho \ln 2}$ and $e^x \geqslant 1 + x$ for all real x). This gives us from Lemma 3.14 that $\mathcal{A}$ is $(O(\rho), O(\rho), O(\rho))$-one-way perfectly generalizing. $\qquad\square$

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 $CS(Q, \mathcal{Y}, r')$ represent a correlated sampling procedure over domain $\mathcal{Y}$ sampling from a distribution Q over $\mathcal{Y}$ with public randomness r'. (See Section 2.5 for background on correlated sampling). We now describe our transformation.

Algorithm 2: Transformation from one-way perfectly generalizing algorithm to replicable algorithm

1   **Input:** dataset $S = (s_1, \dots, s_n)$, description of one-way perfectly generalizing algorithm $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$

2   **Output:** $i \in \mathcal{U}$

    1: Let Q_S represent the distribution of $\mathcal{A}(S)$.

    2: Output $CS(Q_S, \mathcal{Y}, r')$ 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 3.17. *Fix $m \in \mathbb{N}$ and $\beta, \varepsilon, \delta \in (0,1)$. Let $A : \mathcal{X}^m \to \mathcal{Y}$ be a $(\beta, \varepsilon, \delta)$-one-way perfectly generalizing algorithm with finite output space. Then, for any distribution D over $\mathcal{X}$, if $S \sim D^m$, Algorithm 2 when run on dataset S and with access to $\mathcal{A}$ is $4(\beta + 2\varepsilon + \delta)$-replicable.*

Proof. By Lemma 3.8, we have that $\mathcal{A}$ is also $(\beta, 0, 2\varepsilon + \delta)$-perfectly generalizing. For any distribution D over $\mathcal{X}$, 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 $S \sim D^m$,
$$d_{TV}(\mathcal{A}(S), Sim_D) \leqslant 2\varepsilon + \delta.$$
From the guarantee of correlated sampling, we have that
$$\mathbf{Pr}_{r' \sim R}[CS(Q_S, \mathcal{Y}, r') \neq CS(Sim_D, \mathcal{Y}, r')] \leqslant 2d_{TV}(Q_S, Sim_D).$$

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
$$\mathbf{Pr}_{r' \sim R}[CS(Q_{S_1}, \mathcal{Y}, r') \neq CS(Sim_D, \mathcal{Y}, r')] \leqslant 2(2\varepsilon + \delta)$$
and
$$\mathbf{Pr}_{r' \sim R}[CS(Q_{S_2}, \mathcal{Y}, r') \neq CS(Sim_D, \mathcal{Y}, r')] \leqslant 2(2\varepsilon + \delta).$$
Consider the event $CS(Q_{S_1}, \mathcal{Y}, r') \neq CS(Q_{S_2}, \mathcal{Y}, r')$. It is clear that this implies that either $CS(Q_{S_1}, \mathcal{Y}, r') \neq CS(Sim_D, \mathcal{Y}, r')$ or $CS(Q_{S_2}, \mathcal{Y}, r') \neq CS(Sim_D, \mathcal{Y}, r')$. Hence, we can write that with probability $1 - 2\beta$ over draws of S_1 and S_2,
$$\mathbf{Pr}_{r' \sim R}[CS(Q_{S_1}, \mathcal{Y}, r') \neq CS(Q_{S_2}, \mathcal{Y}, r')] \leqslant 4(2\varepsilon + \delta).$$

Taking the expectation with respect to the draws of S_1 and S_2 gives us
$$\mathbf{Pr}_{r' \sim R, S_1, S_2 \sim D^m}[CS(Q_{S_1}, \mathcal{Y}, r') \neq CS(Q_{S_2}, \mathcal{Y}, r')] \leqslant 4(2\varepsilon + \delta + \beta)$$
which proves the result. $\qquad \square$

Combining the above result and Corollary 3.16, we get a transformation from approximate differentially private algorithms to replicable algorithms.

Corollary 3.18. *Fix $m \in \mathbb{N}$, sufficiently small $\rho \in (0,1)$. Let $\varepsilon = \frac{\rho}{\sqrt{8m \log(1/\rho)}}$, $\delta \leqslant \frac{\varepsilon \rho^6}{m^2}$. Let $\mathcal{A} : \mathcal{X}^m \to \mathcal{Y}$ be an (ε, δ)-DP algorithm with finite output space. Fix a distribution D over $\mathcal{X}$, and let $S \sim D^m$. Then, Algorithm 2 run with inputs S and algorithm $\mathcal{A}$ is $O(\rho)$-replicable. Additionally, on any fixed dataset, the output distribution of Algorithm 2 is the same as that of $\mathcal{A}$.*

Proof. From Corollary 3.16, we have that Algorithm $\mathcal{A}$ is $(O(\rho), O(\rho), O(\rho))$-one-way perfectly generalizing. Then, applying Theorem 3.17 proves that the transformation in Algorithm 2 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. $\qquad \square$

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) 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 3: Transformation from replicable algorithm $\mathcal{A}$ to perfectly generalizing algorithm $\mathcal{A}'$

---

1   **Input:** Sample access to distribution D, description of $(0.01, 0.01)$-replicable algorithm $\mathcal{A} : \mathcal{X}^* \to \mathcal{Y}$, sample complexity parameter m, perfect generalization parameters $\varepsilon, \delta, \beta$

2   **Output:** $y \in \mathcal{Y}$

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 $\mathcal{A}$.

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 $\mathcal{A}$ with coins r_j and sample $S_{i,j}$ to get output $z_{i,j}$.

7:    **end for**

8:    Let $c_j = \arg\max_{z \in \mathcal{Y}} \sum_{i=1}^t \mathbb{1}[z_{i,j} = z]$, and let
score $((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k)) = \sum_{i=1}^t \mathbb{1}[z_{i,j} = c_j]$.

9: **end for**

10: 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 ε to get value (j^*, c_{j*}).

11: **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 3.6.

Theorem 3.19. *Fix sufficiently small $\delta, \gamma > 0$ and $0 < \varepsilon \leqslant 1$. Every $(0.01, 0.01)$-replicable algorithm $\mathcal{A}$ with m samples that succeeds on a statistical task with probability at least $1 - \gamma^2$ can be converted to a $(2\delta, \varepsilon, 2\delta)$-sample perfectly generalizing algorithm $\mathcal{A}'$ taking $O\left(\frac{m \log(1/\varepsilon)}{\varepsilon^2} \text{poly} \log(1/\delta)\right)$ 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 $\mathcal{X}$. Our proof is constructive; the corresponding algorithm is given in Algorithm 3 ($\mathcal{A}'$), and we feed it with the following inputs: a description of algorithm $\mathcal{A}$, 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 3.

Claim 3.20. *Algorithm $\mathcal{A}'$ (represented in Algorithm 3) 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 3 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 $\mathcal{A}$ 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 3 are bad is at most $(0.01)^k \leqslant \delta^2$ 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 $\mathcal{A}$ 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 3 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 $Score\left((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_k)\right)$, 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 $\ell \in \mathcal{Y}$. 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 $\mathcal{A}$ when run with coin r_j on a random sample.

Then, using uniform convergence (Theorem 2.16), we get for every fixed j and for every $\gamma > 0$, that

$$\mathbf{Pr}_{z_{1,j},\dots,z_{t,j} \sim Q_j} \left[\sup_{h_\ell \in H} \left| \frac{1}{t} \sum_{i=1}^t \mathbb{1}[h_\ell(z_{i,j}) = 1] - \mathbf{Pr}_{z \sim Q_j}[h_\ell(z) = 1] \right| \geqslant \gamma \right] \leqslant 8t e^{-\gamma^2 t/8}.$$

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

$$\mathbf{Pr}_{z_{1,j},\dots,z_{t,j} \sim Q_j} \left[\sup_{\ell \in \mathcal{Y}} \left| \frac{1}{t} \sum_{i=1}^t \mathbb{1}[z_{i,j} = \ell] - \mathbf{Pr}_{z \sim Q_j}[z = \ell] \right| \geqslant \gamma \right] \leqslant 8t e^{-\gamma^2 t/8}.$$

Setting $\gamma = 2\sqrt{\frac{\log(8kt/\delta)}{t}}$, we get that

$$\mathbf{Pr}_{z_{1,j},\dots,z_{t,j} \sim Q_j} \left[\sup_{\ell \in \mathcal{Y}} \left| \sum_{i=1}^t \mathbb{1}[z_{i,j} = \ell] - t\mathbf{Pr}_{z \sim Q_j}[z = \ell] \right| \geqslant 2\sqrt{t \log(8kt/\delta)} \right] \leqslant \frac{\delta^2}{2k}.$$

Using a union bound over all k stages of the algorithm, this guarantees us that the empirical frequencies (and in particular, the values $score\left((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_k)\right)$ 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 $S' = (\mathbf{S}_1', \mathbf{S}_2', \dots, \mathbf{S}_k')$ 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 $\ell \in \mathcal{Y}$ that $score\left((j, \ell), (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_k)\right)$ and $score\left((j, \ell), (\mathbf{S}_1', \mathbf{S}_2', \dots, \mathbf{S}_k')\right)$ 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 $j \in [k]$,

$$\left| score\left((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_k)\right) - score\left((j, c_j'), (\mathbf{S}_1', \mathbf{S}_2', \dots, \mathbf{S}_k')\right) \right| \leqslant 4\sqrt{t \log(8kt/\delta)}. \qquad (6)$$

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 $score\left((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_k)\right) - score\left((j, c_j'), (\mathbf{S}_1', \mathbf{S}_2', \dots, \mathbf{S}_k')\right) \leqslant 4\sqrt{t \log(8kt/\delta)}$; 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 3 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 $2\sqrt{t \log(8kt/\delta)} \leqslant 0.09t$. 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 $((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k))$ 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 2.7), we have that

$$\mathbf{Pr}\left[\max_{j \in [k]} \text{score}\,((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k)) - \text{score}\,((j^*, c_{j*}), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k)) \geqslant 2\Delta \frac{\ln k + k}{\varepsilon}\right] \leqslant e^{-k} = \delta,$$

where $\Delta = 4\sqrt{t \log(8kt/\delta)}$. Hence, 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

$$\mathbf{Pr}\left[\text{score}\,((j^*, c_{j*}), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k)) \geqslant 0.9t - 2\Delta \frac{\ln k + k}{\varepsilon}\right] \leqslant e^{-k} = \delta,$$

which implies that for sufficiently small δ,

$$\mathbf{Pr}\left[\text{score}\,((j^*, c_{j*}), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k)) \geqslant 0.8t\right] \leqslant e^{-k} = \delta.$$

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 3 is run on the sample S'. Hence, c'_{j*} is also equal to c_{j*}.

Step 5: Proving that $\mathcal{A}'(S, r) \approx_{\varepsilon, \delta} \mathcal{A}'(S', r)$ w.h.p. We exploit the fact that two random variables C and D are (ε, δ)-indistinguishable if w.p. $\geqslant 1 - \delta$ over a draw o from the distribution of C, we have $e^{-\varepsilon}\mathbf{Pr}[D = o] \leqslant \mathbf{Pr}[C = o] \leqslant e^{\varepsilon}\mathbf{Pr}[D = o]$, 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 6, which points out that even though the output sets are different, the scores score $((j, c_j), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k))$ and score $\left((j, c'_j), (\mathbf{S}'_1, \mathbf{S}'_2, \ldots, \mathbf{S}'_k)\right)$ can differ by at most the sensitivity specified in Step 10 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 $\mathcal{A}'(S, r)$, we get that

$$e^{-\varepsilon}\mathbf{Pr}[\mathcal{A}'(S, r) = (j^*, c_{j*})] \leqslant \mathbf{Pr}[\mathcal{A}'(S', r) = (j^*, c'_{j*})] \leqslant e^{\varepsilon}\mathbf{Pr}(\mathcal{A}'(S, r) = (j^*, c_{j*})]$$

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 $\mathcal{A}'(S', r)$,

$$e^{-\varepsilon}\mathbf{Pr}[\mathcal{A}'(S, r) = (j^*, c_{j*})] \leqslant \mathbf{Pr}[\mathcal{A}'(S', r) = (j^*, c_{j*})] \leqslant e^{\varepsilon}\mathbf{Pr}[\mathcal{A}'(S, r) = (j^*, c_{j*})].$$

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 $\mathcal{A}'(S, r) \approx_{\varepsilon, \delta} \mathcal{A}'(S', r)$. Now, using the law of total probability, we get that

$$\mathbf{Pr}_{r_1, \ldots, r_k} \left[\mathbf{Pr}_{S, S' \sim D^{mkt}} \left[\mathcal{A}'(S; r_1, \ldots, r_k) \approx_{\varepsilon, \delta} \mathcal{A}'(S'; r_1, \ldots, r_k) \right] \geqslant 1 - \delta^2 \right] \geqslant 1 - \delta^2. \qquad (7)$$

Step 6: Switch quantifiers to get sample perfect generalization: Now, we switch the quantifiers in equation 7.

$$\mathbf{Pr}_{S, S' \sim D^{mkt}, r_1, \ldots, r_k} \left[\mathcal{A}'(S; r_1, \ldots, r_k) \approx_{\varepsilon, \delta} \mathcal{A}'(S'; r_1, \ldots, r_k) \right] \geqslant 1 - 2\delta^2$$

$$\implies \mathbb{E}_{S, S' \sim D^{mkt}} \left[\mathbf{Pr}_{r_1, \ldots, r_k} \left[\mathcal{A}'(S; r_1, \ldots, r_k) \approx_{\varepsilon, \delta} \mathcal{A}'(S'; r_1, \ldots, r_k) \right] \right] \geqslant 1 - 2\delta^2$$

$$\implies \mathbf{Pr}_{S, S' \sim D^{mkt}} \left[\mathbf{Pr}_{r_1, \ldots, r_k} \left[\mathcal{A}'(S; r_1, \ldots, r_k) \approx_{\varepsilon, \delta} \mathcal{A}'(S'; r_1, \ldots, r_k) \right] \geqslant d \right] \geqslant \frac{1 - 2\delta^2 - d}{1 - d}.$$

Here, the last inequality holds by the reverse Markov inequality. Setting $d = 1 - \delta$, we get that

$$\mathbf{Pr}_{S, S' \sim D^{mkt}} \left[\mathbf{Pr}_{r_1, \ldots, r_k} \left[\mathcal{A}'(S; r_1, \ldots, r_k) \approx_{\varepsilon, \delta} \mathcal{A}'(S'; r_1, \ldots, r_k) \right] \geqslant 1 - \delta \right] \geqslant 1 - 2\delta.$$

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

$$\mathbf{Pr}_{S, S' \sim D^{mkt}} \left[\mathcal{A}'(S) \approx_{\varepsilon, 2\delta} \mathcal{A}'(S') \right] \geqslant 1 - 2\delta.$$

This proves that $\mathcal{A}'$ with the specified inputs is $(2\delta, \varepsilon, 2\delta)$-sample perfectly generalizing, as required. Next, we deal with accuracy. $\qquad \square$

Claim 3.21. *If Algorithm $\mathcal{A}$ succeeds at a statistical task with probability at least $1 - \gamma^2$, Algorithm $\mathcal{A}'$ 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 $\mathcal{A}$ 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

$$\mathbf{Pr}_r \left[\mathbf{Pr}_{S \sim D^m} \left[\mathcal{A}(S; r) \in O_D \right] \geqslant 1 - \gamma \right] \geqslant 1 - \gamma \qquad (8)$$

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 $(1 - \gamma)^{\log(1/\delta)} \geqslant 1 - \gamma \log(1/\delta)$. (This follows from Bernoulli's inequality $(1 + a)^k \geqslant 1 + ak$ for all $a \geqslant -1$ 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

$$\mathbf{Pr} \left[\text{score} \left((j^*, c_{j*}), (\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k) \right) \geqslant 0.8t \right] \leqslant e^{-k} = \delta,$$

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 8 would not be satisfied). Hence, conditioned on $E_{coin-acc}$, we have that $\mathcal{A}'$ 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 $\mathcal{A}'$ outputs an element of the good set with probability at least $1 - \delta - 2\delta^2 - \gamma \log(1/\delta) = 1 - O(\delta) - \gamma \log(1/\delta)$. $\square$

Hence, combining the two claims on perfect generalization and accuracy, we complete the proof of the theorem. $\square$

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 3.18).

In Section 4.1, we define $\Pi_{RandEnc}$, a statistical promise problem. Given a public key $\mathbf{pk}$ and a dataset of ciphertexts encrypting the same bit b under $\mathbf{pk}$, a solution to $\Pi_{RandEnc}$ is any encryption of b under $\mathbf{pk}$. In Section 4.1.1, we give a simple algorithm DPRandEnc solving $\Pi_{RandEnc}$. DPRandEnc is (ε, δ)-differentially private and runs in polynomial time. In Section 4.1.2, 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 4.2, but to demonstrate that these requirements are not unreasonable, in Section 4.1.3 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 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 2, to implement the correlated sampling step of the transformation from a one-way perfectly generalizing algorithm to a replicable one, giving an efficient transformation.

4.1 Cryptographic Hardness of Replicability

We define a promise problem $\Pi_{RandEnc}$ (Definition 4.3) for a public-key encryption scheme. $\Pi_{RandEnc}$ is parameterized by a public-key $\mathbf{pk}$ for a public-key encryption scheme $\mathcal{E}$ with message space $\{0, 1\}$ and ciphertext space $\mathcal{C}$. An instance of $\Pi_{RandEnc}$ consists of a sample of m elements c_i, drawn i.i.d. from an unknown distribution D over $\mathcal{C}$. 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 4.1 (Public-Key Encryption Scheme)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a tuple of algorithms running in time* $\mathrm{poly}(\lambda)$*, with* $\mathsf{KeyGen} : 1^* \to \mathcal{K}_p \times \mathcal{K}_s$*,* $\mathsf{Enc} : \mathcal{K}_p \times \{0,1\} \to \mathcal{C}$*, and* $\mathsf{Dec} : \mathcal{K}_s \times \mathcal{C} \to \{0,1\} \cup \bot$*. We say $\mathcal{E}$ is a* public-key encryption scheme *if it has the following properties.*

- *Correctness: Let $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$ and $c \leftarrow \mathsf{Enc}(\mathbf{pk}, b)$ for $b \in \{0,1\}$. Then $\mathsf{Dec}(\mathbf{sk}, c) = b$.*

- *Security: There exists a negligible function $\varepsilon(\lambda)$, such that for all adversaries $\mathcal{A}$ running in time* $\mathrm{poly}(\lambda)$*, letting $(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$ we have*

$$|\mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)]| < \varepsilon(\lambda).$$

We also require that a public-key encryption scheme allows for efficient, publicly computable ciphertext verification and rerandomization procedures.

Definition 4.2 (Randomizeable Encryption Scheme)**.** *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. We call $\mathcal{E}$ a* randomizeable encryption scheme *if it supports the following additional procedures.*

- *(Perfect) Verification of Ciphertexts: There exists a deterministic polytime algorithm $\mathcal{V}$ 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 $\mathcal{V}(\mathbf{pk}, c) = 1$*
    - *If $\mathsf{Dec}(\mathbf{sk}, c) = \bot$, then $\mathcal{V}(\mathbf{pk}, c) = 0$*

- *(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 $\mathbf{pk}$ for an encryption scheme $\mathcal{E}$, and an i.i.d. sample of m elements from a distribution D supported on encryptions under $\mathbf{pk}$ of a fixed bit $b \in \{0,1\}$, output an encryption of b under $\mathbf{pk}$.

Definition 4.3 (Ciphertext Identification Problem)**.** *An instance of $\Pi_{RandEnc}$ is defined as follows. Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a randomizeable encryption scheme (Definition 4.2). Let $\lambda, m \in \mathbb{N}$, and let D be a distribution over the ciphertext space $\mathcal{C}$ of $\mathcal{E}$. 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 $c^* \in \mathcal{C} \cup \bot$ such that*

1. *If $\mathsf{Dec}(\mathbf{sk}, c) = 1$ for all $c \in \mathrm{supp}(D)$, $\mathsf{Dec}(\mathbf{sk}, c^*) = 1$*

2. *If $\mathsf{Dec}(\mathbf{sk}, c) = 0$ for all $c \in \mathrm{supp}(D)$, $\mathsf{Dec}(\mathbf{sk}, c^*) = 0$*

4.1.1 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., $\mathcal{V}(\mathbf{pk}, c_i) = 0$. 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 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 4: DPRandEnc

Result: Outputs a ciphertext $c \in \mathcal{C}$

Input: Sample S of m elements drawn i.i.d. from D

Parameters:

- Privacy ε, failure probability β, 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 $\mathcal{V}(\mathbf{pk}, c_i) = 0$

2. Add k ciphertexts Enc$(\mathbf{pk}, 0)$ to the dataset

3. Add k ciphertexts Enc$(\mathbf{pk}, 1)$ to the dataset

4. Choose c uniformly at random from the new dataset

return Rerandomize$(\mathbf{pk}, c)$

Lemma 4.4. *Let $\varepsilon, \beta \in (0, 1/2)$. Then for $m \in \Omega(1/(\varepsilon\beta))$ and $k = 1/\varepsilon$, $\Pi_{RandEnc}$ (Algorithm 4) 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 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 Dec$(\mathbf{sk}, c) = b$,

$\mathsf{Dec}(\mathbf{sk}, c) = \neg b$, and $\mathsf{Dec}(\mathbf{sk}, c) = \bot$ respectively, and let S'_b, $S'_{\neg b}$, and $S'_\bot$ be defined analogously. Then

$$
\begin{aligned}
\frac{\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = b \mid S]}{\mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = b \mid S']} &= \frac{|S_b| + k}{m - |S_\bot| + 2k} \cdot \frac{m - |S'_\bot| + 2k}{|S'_b| + k} \\
&= \frac{|S_b| + k}{|S_b| + |S_{\neg b}| + 2k} \cdot \frac{|S'_b| + |S'_{\neg b}| + 2k}{|S'_b| + k} \\
&\leqslant \frac{|S_b| + k}{|S_b| + |S_{\neg b}| + 2k} \cdot \frac{|S_b| + |S_{\neg b}| + 2k}{|S_b| - 1 + k} \\
&= \frac{|S_b| + k}{|S_b| - 1 + k} \\
&\leqslant \frac{k + 1}{k},
\end{aligned}
$$

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 $|S_b| \geqslant 1$ 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 $T \subseteq \mathcal{C}$,

$$
\begin{aligned}
\mathbf{Pr}[\texttt{DPRandEnc}(S) \in T] = \ &\mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk}, c) \in T \mid \mathsf{Dec}(\mathbf{sk}, c) = 1] \cdot \mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = 1 \mid S] \\
&+ \mathbf{Pr}[\mathsf{Rerandomize}(\mathbf{pk}, c) \in T \mid \mathsf{Dec}(\mathbf{sk}, c) = 0] \cdot \mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = 0 \mid S].
\end{aligned}
$$

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

$$
\begin{aligned}
\mathbf{Pr}[\texttt{DPRandEnc}(S) \in T] &= p_1 \cdot \mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = 1 \mid S] + p_0 \cdot \mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = 0 \mid S] \\
&\leqslant \frac{k + 1}{k} \left(p_1 \cdot \mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = 1 \mid S'] + p_0 \cdot \mathbf{Pr}[\mathsf{Dec}(\mathbf{sk}, c) = 0 \mid S'] \right) \\
&= \frac{k + 1}{k} \cdot \mathbf{Pr}[\texttt{DPRandEnc}(S') \in T] \\
&\leqslant e^\varepsilon \cdot \mathbf{Pr}[\texttt{DPRandEnc}(S') \in T]
\end{aligned}
$$

where the final inequality follows from taking $k = 1/\varepsilon$ and observing $1 + x \leqslant e^x$.

It remains to argue correctness of $\texttt{DPRandEnc}$ 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. $\qquad\square$

4.1.2 Cryptographic Adversary from Replicable Algorithm for $\Pi_{RandEnc}$

In this subsection, we show that if there exists a replicable polytime algorithm, $\mathcal{B}$, for $\Pi_{RandEnc}$, instantiated with a randomizable encryption scheme $\mathcal{E}$, then there exists an adversary breaking the security guarantee of $\mathcal{E}$. 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 $\mathcal{B}$ 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 $\mathcal{B}$ is a replicable algorithm for $\Pi_{RandEnc}$, if c encrypts a 0, $\mathcal{B}$ will with high probability produce the same output ciphertext for both invocations. If c encrypts a 1, $\mathcal{B}$ 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 5: Adversary $\mathcal{A}$

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. $c_0 \leftarrow \mathcal{B}(\mathbf{pk}, S_0; r)$

5. Generate a sample S of m ciphertexts by running $\mathsf{Rerandomize}(\mathbf{pk}, c)$ m times

6. $c \leftarrow \mathcal{B}(\mathbf{pk}, S; r)$

7. if $c_0 = c$, then $b' = 0$, otherwise $b' = 1$

return b'

Lemma 4.5. *Let $\mathcal{E}$ be a randomizeable encryption scheme, let $\Pi^{\mathcal{E}}_{RandEnc}$ denote the instantiation of $\Pi_{RandEnc}$ with $\mathcal{E}$. Let $\mathcal{B}$ be a ρ-replicable algorithm for $\Pi^{\mathcal{E}}_{RandEnc}$ with failure probability β, running in time $\mathrm{poly}(\lambda, \rho, \beta)$, and with sample complexity $m \in \mathrm{poly}(\lambda, \rho, \beta)$. Then there exists an adversary $\mathcal{A}$ running in time $\mathrm{poly}(\lambda, \rho, \beta)$ such that*

$$\mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)] \geqslant 1 - 2\beta - \rho.$$

Proof. The adversary $\mathcal{A}(\mathbf{pk}, c)$ 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 $\mathcal{B}$ 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 $\mathcal{B}$ fails to be replicable, which happens with probability at most ρ. Therefore

$$\mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)] \geqslant 1 - 2\beta - \rho.$$

$\square$

In particular, taking β, ρ to be constant in Lemma 4.5 gives an adversary breaking the security of $\mathcal{E}$, yielding the following theorem as a corollary.

Theorem 4.6. *Let $\mathcal{E}$ be a randomizeable encryption scheme, and let $\Pi^{\mathcal{E}}_{RandEnc}$ denote the instantiation of $\Pi_{RandEnc}$ with $\mathcal{E}$. Then there does not exist a ρ-replicable algorithm for $\Pi^{\mathcal{E}}_{RandEnc}$ 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 $\mathcal{B}$ for $\Pi_{RandEnc}^{\mathcal{E}}$ with failure probability $\beta < 1/8$ and replicability parameter $\rho < 1/4$ running in time $\text{poly}(\lambda)$, then by Lemma 4.5, there exists an adversary $\mathcal{A}$ running in time $\text{poly}(\lambda)$ such that

$$\mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 1)] - \mathbf{Pr}[\mathcal{A}(\mathbf{pk}, c) = 1 \mid c \leftarrow \mathsf{Enc}(\mathbf{pk}, 0)] \geqslant 1/2 > \text{negl}(\lambda),$$

and therefore $\mathcal{A}$ breaks the security of $\mathcal{E}$. $\qquad\square$

4.1.3 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 4.7 (Goldwasser-Micali Cryptosystem ([GM82]))**.** *The Goldwasser-Micali cryptosystem is defined over a plaintext message space $\mathcal{M} = \{0, 1\}$ and ciphertext space $\mathcal{C} = \mathbb{Z}_N^*$, 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 $u \leftarrow_{\mathcal{U}} \mathbb{Z}_N^*$ and output $u^2 x^b \mod N$.*

- $\mathsf{Dec}(\mathbf{sk}, c)$*: 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 $\mathcal{V}(\mathbf{pk}, c)$ 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 $(\mathbf{pk}, r^2 c \mod N)$.

Lemma 4.8. *The Goldwasser-Micali cryptosystem is a rerandomizeable encryption scheme (definition 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 $\mathcal{V}(\mathbf{pk}, c) = 0$ if and only if $\gcd(c, N) \neq 1$, $\mathcal{V}$ satisfies the requirement of definition 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 $\mathsf{Dec}(\mathbf{sk}, \mathsf{Rerandomize}(\mathbf{pk}, c)) = \mathsf{Dec}(\mathbf{sk}, c)$.

To show that for all c, c' such that $\mathsf{Dec}(\mathbf{sk}, c) = \mathsf{Dec}(\mathbf{sk}, c')$, $d_{TV}(\mathsf{Rerandomize}(\mathbf{pk}, c), \mathsf{Rerandomize}(\mathbf{pk}, c')) = 0$, let $b \in \{0, 1\}$, $c = u^2 x^b \mod N$ and $c' = v^2 x^b \mod N$ be honest encryptions of b under $\mathbf{pk} = (N, x)$.

It follows that

$$
\begin{aligned}
\mathbf{Pr}[\text{Rerandomize}(\mathbf{pk}, c) = a] &= \mathbf{Pr}[\text{Rerandomize}(\mathbf{pk}, u^2 x^b \mod N) = a] \\
&= \mathbf{Pr}[r^2 u^2 x^b = a \mod N] \\
&= \mathbf{Pr}[r^2 = u^{-2} x^{-b} a \mod N] \\
&= \mathbf{Pr}[(u^{-1} v r)^2 = u^{-2} x^{-b} a \mod N] \\
&= \mathbf{Pr}[r^2 v^2 x^b = a \mod N] \\
&= \mathbf{Pr}[\text{Rerandomize}(\mathbf{pk}, c') = a],
\end{aligned}
$$

where the fourth equality follows from r being chosen uniformly at random from $\mathbb{Z}_N^*$. $\qquad\square$

4.2 Correlated Sampling via One-Way Function Inverters

As we saw in Section 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 4.9. *Assuming uniform one-way function inverters exist (Definition 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 4.14. The correlated sampling property is shown in the proof of Lemma 4.16. The runtime of `CorrSamp` is shown in the proof of Lemma 4.17. $\qquad\square$

4.2.1 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 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 $\nu > 0$. $\mathcal{B}(C, \nu; r)$ is an (m, n, ν)-implicit correlated sampling algorithm if the following conditions hold:*

1. ***Inputs/Outputs:*** *$\mathcal{B}$ takes as input a circuit $C : \{0,1\}^m \to \{0,1\}^n$, a distributional error parameter ν, and a random string r. $\mathcal{B}$ 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 $D_{\mathcal{B}(C,\nu)}$ satisfy $d_{TV}(D_C, D_{\mathcal{B}(C,\nu)}) \leqslant O(\nu)$.*

   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, $D_{\mathcal{B}(C,\nu)}$ denotes the distribution over $\{0,1\}^n$ induced by querying $\mathcal{B}(C, \nu; r)$ with uniformly random strings r.

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

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

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 $\mathcal{I}$ returns "$\perp$" if it does not succeed.

Our correlated sampler randomly samples from pairwise-independent hash families in its subroutines.

Definition 4.12 (Pairwise-Independent Hash Family)**.** *A family of Boolean functions $\mathcal{H} = \{H | H : \{0,1\}^m \rightarrow \{0,1\}^n\}$ is* pairwise-independent *if, for all $r_1 \neq r_2 \in \{0,1\}^m$ and $x_1, x_2 \in \{0,1\}^n$, $\mathbf{Pr}_{H \in \mathcal{H}}[H(r_1) = x_1 \wedge H(r_2) = x_2] = 2^{-2n}$.*

4.2.2 Algorithm Overview

A correlated sampling algorithm $\mathcal{B}$ accomplishes two goals. First, $\mathcal{B}(C, \nu; r)$ needs to accurately sample from the distribution D_C. Second, $\mathcal{B}$ 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, $\mathcal{B}$ 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 $C : \{0,1\}^m \rightarrow \{0,1\}^n$ 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 $H : \{0,1\}^n \rightarrow \{0,1\}^{\ell+k}$

2. Draw a random string $u \in \{0,1\}^{\ell+k}$

3. Run the inverter on u: $r = \mathcal{I}_{\nu'}(C \circ H, u)$

4. If $r = \perp$ (i.e. u is not in the support of $C \circ H$), repeat. Else return $y^* = C(r)$
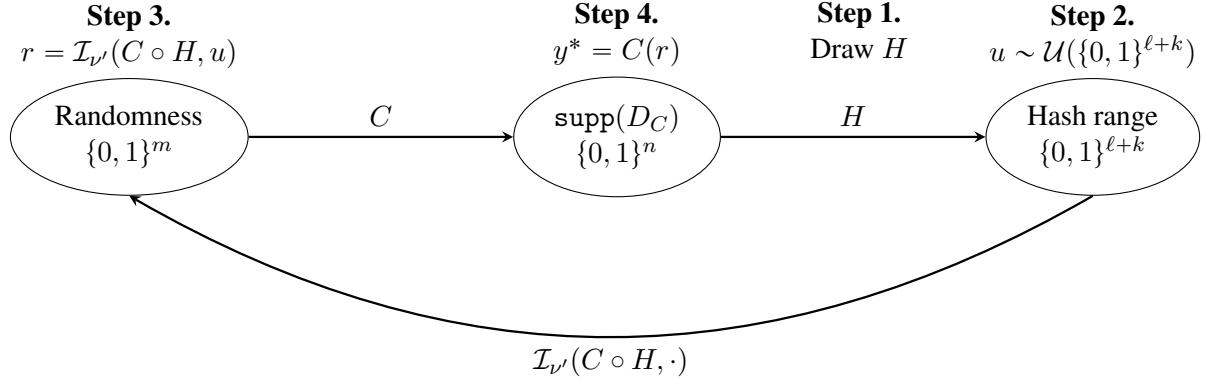
Figure 2: High-level structure of correlated sampler for uniform D_C

The high level idea is that k can be chosen large enough such that $\mathsf{supp}(D_C)$ has few collisions with good probability (for random H), but small enough s.t. 2^k (and therefore the runtime) remains polynomial

---

[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.

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 $\mathsf{supp}(C_1)\Delta\mathsf{supp}(C_2)$. 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). To do this, we call the inverter on the concatenated function $F_{C,l,H,H_2}(r) \overset{\text{def}}{=} H(C(r)) \parallel H_2(r)$[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.

4.2.3 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 4.2.2, 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 $\mathsf{ElemFind}_{C,\nu,\ell,\beta}(H_1, u)$ 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 4.2.2.

In the pseudocode for CorrSamp (Algorithm 6), 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 to ensure polynomial runtime. The value T_1 is chosen to be large enough that CorrSamp returns $x \neq \bot$ 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.

---

[9]We use '$\parallel$' 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.
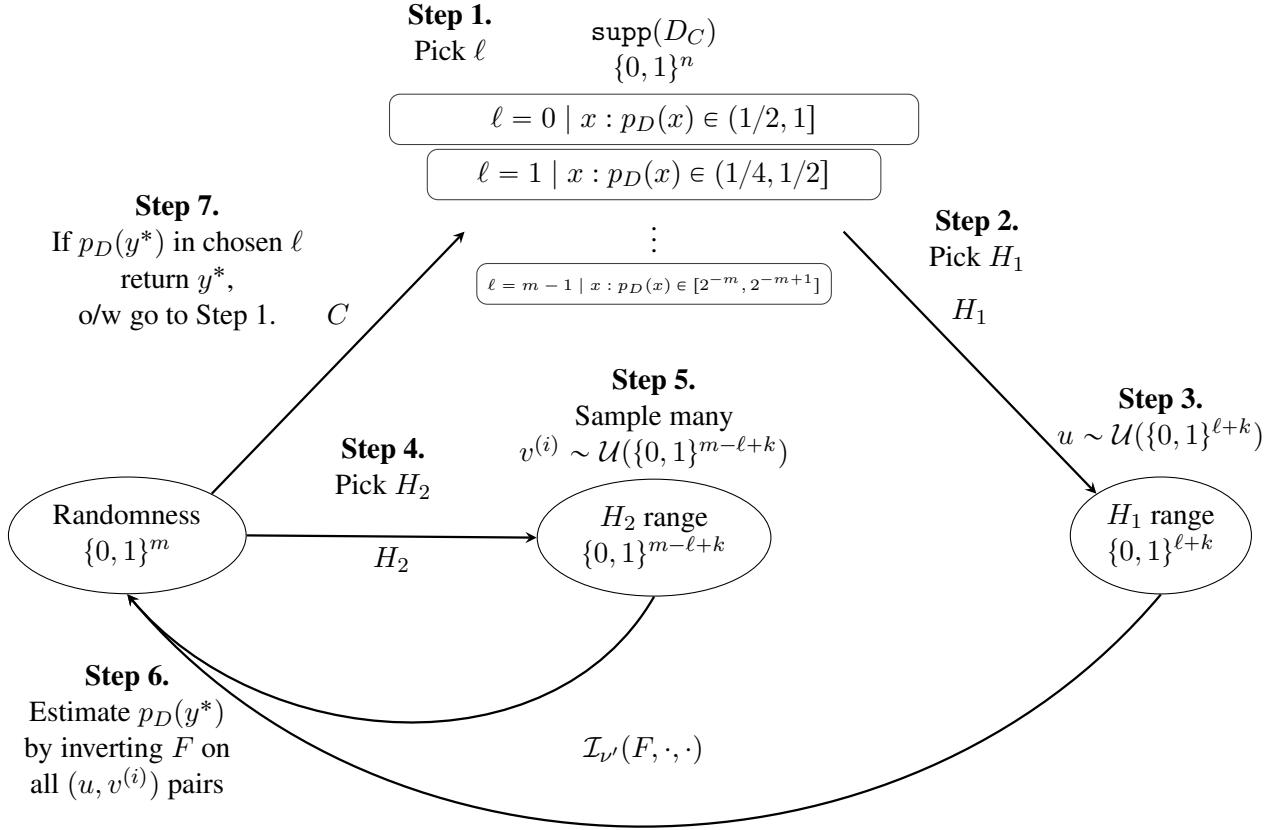
Figure 3: High-level structure of correlated sampler for general D_C

Algorithm 6: $\texttt{CorrSamp}(C, \nu; r)$

Input: Circuit $C : \{0,1\}^m \to \{0,1\}^n$, distributional error parameter ν, and random string r

Output: An element $x \in \{0,1\}^n$

$\quad k \leftarrow \Theta(\log(m) + \log(1/\nu))$

$\quad T_1 \leftarrow \Theta(mk2^k \log(1/\nu))$

$\quad \textbf{for } t = 1 \text{ to } t = T_1 \textbf{ do}$

$\quad\quad \beta \leftarrow_r [1, 2]$

$\quad\quad \ell \leftarrow_r \{0, 1, \ldots, m\}$

$\quad\quad H_1 \leftarrow_r$ pairwise independent hash function from n bits to $\ell + k$ bits

$\quad\quad u \leftarrow_r \{0, 1\}^{\ell+k}$

$\quad\quad x \leftarrow \texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u; r)$

$\quad\quad \textbf{if } x \neq \perp \textbf{ then}$

$\quad\quad\quad \textbf{return } x$

$\quad\quad \textbf{end if}$

$\quad \textbf{end for}$

$\quad \textbf{return } \perp$

The subroutine $\text{ElemFind}_{C,\nu,\ell,\beta}(H_1, u; r)$ follows the approach described in Section 4.2.2, 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 $\text{HashCheck}_{C,\nu,\ell,H_1,u}(H_2, v)$ 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 $\hat{q}(y^*)$ that is close to $p_D(y^*)2^{\ell-k}$. It will then return y^* only if $\frac{\beta}{2}2^{-k} < \hat{q}(y^*) \leqslant \beta 2^{-k}$, i.e. roughly when $\beta 2^{-\ell-1} < p_D(y^*) \leqslant \beta 2^{-\ell}$.

We now reach the core reason for choosing our threshold β randomly. As in Section 4.2.2, 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 4.14 that choosing β uniformly at random exactly achieves this.

In the pseudocode for ElemFind (Algorithm 7), 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 7: $\text{ElemFind}_{C,\nu,\ell,\beta}(H_1, u; r)$

(Explicit) Input: Hash function $H_1 : \{0, 1\}^n \rightarrow \{0, 1\}^{\ell+k}$, string $u \in \{0, 1\}^{\ell+k}$, random string r

(Implicit) Input: Circuit $C : \{0, 1\}^m \rightarrow \{0, 1\}^n$, 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 $\text{HashCheck}_{C,\nu,\ell,H_1,u}(H_2^i, v^i; r)$
    end for
    Let $\hat{q}_x$ denote the fraction of times x was returned by HashCheck
    if $\exists$ unique x s.t. $x \neq \perp$ and $\hat{q}_x \in ((\beta/2)2^{-k}, \beta 2^{-k}]$ **then**
        return x
    else
        return $\perp$
    end if

Algorithm 8: $\text{HashCheck}_{C,\nu,\ell,H_1,u}(H_2, v; r)$

(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 $\ell \in \{0, 1, \ldots, m\}$, 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 $m, n, 1/\nu$.

$r' \leftarrow \mathcal{I}_{\nu'}(F_{C,\ell,H_1,H_2}, (u \,\|\, v); r)$

if $r' = \bot$ **then**

    return $\bot$

else

    return $C(r')$

end if

4.2.4 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 4.2.5, we analyze the distributional accuracy of CorrSamp, showing that its distribution over outputs is close to the target distribution. In Section 4.2.6, we analyze the success probability of CorrSamp as a correlated sampler, showing that for two circuits C_1 and C_2, $\text{CorrSamp}(C_1, \nu; r) = \text{CorrSamp}(C_2, \nu; r)$ except with probability proportional to $d_{\text{TV}}(D_{C_1}, D_{C_2})$. In Section 4.2.7, we show that CorrSamp runs in time polynomial in the m, n, and $1/\nu$. Finally, in Section 4.2.8 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 4.13 (Ideal Conditions). *We collectively refer to the following as the* ideal conditions.

- *The inverter $\mathcal{I}_{\nu'}$ never fails: for all x and r on which CorrSamp invokes $\mathcal{I}_{\nu'}(x; r)$, $\mathcal{I}_{\nu'}(x; r) \neq \bot$.*

- *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 '$\bot$'. 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 $\hat{q}(y^*)$ is good:*

$$\hat{q}(y^*) \in (1 \pm O(\nu))\, p_D(y^*) 2^{\ell-k}$$

4.2.5 Analysis – Distributional Accuracy.

In this section, we analyze the distributional accuracy of CorrSamp. Denote by D_{CorrSamp} the distribution over outputs of $\text{CorrSamp}(C, \nu)$. We show that $d_{\text{TV}}(D_{\text{CorrSamp}}, D_C) \leqslant O(\nu)$, assuming the ideal conditions of Definition 4.13.

Lemma 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 $d_{TV}(D_C, D_{\texttt{CorrSamp}}) \leqslant O(\nu)$, assuming the ideal conditions of Definition 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 4.15. *Fix an $x \in \text{supp}(D_C)$. 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 $j - 1 \leqslant \ell \leqslant j + 2$, from the assumptions of Definition 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,

$$\mathbf{Pr}_\beta\left[\widehat{q}(x) \in (\tfrac{\beta}{2}2^{-k}, \beta 2^{-k}]\right] \in \mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell(1 \pm O(\nu)) \leqslant \beta]$$

$$\in \mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leqslant \beta] \pm O(\nu)p(x)2^\ell.$$

Recalling that ℓ is chosen uniformly at random from $\{0, \ldots, m\}$, we can then write the probability that `ElemFind` returns x as

$$\mathbf{Pr}_{H_1, u, \ell, \beta}[\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u) = x] \in \sum_{\ell \in [j-1, j+2]} \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leqslant \beta] \pm O(\nu)p_D(x)2^\ell}{(m+1)2^{\ell+k}}$$

$$\in \left(\sum_{\ell \in [j-1, j+2]} \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leqslant \beta]}{(m+1)2^{\ell+k}}\right) \pm \frac{O(\nu)p_D(x)}{(m+1)2^k}.$$

Observing that $\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leqslant \beta] = 0$ except for $\ell \in \{j, j+1\}$, we can simplify the series:

$$\sum_{\ell \in [j-1, j+2]} \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leqslant \beta]}{(m+1)2^{\ell+k}} = \sum_{\ell \in [j, j+1]} \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^\ell \leqslant \beta]}{(m+1)2^{\ell+k}}$$

$$= \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^j \leqslant \beta]}{(m+1)2^{j+k}} + \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < p_D(x)2^{j+1} \leqslant \beta]}{(m+1)2^{j+k+1}}$$

$$= \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < \gamma \leqslant \beta]}{(m+1)2^{j+k}} + \frac{\mathbf{Pr}_\beta[\tfrac{\beta}{2} < 2\gamma \leqslant \beta]}{(m+1)2^{j+k+1}}$$

$$= \frac{\mathbf{Pr}_\beta[\beta < 2\gamma]}{(m+1)2^{j+k}} + \frac{\mathbf{Pr}_\beta[\beta \geqslant 2\gamma]}{(m+1)2^{j+k+1}}$$

$$= \frac{2\gamma - 1}{(m+1)2^{j+k}} + \frac{2 - 2\gamma}{(m+1)2^{j+k+1}}$$

$$= \frac{2\gamma - 1 + 1 - \gamma}{(m+1)2^{j+k}}$$

$$= \frac{p_D(x)}{(m+1)2^k}.$$

Plugging back into the probability that `ElemFind` returns x, we have

$$\mathbf{Pr}_{H_1, u, \ell, \beta}[\texttt{ElemFind}_{C,\nu,\ell,\beta}(H_1, u) = x] \in \frac{p_D(x)(1 + O(\nu))}{(m+1)2^k}.$$

$\square$

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 4.14:

From Lemma 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 $d_{TV}(D_C, D_{\texttt{CorrSamp}}) \in O(\nu)$ as desired. $\square$

4.2.6 Analysis — Correlated Sampling.

Next, we show that `CorrSamp` satisfies the correlated sampling requirement of Definition 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 4.16 (Correlated Sampling of `CorrSamp`). *For all pairs of circuits $C_1, C_2 : \{0,1\}^m \to \{0,1\}^n$, assuming the ideal conditions hold for all rounds of* `CorrSamp`$(C_1, \nu; r)$ *and* `CorrSamp`$(C_2, \nu; r)$,

$$\mathbf{Pr}_r[\texttt{CorrSamp}(C_1, \nu; r) \neq \texttt{CorrSamp}(C_2, \nu; r)] \leqslant O(d_{TV}(D_{C_1}, D_{C_2}) + \nu).$$

Proof. Let E_0 denote the event that `CorrSamp`$(C_1, \nu; r) \neq$ `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 $\bot$ is always at most $O(\nu)$, we can condition on the fact that both `CorrSamp`$(C_2, \nu; r) \neq \bot$ and `CorrSamp`$(C_1, \nu; r) \neq \bot$ without loss of generality. Second, we can assume by symmetry that `CorrSamp`$(C_1, \nu; r)$ does not return before `CorrSamp`$(C_2, \nu; r)$ (else relabel C_1 as C_2).

With this in mind, let E_i denote the event that `CorrSamp`$(C_2, \nu; r)$ returns in round i. The event E_0 can then be bounded by

$$\mathbf{Pr}[E_0] \leqslant \sum_i \mathbf{Pr}[E_i]\mathbf{Pr}_{\beta,\ell}[\texttt{EF}_{C_1}^{(i)} \neq \texttt{EF}_{C_2}^{(i)} \mid E_i].$$

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 $\texttt{EF}_{C_1}^{(i)}$, $\hat{q}_1(x) \notin [\frac{\beta}{2}2^{-k}, \beta 2^{-k}]$. We will show conditioned on any value of x and ℓ, this occurs with probability at most:

$$\mathbf{Pr}_\beta[\texttt{EF}_{C_1}^{(i)} \neq x \mid E_i, \texttt{EF}_{C_2}^{(i)} = x, \ell] \leqslant O(2^\ell|p_2(x) - p_1(x)| + \nu). \tag{9}$$

In this case, we can bound E_0 by conditioning further on x and ℓ as:

$$\mathbf{Pr}_r[E_0] \leqslant \sum_{i\in[T_1]} \mathbf{Pr}_{\beta,\ell}[E_i]\mathbf{Pr}_{\beta,\ell}[\texttt{EF}_{C_1}^{(i)} \neq \texttt{EF}_{C_2}^{(i)} \mid E_i]$$

$$\leqslant \sum_{i\in[T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x\neq\bot} \mathbf{Pr}_{\beta,\ell}[\texttt{EF}_{C_1}^{(i)} \neq \texttt{EF}_{C_2}^{(i)} \mid E_i, \texttt{EF}_{C_2}^{(i)} = x]\mathbf{Pr}_{\beta,\ell}[\texttt{EF}_{C_2}^{(i)} = x|E_i]$$

$$\leqslant O(\nu) + \sum_{i\in[T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x\neq\bot} \mathbf{Pr}_{\beta,\ell}[\texttt{EF}_{C_1}^{(i)} \neq \texttt{EF}_{C_2}^{(i)} \mid E_i, \texttt{EF}_{C_2}^{(i)} = x]p_2(x)$$

$$\leqslant O(\nu) + \sum_{i\in[T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x\neq\bot} p_2(x) \sum_\ell \mathbf{Pr}[\ell \mid E_i, \texttt{EF}_{C_2}^{(i)} = x]\mathbf{Pr}_\beta[\texttt{EF}_{C_1}^{(i)} \neq \texttt{EF}_{C_2}^{(i)} \mid E_i, \texttt{EF}_{C_2}^{(i)} = x, \ell]$$

$$\leqslant O(\nu) + \sum_{i\in[T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x\neq\bot} p_2(x) \sum_\ell \mathbf{Pr}[\ell \mid E_i, \texttt{EF}_{C_2}^{(i)} = x]O(2^\ell|p_2(x) - p_1(x)|).$$

Under the ideal conditions, the posterior of ℓ is 0 unless $p_2(x) \in [2^{-\ell-2}, 2^{-\ell+2}]$, so altogether:

$$\mathbf{Pr}_r[E_0] \leqslant O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} p_2(x) \sum_\ell \mathbf{Pr}[\ell \mid E_i, \mathrm{EF}^{(i)}_{C_2} = x] O\left(\frac{1}{p_2(x)} |p_2(x) - p_1(x)|\right)$$

$$\leqslant O(\nu) + \sum_{i \in [T_1]} \mathbf{Pr}_{\beta,\ell}[E_i] \sum_{x \neq \perp} O(|p_2(x) - p_1(x)|)$$

$$\leqslant O(d_{TV}(D_{C_1}, D_{C_2}) + \nu)$$

as desired.

It is therefore left to prove Equation (9), which we analyze the probability by splitting into two cases based on $p_1(x)$:

1. $p_2(x)/4 \leqslant p_1(x) \leqslant 4p_2(x)$

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 4.13 and have conditioned on $\mathrm{ElemFind}_{C_2,\nu,\ell,\beta}(H_1, u; r) = x$ 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 $p_2(x)/4 \leqslant p_1(x) \leqslant 4p_2(x)$)

- $\hat{q}_2(x) \in (1 \pm O(\nu))p_2(x)2^{\ell-k}$

- $\hat{q}_1(x) \in (1 \pm O(\nu))p_1(x)2^{\ell-k}$.

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 $\mathrm{ElemFind}_{C_1,\nu,\ell,\beta}(H_1, u; r)$ outputs either x or $\perp$ in this round. Therefore, we can bound the probability $\mathrm{ElemFind}_{C_1,\nu,\ell,\beta}(H_1, u; r) \neq \mathrm{ElemFind}_{C_2,\nu,\ell,\beta}(H_1, u; r)$ by the probability that $\mathrm{ElemFind}_{C_1,\nu,\ell,\beta}(H_1, u; r)$ outputs $\perp$ conditioned on $\mathrm{ElemFind}_{C_2,\nu,\ell,\beta}(H_1, u; r)$ returning x. This occurs whenever β is chosen so that either

$$(1 + O(\nu))p_1(x)2^{\ell-k} < \frac{\beta 2^{-k}}{2} \leqslant p_2(x)2^{\ell-k}(1 + O(\nu))$$

or

$$p_2(x)2^{\ell-k}(1 - O(\nu)) \leqslant \beta 2^{-k} < p_1(x)2^{\ell-k}(1 + O(\nu)).$$

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

$$\beta \leqslant 2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu)(p_2(x) + p_1(x))2^\ell$$

$$\leqslant 2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu),$$

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 $\mathrm{ElemFind}_{C_2,\nu,\ell,\beta}(H_1, u; r) = x$ and $p_2(x) \in [2^{-\ell-4}, 2^{-\ell+4}]$, we have

$$\mathbf{Pr}_\beta[\mathrm{ElemFind}_{C_1,\nu,\ell\,\beta}(H_1, u; r) = \perp] \leqslant \mathbf{Pr}_\beta[\beta \leqslant 2^{\ell+1}|p_2(x_2) - p_1(x_2)| + O(\nu)]$$

$$\leqslant 2^{\ell+1}|p_2(x) - p_1(x)| + O(\nu).$$

45

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

$$\mathbf{Pr}_\beta[\texttt{ElemFind}_{C_1,\nu,\ell,\beta}(H_1, u; r) \neq x \mid \texttt{ElemFind}_{C_2,\nu,\ell,\beta}(H_1, u; r) = x] \leqslant 1$$
$$\leqslant O(2^\ell p_2(x_2))$$
$$\leqslant O(2^\ell |p_2(x_2) - p_1(x_2)|).$$

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. □

4.2.7 Analysis — Runtime

Lemma 4.17 (Runtime of CorrSamp). *Assuming inverter $\mathcal{I}$ 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 $2^k \in O(\mathrm{poly}(m, 1/\nu))$. Randomly sampling a pairwise-independent hash function from n bits to $\ell + k$ bits can be done in $\mathrm{poly}(m, n, 1/\nu)$ time.

Each round of CorrSamp contains a call to ElemFind. In ElemFind, there are $O(\mathrm{poly}(m, 1/\nu))$ rounds in which a hash function from m bits to $m - \ell + k$ bits is randomly sampled ($O(\mathrm{poly}(m, 1/\nu))$ time). Furthermore, each round contains a call to HashCheck, which runs in time $\mathrm{poly}(m, n, 1/\nu)$ by the assumption that inverter $\mathcal{I}_{\nu'}$ 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 $\mathrm{poly}(m, n, 1/\nu)$ random bits. Thus, r can efficiently be canonically proportioned for all uses of randomness in the algorithm. For more details, see Appendix C.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. □

4.2.8 Analysis – Removing Assumption of Ideal Conditions

Proposition 4.18. *The ideal conditions of Definition 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 4.19 (Inverter Never Fails). *Let $\mathcal{S}$ denote the set of strings $(x; r)$ on which CorrSamp invokes $\mathcal{I}_{\nu'}(x; r)$. The probability the inverter fails on $\mathcal{S}$ is negligible:*

$$\mathbf{Pr}[\exists (x; r) \in \mathcal{S} : \mathcal{I}_{\nu'}(x; r) = \bot] \leqslant O(\nu),$$

as long as $\nu' = \mathrm{poly}(\nu^{-1}, m, n)$ is sufficiently small.

Proof. Recall our inverter has the following guarantee

$$\mathbf{Pr}_{r' \sim \{0,1\}^m} [C(\mathcal{I}_{\nu'}(C, C(r')); r) = C(r')] \geqslant 1 - \nu',$$

where r stands for the internal randomness of $\mathcal{I}$ and ν' can be taken to be polynomially small in m, n and ν. It will be enough to take the failure rate $\nu' \leqslant O(\frac{\nu^2}{T_1 T_2}) = \text{poly}(\nu^{-1}, m, n)$. 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 $r' \in \{0, 1\}^m$:

$$\mathbf{Pr}_r \left[\mathbf{Pr}_{r'} \left[\mathcal{I}_{\nu'}(C, C(r')); r \right) \neq C(r') \right] \geqslant O\left(\frac{\nu}{T_1 T_2} \right) \right] \leqslant O(\nu).$$

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 $\mathcal{I}_{\nu'}(C, C(r')); r) \neq C(r')$. 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. $\qquad \square$

To prove the remaining conditions, it will be useful first to bound the number of collisions experienced by our hash functions.

Claim 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:*

$$\left| \{H_2, (r, r') \in C^{-1}(x) : H_2(r) = H_2(r')\} \right| \leqslant \delta_x T_2 |C^{-1}(x)|,$$

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 $2^{-\Omega(k)} \leqslant O(\nu)$ 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

$$T_2 \frac{|C^{-1}(x)|^2}{2^{2m-2\ell+2k}} \cdot \frac{T_1 2^m}{\nu} \leqslant T_2 |C^{-1}(x)| \cdot \frac{2^\ell p_D(x) mk \log(1/\nu)}{\nu 2^k}$$

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. $\qquad \square$

Lemma 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. $\qquad\square$

Lemma 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:*

$$\hat{q}(y^*) \in \left((1 \pm O(\nu))p_D(x)2^{\ell-k} \right)$$

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

$$\mathbf{Pr}_{v,H_2}[|\hat{q}(x) - p_D(x)2^{\ell-k}| \geqslant O(\nu)p_D(y^*)2^{\ell-k}] \leqslant e^{-\Omega(\nu^2 T_2)} \leqslant \frac{O(\nu)}{T_1}$$

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 4.20). In this case, since $p_D(y^*) \leqslant 2^{\ell+4}$, we have

$$C_{col} \leqslant O(\nu)|C^{-1}(y^*)|T_2$$

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. $\qquad\square$

Lemma 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 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 $\hat{q}(x) < 2^{-k-1} \leqslant (\beta/2)2^{-k}$. 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 $2^m e^{-\Omega(2^{-k}T_2)} \leqslant O(\nu)$.

Finally, consider x satisfying $p_D(x) \geqslant 2^{-\ell+4}$. In this case, we will aim show $\hat{q}(x) > \frac{2}{2^k} \geqslant \beta 2^{-k}$, so it is sufficient to consider the worst case when $p_D(x) = 2^{-\ell+4}$. By Claim 4.20, we can assume there are at most

$$C_{col} \leqslant \frac{1}{4} T_2 |C^{-1}(x)|$$

total collisions over the choices of H_2, thus as in Lemma 4.22, the collision-corrected Chernoff bound still promises $\hat{q}(x) > \frac{2}{2^k}$ with probability at least $e^{-\Omega(2^{-k}T_2)} \leqslant \frac{O(\nu)}{2^m T_1}$. 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 Lemma 4.22 gives us $\hat{q}(y^*) \in (1 \pm O(\nu))p_D(y^*)2^{\ell-k}$. Bounding $(1 \pm O(\nu)) \in (1/2, 2)$, we have

$$p_D(y^*) \in (\hat{q}(y^*)2^{k-\ell-1}, \hat{q}(y^*)2^{k-\ell+1}).$$

If `ElemFind` returned y^*, it must be the case that

$$\hat{q}(y^*) \in (\tfrac{\beta}{2}2^{-k}, \beta 2^{-k}] \in (2^{-k-1}, 2^{-k+1}],$$

and so

$$p_D(y^*) \in [2^{-\ell-2}, 2^{-\ell+2}],$$

conditioned on `ElemFind` returning y^*, as claimed.

$\square$

5 Separating Stability: Statistical Barriers

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 5.1. *Consider a product of d Rademacher distributions with expectations $p = (p_1, \ldots, p_d)$ respectively. A vector $v \in \mathbb{R}^d$ is said to be an α-accurate solution to the one-way marginals problem if $\|v - p\|_\infty \leqslant \alpha$.*

Definition 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 $\tilde{O}(\sqrt{d})$ 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 5.3. *Fix sufficiently large $d > 0$. For any 0.0001-replicable algorithm $\mathcal{A}$ that $(0.01, 0.001)$-accurately solves the one-way marginals problem over d coordinates with m samples,*

$$m = \tilde{\Omega}(d).$$

5.1.1 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 $\mathcal{A}$ 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

$$Z = \sum_{j \in [d]} \mathcal{A}^j(S) \sum_{i \in [m]} (S_i^j - p_j).$$

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, $\mathcal{A}^j(S)$ 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 5.4 (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*

$$\mathbb{E}_{r,q}[f(q) \sum_i (q_i - r) + 2|f(q) - \mu_q|] \geqslant \frac{1}{3}.$$

The lower bound for differential privacy proceeds by arguing that $\mathbb{E}[Z] = \mathbb{E}[\sum_j \mathcal{A}^j(S) \sum_i (S_i^j - p_j)]$ 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

$$\mathbb{E}[Z_{i*}] = \mathbb{E}[\sum_{j \in [d]} \mathcal{A}^j(S)(S_{i*}^j - p_j)]$$

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*}, $\mathcal{A}^j(S')$ is uncorrelated with $(S_{i*}^j - p_j)$, and hence the "1-neighboring" quantity

$$\mathbb{E}[Z_{i*}'] = \mathbb{E}[\sum_{j \in [d]} \mathcal{A}^j(S')(S_{i*}^j - p_j)]$$

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 $p = (p_1, \ldots, p_d)$), and consider the quantity $Z' = \sum_j \mathcal{A}^j(S') \sum_i (S_i^j - p_j)$. 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 $\mathcal{A}(S)$ and $\mathcal{A}(S')$ 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.

5.1.2 Formal argument

We start by proving our new lower bound for perfectly generalizing algorithms.

Theorem 5.5. *Fix any $m > 0$ and sufficiently large $d > 0$. Let $\mathcal{A}$ 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

$$Z = \sum_{j \in [d]} \mathcal{A}^j(S) \sum_{i \in [m]} (S_i^j - p_j), \quad Z' = \sum_{j \in [d]} \mathcal{A}^j(S) \sum_{i \in [m]} (S_i'^j - p_j).$$

As discussed above, we will argue that $\mathbb{E}[Z]$ is large (by the fingerprinting lemma):

$$\mathbb{E}[|Z|] \geqslant \frac{d}{10}, \tag{10}$$

that $\mathbb{E}[|Z'|]$ is small (since S' is independent of S):

$$\mathbb{E}[|Z'|] \leqslant 2\sqrt{dm}, \tag{11}$$

and finally that $\mathbb{E}[|Z|]$ and $\mathbb{E}[|Z'|]$ are close (by perfect generalization):

$$\mathbb{E}[|Z|] \leqslant e^2 \mathbb{E}[|Z'|] + \frac{8d}{m^2}. \tag{12}$$

---

[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.

51

Combining the inequalities we get

$$\frac{d}{10} \leqslant \mathbb{E}[|Z|] \leqslant e^2 \mathbb{E}\Big[|Z'|\Big] + \frac{8d}{m^2} \leqslant 2e^2\sqrt{dm} + \frac{8d}{m^2}$$

which implies $m \geqslant \Omega(d)$ as desired.

It remains to show Inequalities (10), (11), (12). We start with the first. Apply the fingerprinting lemma to the function corresponding to the jth coordinate of the output of $\mathcal{A}$, when run on the jth column of the input S with all other columns set to any fixed values. Then, we get that

$$\mathbb{E}_{p_j \sim [-1,1], S^j \sim Rad(p_j)^m}[\mathcal{A}^j(S;r)\sum_i (S_i^j - p_j) + 2|\mathcal{A}^j(S;r) - \mu_j|] \geqslant \frac{1}{3},$$

where μ_j is the empirical average of the jth 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

$$\mathbb{E}_{p,S \sim D_p^m, \mathcal{A}}[\sum_j \mathcal{A}^j(S) \sum_i (S_i^j - p_j) + 2|\mathcal{A}^j(S) - \mu_j|] = \mathbb{E}[Z] + 2\mathbb{E}[\sum_j |\mathcal{A}^j(S) - \mu_j|] \geqslant \frac{d}{3}$$

where we have used that S is drawn from a product distribution. It is therefore enough to argue that $\mathbb{E}[\sum_j |\mathcal{A}^j(S) - \mu_j|]$ is small.

Since $\mathcal{A}$ is $(0.01, 0.01)$-accurate, we can say that with probability at least 0.99, for all $j \in [d]$, $|\mathcal{A}^j(S) - p_j| \leqslant \frac{1}{100}$. Taking expectation, we get that $\mathbb{E}[\sum_{j \in [d]} |\mathcal{A}^j(S) - p_j|] \leqslant \frac{3d}{100}$. By a Chernoff bound, we can argue that since $m = \Omega(\log d)$, $\max_j |p_j - \mu_j| \leqslant 0.01$ with probability at least 0.99. By the triangle inequality, this gives us that $\mathbb{E}[\sum_{j \in [d]} |\mathcal{A}^j(S) - \mu_j|] \leqslant \frac{6d}{100}$. 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 (10).

Next, we show Equation (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 $M = \mathcal{A}^j(S)(S_i'^j - p_j)$ 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).

$$\begin{aligned}
\mathbb{E}[M] &= \mathbb{E}_p \mathbb{E}_{S,S',\mathcal{A}}[M \mid p] \\
&= \mathbb{E}_p \mathbb{E}_{S,S',\mathcal{A}}[\mathcal{A}^j(S)(S_i'^j - p_j) \mid p] \\
&= \mathbb{E}_p\Big[\mathbb{E}_{S,S',\mathcal{A}}[\mathcal{A}^j(S) \mid p]\,\mathbb{E}_{S,S',\mathcal{A}}[(S_i'^j - p_j) \mid p]\Big] = 0,
\end{aligned}$$

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

$$\begin{aligned}
\mathbb{E}_{p,S,S',\mathcal{A}}[MN] &= \mathbb{E}_p \mathbb{E}_{S,S',\mathcal{A}}[MN \mid p] \\
&= \mathbb{E}_p\Big[\mathbb{E}_{S,S',\mathcal{A}}[\mathcal{A}^j(S)^2 \mid p]\,\mathbb{E}_{S,S',\mathcal{A}}[(S_i'^j - p_j)(S_{i'}'^j - p_j) \mid p]\Big]
\end{aligned}$$

since S and S' are independent after conditioning. Since this is also the case for $(S_i'^j - p_j)$ and $(S_{i'}'^j - p_j)$ we have

$$\mathbb{E}[(S_i'^j - p_j)(S_{i'}'^j - p_j) \mid p] = \mathbb{E}[(S_i'^j - p_j) \mid p]\,\mathbb{E}[(S_{i'}'^j - p_j) \mid p] = 0.$$

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 $\mathcal{A}$ 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

$$\mathbb{E}^2[|Z'|] = \mathbb{E}^2_{p,S,S',\mathcal{A}}\left[\left|\sum_{j\in[d]}\mathcal{A}^j(S)\sum_i(S_i'^j - p_j)\right|\right] \qquad (S \text{ and } S' \text{ are i.i.d})$$

$$\leqslant \mathbb{E}_{p,S,S',\mathcal{A}}\left[\left(\sum_{j\in[d]}\mathcal{A}^j(S)\sum_i(S_i'^j - p_j)\right)^2\right] \qquad (\text{Jensen's Inequality})$$

$$= Var(Z'). \qquad (\mathbb{E}[Z'] = 0)$$

Since Z' is a sum of uncorrelated random variables and $\mathcal{A}^j(S)^2 \leqslant 1$, we then get

$$Var(Z') = \sum_j\sum_i Var(\mathcal{A}^j(S)(S_i'^j - p_j))$$

$$\leqslant \sum_j\sum_i \mathbb{E}[(S_i'^j - p_j)^2]$$

$$\leqslant 4dm$$

as desired.

It is left to show Equation (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 3.5 and Lemma 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 distributibly 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:

$$\mathbb{E}[|Z_p|] = \mathbb{E}_{S,A}\left[\left|\sum_{j\in[d]}\mathcal{A}^j(S)\sum_i(S_i^j - p_j)\right|\right] = \int_0^{2dm}\mathbf{Pr}[|Z_p| > z]dz$$

$$= \int_0^{2dm}\left[\mathbf{Pr}[|Z_p| > z \mid E]\mathbf{Pr}[E] + \mathbf{Pr}[|Z_p| > z \mid \overline{E}]\mathbf{Pr}[\overline{E}]\right]dz$$

$$\leqslant \int_0^{2dm}(e^2\mathbf{Pr}[|Z'_p| > z \mid E] + \frac{3}{m^3})\mathbf{Pr}[E]dz + \mathbf{Pr}[\overline{E}]dz$$

$$\leqslant \int_0^{2dm}e^2\mathbf{Pr}[|Z'_p| > z \mid E]\mathbf{Pr}[E]dz + \int_0^{2dm}\left[\frac{3}{m^3} + \mathbf{Pr}[\overline{E}]\right]dz$$

$$\leqslant \int_0^{2dm}e^2\mathbf{Pr}[|Z'_p| > z]dz + \int_0^{2dm}\left[\frac{3}{m^3} + \frac{1}{m^3}\right]dz$$

$$= e^2\mathbb{E}_{S,S',\mathcal{A}}[|Z'_p|] + \frac{8d}{m^2},$$

where the first inequality follows since $|Z_p|$ and $|Z'_p|$ are distributibly close conditioned on E, the second inequality is by the fact that $\mathbf{Pr}(E) \leqslant 1$, 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

$$\mathbb{E}[|Z|] \leqslant e^2 \mathbb{E}[|Z'|] + \frac{8d}{m^2}$$

as desired. $\qquad\square$

Now, we are ready to prove the lower bound for replicable algorithms.

Proof of Theorem 5.3. Let m be larger than an absolute constant K, without loss of generality.[12] By Claim 2.12, we have that $\mathcal{A}$ 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 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 $\mathcal{A}'$ 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 $\mathcal{A}'$ $k = 20\log(1/\gamma)$ 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 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 $m' = O(m\log^3(1/\gamma)\text{polylog}(1/\delta))$ 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, we get that $m' = \Omega(d)$, which gives us that $m = \tilde{\Omega}(d)$, completing the proof. $\qquad\square$

5.2 Quadratic separation: Agnostic Learning

In this section, we prove a lower bound for agnostic learning (See Section 2.4 for the definition of agnostic learning) under the constraint of replicability.

Theorem 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 $\mathcal{A}$ 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.

---

[12]We will show under this condition that $m = \tilde{\Omega}(d)$. Any algorithm taking fewer than K samples implies one taking between K and $\tilde{O}(d)$ samples, which would give a contradiction.

5.2.1 Sign-One-Way Marginals

Definition 5.7. *Consider a product of d Rademacher distributions with expectations $p = (p_1, \ldots, p_d)$. 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 $\frac{1}{d} \sum_{j=1}^{d} v_j p_j \geqslant \frac{1}{d} \sum_{j=1}^{d} |p_j| - \alpha$.*

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 5.8. *Let C be the class of products of d Rademacher random variables. We say that an algorithm $\mathcal{A} : \{\{-1, 1\}^d\}^m \to [-1, 1]^d$ (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.*

5.2.2 Solving Sign-One-Way Marginals using Agnostic Learning

Our reduction in Algorithm 9 shows how to use an agnostic learner $\mathcal{A}_{ag}$ for any class H of VC dimension d to construct an algorithm $\mathcal{A}$ 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 9: Algorithm $\mathcal{A}$ for sign-one-way marginals

1   **Input:** Sample access to a product distribution D over $\{-1, 1\}^d$, agnostic learner $\mathcal{A}_{ag}$ for hypothesis class H with VC dimension d

2   **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 $\mathcal{A}_{ag}$ 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 5.9. *Fix sufficiently large $d > 0$. Let $\mathcal{A}_{ag}$ be a $(0.01, 0.001)$-accurate, 0.0001-replicable agnostic learner for a hypothesis class H with VC dimension d. Then algorithm $\mathcal{A}$ 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 $d_{TV}(D', D'|_{\overline{E}}) \leqslant \mathbf{Pr}[E]$. 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. Now, by the data-processing inequality for total variation distance, we have that $d_{TV}(\mathcal{A}_{ag}(S_{ideal}), \mathcal{A}_{ag}(S_{ag,lab})) \leqslant 0.0001$. 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 $\mathcal{A}_{ag}$ produces outputs in this subset O on seeing $S_{ag,lab}$ is within 0.0001 of the probability that $\mathcal{A}_{ag}$ 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 $\mathcal{A}_{ag}$ and sample $S_{ag,lab}$, that

$$\mathbf{Pr}_{(x,y) \sim D_{ideal}}[f(x) \neq y] \leqslant \inf_{h \in H} \mathbf{Pr}_{(x,y) \sim D_{ideal}}[h(x) \neq y] + 0.01,$$

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 $\mathcal{A}_{ag}$ and sample $S_{ag,lab}$,

$$\mathbb{E}_{(x,y) \sim D_{ideal}}[\mathbb{1}[f(x) \neq y]] \leqslant \inf_{h \in H} \mathbb{E}_{(x,y) \sim D_{ideal}}[\mathbb{1}[h(x) \neq y]] + 0.01$$

$$\implies \mathbb{E}_{(x,y) \sim D_{ideal}}\left[\frac{1 - f(x)y}{2}\right] \leqslant \inf_{h \in H} \mathbb{E}_{(x,y) \sim D_{ideal}}\left[\frac{1 - h(x)y}{2}\right] + 0.01$$

$$\implies \mathbb{E}_{(x,y) \sim D_{ideal}}\left[\frac{1 - f(x)y}{2}\right] \leqslant \mathbb{E}_{(x,y) \sim D_{ideal}}\left[\frac{1 - \mathrm{sign}(p_x)y}{2}\right] + 0.01$$

$$\implies \mathbb{E}_{(x,y) \sim D_{ideal}}[f(x)y] \geqslant \mathbb{E}_{(x,y) \sim D_{ideal}}[\mathrm{sign}(p_x)y] - 0.02.$$

Now, Algorithm $\mathcal{A}$ calls the agnostic learner except with probability 0.0001. Unraveling the expectations, accounting for the fact that $\mathcal{A}$ 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 $\mathcal{A}$ as well as the randomness of S_{inp}, we get that with probability at least 0.998 over the randomness of the algorithm $\mathcal{A}$ and input sample, S_{inp}, that

$$\frac{1}{d}\sum_{j=1}^{d} f(x_j)p_j \geqslant \frac{1}{d}\sum_{j=1}^{d} \mathrm{sign}(p_j)p_j - 0.02,$$

proving that $\mathcal{A}$ is a $(0.02, 0.002)$-accurate algorithm for sign-one-way marginals over d coordinates.

Next, we prove that $\mathcal{A}$ inherits the replicability of the agnostic learner $\mathcal{A}_{ag}$. Consider two sets of independent samples $S_{inp,1}$ and $S_{inp,2}$. Consider any set of random coins r drawn for algorithm $\mathcal{A}$. 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 $\mathcal{A}(S_{inp,1}; r) = \mathcal{A}(S_{inp,2}; r)$ 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 $S_{ag,lab}, S'_{ag,lab} \sim D_{ideal}^m$ versus samples $S_{ag,lab}, S'_{ag,lab} \sim D_{ag}^m$. This allows us to conclude that

$$
\begin{aligned}
\mathbf{Pr}_{S_{inp,1},S_{inp,2}\sim D^m, r}&[\mathcal{A}(S_{inp,1}; r) = \mathcal{A}(S_{inp,2}; r)] \\
&\geqslant \mathbf{Pr}_{S_{inp,1},S_{inp,2}\sim D^m, r}[\mathcal{A}(S_{inp,1}; r) = \mathcal{A}(S_{inp,2}; r) \mid \overline{E}] \\
&\geqslant \mathbf{Pr}_{S_{ag,lab},S'_{ag,lab}\sim D_{ag}, r_{ag}}[\mathcal{A}_{ag}(S_{ag,lab}; r_{ag}) = \mathcal{A}_{ag}(S'_{ag,lab}; r_{ag})] \\
&\geqslant \mathbf{Pr}_{S_{ag,lab},S'_{ag,lab}\sim D_{ideal}^m, r_{ag}}[\mathcal{A}(S_{ag,lab}; r_{ag}) = \mathcal{A}(S'_{ag,lab}; r_{ag})] - 0.0002 \\
&\geqslant 0.9999 - 0.0002 = 0.9997.
\end{aligned}
$$

Hence, we have proved that $\mathcal{A}$ is 0.0005-replicable. $\qquad\square$

5.2.3   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), except that the notion of accuracy is different, and so we use a different version of the fingerprinting lemma, given below.

Lemma 5.10 ([BSU19], Lemma A.1 and A.2)**.** *Let f be a function from $\{-1, 1\}^m \to \mathcal{R}$. 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 5.11. *Fix any $m > 0$, sufficiently large $d > 0$. Let $\mathcal{A}$ 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 $\mathcal{A}$. Following the framework in Theorem 5.5, we will first argue the expected correlation of our algorithm and its input is large:

$$\sum_{j=1}^{d} \mathbb{E}_{p,\mathcal{A},S \sim D_p^m}[\mathcal{A}^j(S) \sum_{i=1}^{m}(S_i^j - p_j)] \geq 0.35d.$$

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 $p = (p_1, \ldots, p_d)$, we have that we have that with probability at least 0.95 over the randomness of the algorithm $\mathcal{A}$ and its input sample,

$$\frac{1}{d}\sum_{j=1}^{d} v_j p_j \geq \frac{1}{d}\sum_{j=1}^{d} \operatorname{sign}(p_j)p_j - 0.05,$$

where v is the vector output by the algorithm.

Now, taking expectation over the randomness of the algorithm $\mathcal{A}$ and the input sample S, we get that

$$\mathbb{E}_{\mathcal{A},S \sim D_p^m}\left[\frac{1}{d}\sum_{j=1}^{d}\mathcal{A}^j(S)p_j\right] \geq 0.95\left[\frac{1}{d}\sum_{j=1}^{d}|p_j| - 0.05\right] - 0.05,$$

which implies that

$$\mathbb{E}_{\mathcal{A},S \sim D_p^m}\left[\frac{1}{d}\sum_{j=1}^{d}\mathcal{A}^j(S)p_j\right] \geq \frac{1}{d}\sum_{j=1}^{d}|p_j| - 0.15,$$

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

$$\mathbb{E}_{p,\mathcal{A},S \sim D_p^m}\left[\sum_{j=1}^{d}\mathcal{A}^j(S)p_j\right] \geq \mathbb{E}_p\left[\sum_{j=1}^{d}|p_j|\right] - 0.15d \tag{13}$$

$$\implies \sum_{j=1}^{d}\mathbb{E}_{p,\mathcal{A},S \sim D_p^m}\left[\mathcal{A}^j(S)p_j\right] \geq \sum_{j=1}^{d}\mathbb{E}_{p_j \sim [-1,1]}\left[|p_j|\right] - 0.15d = 0.35d, \tag{14}$$

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 $\mathcal{A}$, and for any values of columns of S that are not the jth column, we get from Lemma 5.10 applied to the function f corresponding to the algorithm $\mathcal{A}$ on the complete dataset S with internal randomness r, that

$$\mathbb{E}_{p_j,S^j \sim Rad(p_j)^m}\left[\mathcal{A}^j(S;r)p_j\right] = \mathbb{E}_{p_j,S^j \sim Rad(p_j)^m}[\mathcal{A}^j(S;r)\sum_{i=1}^{m}(S_i^j - p_j)].$$

---

[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.

Now, since this holds for any fixed values of internal randomness r and for any values of columns of S that are not the jth 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

$$\mathbb{E}_{p,\mathcal{A},S\sim D_p^m}\left[\mathcal{A}^j(S)p_j\right] = \mathbb{E}_{p,\mathcal{A},S\sim D_p^m}[\mathcal{A}^j(S)\sum_{i=1}^m(S_i^j - p_j)],$$

where we have used that D_p is a product distribution. Now, summing over all coordinates $j \in [d]$, we get that

$$\sum_{j=1}^d \mathbb{E}_{p,\mathcal{A},S\sim D_p^m}[\mathcal{A}^j(S)\sum_{i=1}^m(S_i^j - p_j)] = \sum_{j=1}^d \mathbb{E}_{p,\mathcal{A},S\sim D_p^m}\left[\mathcal{A}^j(S)p_j\right] \geqslant 0.35d, \qquad (15)$$

where we have used Equation (14).

Now, we can proceed exactly as in the proof of Theorem 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 $Z = \sum_{j\in[d]} \mathcal{A}^j(S) \sum_i (S_i^j - p_j)$ and $Z' = \sum_{j\in[d]} \mathcal{A}^j(S') \sum_i (S_i^j - p_j)$.

First, note that Z' is a sum of uncorrelated random variables with mean 0 (see the proof of Theorem 5.5 for a proof of this).

We can then prove (as in the proof of Theorem 5.5) that

$$\mathbb{E}[|Z'|] = \mathbb{E}_{p,\mathcal{A},S,S'\sim D_p^m}\left[\left|\sum_{j\in[d]} \mathcal{A}^j(S')\sum_i(S_i^j - p_j)\right|\right] \leqslant 2\sqrt{dm},$$

Then, we can invoke the perfect generalization guarantee to prove (as in the proof of Theorem 5.5) that

$$\mathbb{E}[|Z|] \leqslant e^2\mathbb{E}[|Z'|] + \frac{8d}{m^2},$$

Hence, combining the inequality above with equation 15, we get that

$$0.35d \leqslant \mathbb{E}[|Z|] \leqslant e^2\mathbb{E}[|Z'|] + \frac{8d}{m^2} \leqslant 2e^2\sqrt{dm} + \frac{8d}{m^2}.$$

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 5.12. *Fix sufficiently large $d > 0$. For any 0.0005-replicable algorithm $\mathcal{A}$ 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 2.12, we have that $\mathcal{A}$ is $(0.01, 0.05)$-replicable and $(0.02, 0.002)$-accurate when given m samples. Consider any sufficiently small

---

[14]We will show under this condition that $m = \tilde{\Omega}(d)$. Any algorithm on $\leqslant K$ samples implies one taking between K and $\tilde{O}(d)$ samples, which would give a contradiction.

$\gamma > 0$. Next, applying Theorem 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 $\mathcal{A}'$ 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 $\mathcal{A}$ 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 $\mathcal{A}$ on the k runs be $v^1, \ldots, v^k$. We are interested in the quantity $\frac{1}{d} \sum_{j=1}^{d} \left[|p_j| - \frac{p_j}{k} \sum_{i=1}^{k} v_j^i \right]$. First, we analyze the expectation of this quantity.

$$
\mathbb{E}\left[\frac{1}{d} \sum_{j=1}^{d} \left[|p_j| - \frac{p_j}{k} \sum_{i=1}^{k} v_j^i \right] \right] = \mathbb{E}\left[\frac{1}{d} \sum_{j=1}^{d} p_j \left[\mathrm{sign}(p_j) - \frac{1}{k} \sum_{i=1}^{k} v_j^i \right] \right]
$$
$$
= \mathbb{E}\left[\frac{1}{d} \sum_{j=1}^{d} \frac{1}{k} \sum_{i=1}^{k} p_j \left[\mathrm{sign}(p_j) - v_j^i \right] \right]
$$
$$
= \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}\left[\frac{1}{d} \sum_{j=1}^{d} p_j \left[\mathrm{sign}(p_j) - v_j^i \right] \right] \leqslant 0.03
$$

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 $\frac{1}{k} \sum_{i=1}^{k} \left(\frac{1}{d} \sum_{j=1}^{d} p_j \left[\mathrm{sign}(p_j) - v_j^i \right] \right)$ is a sum of k independent random variables (since the ith 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 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 5.11, we get that $m' = \Omega(d)$, which gives us that $m = \tilde{\Omega}(d)$, completing the proof. $\qquad \square$

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 5.6. 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 5.9, 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 5.12. $\qquad \square$

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.

5.3 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 6.13).

Theorem 5.13 (Finite Classes are Replicably Learnable). *Any class H is replicably Agnostic learnable with sample complexity:*

$$m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log^2 |H| + \log \frac{1}{\rho\beta}}{\alpha^2 \rho^2} \log^3 \frac{1}{\rho}\right).$$

In the realizable setting, the α-dependence can be improved to linear:

$$m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log^2 |H| + \log \frac{1}{\rho\beta}}{\alpha \rho^2} \log^3 \frac{1}{\rho}\right).$$

Theorem 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.

5.3.1 Algorithm

At its core, the algorithm achieving Theorem 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 $err_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \mathbf{1}[f(x) \neq y]$ at most v. Implementing this strategy requires a bit more

effort, and is achieved formally by the following algorithm.

Algorithm 10: (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 $m = m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log^2 |H| \log \frac{1}{\rho} + \rho^2 \log \frac{1}{\beta}}{\alpha^2 \rho^4}\right)$

- Replicability bucket size $\tau \leqslant O(\frac{\alpha\rho}{\ln |H|})$

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 12)

3. Select random threshold $v \leftarrow_r \{v_{\text{init}} + \frac{3}{2}\tau, v_{\text{init}} + \frac{5}{2}\tau, \dots, v_{\text{init}} + \alpha/4 - \tau/2\}$

4. Randomly order all $f \in H$

return Output the first hypothesis f in the order s.t. $err_S(f) \leqslant v$.

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 Appendix A for completeness.

We note that while Algorithm `rFiniteLearner` is a replicable agnostic PAC learner, it is not quite sufficient to prove Theorem 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.

5.3.2 Analysis

We'll start by proving the following weaker bound for our intermediate learner.

Theorem 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:*

$$m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log^2 |H| \log(\frac{1}{\rho}) + \rho^2 \log \frac{1}{\beta}}{\alpha^2 \rho^4}\right).$$

In the realizable setting, the (α, β)-dependence can be improved to:

$$m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log^2 |H| \log(\frac{1}{\rho}) + \rho^4 \log \frac{1}{\beta}}{\alpha \rho^4}\right).$$

The main challenge in Theorem 5.14 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 5.15. *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

$$H_1^{(i)} = \{h \in H : err_{S_1}(h) \leqslant v_i\}, \quad H_2^{(i)} = \{h \in H : err_{S_2}(h) \leqslant v_i\}$$

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 5.16. *Let $v_{init} \in [OPT, OPT + \alpha/2]$ and $\tau \leqslant O\left(\frac{\alpha \rho^2}{\log |H|}\right)$ 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

$$H_1^{(i)} = \{h \in H : err_{S_1}(h) \leqslant v_i\}, \quad H_2^{(i)} = \{h \in H : err_{S_2}(h) \leqslant v_i\}$$

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:

$$\frac{|H_1^{(i)} \Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|} \leqslant \rho/4.$$

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:

$$\exists j \geqslant 1 : |I_{i+j}| \geqslant e^j |I_{[i-1]}|.$$

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

$$\mathop{\mathbf{Pr}}_{S_1, S_2} \left[\frac{|H_1^{(i)} \Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|} \leqslant \frac{\rho}{4} \right] \geqslant 1 - \frac{\rho}{8}.$$

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

$$\mathbf{Pr}[err_S(h) \leqslant v_i] \leqslant e^{-\Omega(j^2 \tau^2 |S|)} \tag{16}$$

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

$$\mathbb{E}[x_i] \leqslant |I_{[i-1]}| \sum_{j>0} e^{-\Omega(j^2 \tau^2 |S| - j)} \leqslant \frac{\rho^2}{2000} |I_{[i-1]}|$$

for our choice of $|S|$. Markov's inequality then promises

$$\mathbf{Pr}\left[x_i \geqslant \frac{\rho}{30} |I_{[i-1]}|\right] \leqslant \frac{\rho}{64}.$$

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

$$|H_1^{(i)} \Delta H_2^{(i)}| \leqslant \frac{\rho}{5} |I_{[i-1]}|.$$

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:

$$|H_1^{(i)} \cup H_2^{(i)}| \geqslant \left(1 - \frac{\rho}{15}\right) |I_{[i-1]}|$$

with probability at least $1 - \frac{\rho}{64}$. Thus altogether a union bound gives

$$\mathbf{Pr}_{S_1, S_2}\left[\frac{|H_1^{(i)} \Delta H_2^{(i)}|}{|H_1^{(i)} \cup H_2^{(i)}|} \leqslant \frac{\rho}{4}\right] \geqslant 1 - \frac{\rho}{8}$$

as desired.

Finally, we need to show that almost all thresholds are good. To see this, first observe that since $v_{\text{init}} \geqslant OPT$, $|I_{[i]}| > 0$ for all $i \geqslant 0$. To count the number of bad thresholds, let $i_1 \geqslant 1$ 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 $|I_0| \geqslant 1$ and the total number of hypotheses is $|H|$ by definition, we may therefore write:

$$|H| \geqslant |I_{[i_\ell + t_\ell]}| \geqslant \left(1 + \frac{\rho}{30}\right)^{\sum_{j=1}^{\ell} t_j}$$

64

and thus that the total number of bad intervals is at most

$$\sum_{j=1}^{\ell} t_j \leqslant O\left(\frac{\log(|H|)}{\rho}\right).$$

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. $\qquad\square$

To complete the argument, it is enough to show we can find a good starting point v_{init}.

Lemma 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:*

$$\mathbf{Pr}_{r,S}\big[\mathcal{A}(S) \in [OPT, OPT + \alpha/2]\big] \geqslant 1 - \beta$$

Proving this Lemma largely follows from prior techniques but is a bit tedious, so we leave the proof for Appendix A. With these tools in hand, we are finally ready to prove Theorem 5.13.

Proof of Theorem 5.14. We start by showing `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 $H^{(i)} = \{h : R_{\text{emp}}(h, S) \leqslant v_i\}$. By Lemma 5.16 and Observation 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 (16) to

$$\mathbf{Pr}\big[err_S(h) \leqslant v_i\big] \leqslant e^{-\Omega(\frac{j^2\tau|S|}{(i+j)})} \leqslant e^{-\Omega(\frac{j^2\tau^2|S|}{\alpha})}.$$

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. $\qquad\square$

Finally, we amplify the above to prove Theorem 5.13.

Proof of Theorem 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 $\mathcal{D}_i$ 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 $S_1, \ldots, S_t$ 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 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.

$\square$

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.

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

---

[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.

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 3.2 and 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 6.1. *There are universal constants $c, K > 0$ such that the following holds. Let $\mathcal{T}$ be a statistical task with a finite output space. Given a $(0.1, \frac{c}{n^3})$-item level differentially private algorithm $\mathcal{A}$ that solves $\mathcal{T}$ using n samples and with failure probability β, for every $1 \geqslant \varepsilon, \delta > 0$, there exists an (ε, δ)-user level differentially private algorithm $\mathcal{A}_u$ that solves $\mathcal{T}$ 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 $\mathcal{A}$ by subsampling. By Lemma 2.9, the algorithm $\mathcal{A}'$ that, given $m = Kn^2$ samples, subsamples n items without replacement and runs $\mathcal{A}'$ 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 $\mathcal{A}'$ is identically distributed to that of $\mathcal{A}$, so $\mathcal{A}'$ also solves $\mathcal{T}$ with m examples and failure probability β.

For a sufficiently large constant K and sufficiently small constant c, Corollary 3.18 then implies that $\mathcal{A}'$ gives rise to a c-replicable algorithm solving $\mathcal{T}$ with m examples and failure probability β.

Now, consider Algorithm 1 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 $\mathcal{A}'$ 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 1). Let's call this algorithm $\mathcal{A}_u$.

The accuracy guarantees proved for Algorithm 1 give us that the failure probability of this Algorithm $\mathcal{A}_u$ 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 2.8), we have that $\mathcal{A}_u$ 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. $\square$

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 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 $\mathcal{C}$ 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 $\mathcal{C}$ 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 $\mathcal{C}$ using $\mathrm{poly}(d, 1/\varepsilon, \log(1/\delta))$ samples and a private query release algorithm for $\mathcal{C}$ 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 6.2. *There is a universal constant $c > 0$ such that the following holds. Let $\mathcal{T}$ 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 $\mathcal{T}$ using n samples and with failure probability β. Then for every $\varepsilon, \delta > 0$, there exists an (ε, δ)-differentially private algorithm solving $\mathcal{T}$ 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 $\mathcal{T}$ with n samples and failure probability $O(1/\log(1/\beta))$. By Lemma 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 $\mathcal{T}$ with m samples and failure probability β.

For a sufficiently large constant K and sufficiently small constant c, Corollary 3.18 implies that A' gives rise to a c-replicable algorithm solving $\mathcal{T}$ with m samples and failure probability β. Applying Theorem 3.1 thus results in an (ε, δ)-differentially private algorithm solving $\mathcal{T}$ with

$$
m \cdot O\left(\frac{\log(1/\delta)\log(1/\beta)}{\varepsilon} + \log^2(1/\beta)\right)
$$

samples and failure probability $O(\beta \log 1/\beta)$. $\square$

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 6.3. *There is a universal constant $c > 0$ such that the following holds. Let $\mathcal{T}$ 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 $\mathcal{T}$ using n samples and with failure probability β. Then for every $\varepsilon, \delta > 0$, there exists an $(\delta, \varepsilon, \delta)$- perfectly generalizing algorithm solving $\mathcal{T}$ using

$$O\left(n \cdot \frac{\log(1/\varepsilon)\operatorname{poly}\log(1/\delta)}{\varepsilon^2}\right)$$

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 $\mathcal{T}$ with n samples and failure probability β. By Lemma 3.17, there is a (0.0001)-replicable algorithm solving $\mathcal{T}$ with the same failure probability and the same number of samples. By Claim 2.12 and Theorem 3.19, there exists a $(\delta^2/4, \varepsilon, \delta^2/4)$-sample perfectly generalizing algorithm solving $\mathcal{T}$ using $O(n \cdot \frac{\log(1/\varepsilon)\operatorname{poly}\log(1/\delta)}{\varepsilon^2})$ samples with failure probability $O(\delta) + \sqrt{\beta}\log(1/\delta)$. Lemma 3.6 converting sample perfect generalization to perfect generalization gives the result. □

6.3 A Realizable-to-Agnostic 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 highlight the differences from the standard model below.

Definition 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 $\mathcal{X}$. We say an algorithm $\mathcal{A}$ 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 $D_{\mathcal{X}} \in \mathscr{D}$, $\mathcal{A}$ 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,

$$\mathtt{err}_D(h) \leqslant \inf_{f \in H} \mathtt{err}_D(f) + \alpha,$$

*where $\mathtt{err}_D(h) = \mathbf{Pr}_{(x,y) \in D}[h(x) \neq y]$. When the adversary is additionally restricted to choosing D s.t. $\inf_{f \in H} \mathtt{err}_D(f) = 0$, 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 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*

$$m(\alpha, \beta) \leqslant O\left(\frac{n^2\alpha^2 + \log^3(\Pi_H(cn^2))}{\alpha^2} \log \beta^{-1} \log\left(\frac{n \log \beta^{-1}}{\alpha}\right)\right)$$

samples for some universal constant $c > 0$.

In the statement of Theorem 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 $\Pi_H(cn^2) \leqslant 2^{cn^2}$, this means agnostic learning experiences at most a polynomial blowup over the realizable setting:

Corollary 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*

$$m(\alpha, \beta) \leqslant \tilde{O}\left(\frac{n^6 \log \beta^{-1}}{\alpha^2}\right)$$

samples.

At a high level, the proof of this result is (comparatively) simple. Given a realizable private learner $\mathcal{A}$, we will transform $\mathcal{A}$ 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 12):

1. **Sample:** Draw an unlabeled sample $S \sim D^n$, and random string r

2. **Generate Candidates:** Run $\mathcal{A}$ 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.

6.3.1 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 5.13), 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 $\mathcal{D}$ a distribution over subsets of H. We call $h \in H$ an η-heavy-hitter of $\mathcal{D}$ if

$$\mathbf{Pr}_{S \sim \mathcal{D}}[h \in S] \geq \eta.$$

We prove it is possible to replicably output a heavy hitter with complexity scaling that is only polylogarithmic in the largest set supported by $\mathcal{D}$.

Theorem 6.7. *For any finite set Ω, $\rho, \eta, \beta > 0$, and distribution $\mathcal{D}$ over subsets of Ω with an η-heavy-hitter, there exists a ρ-replicable algorithm $\mathcal{A}$ with the following guarantees:*

1. *$\mathcal{A}$ outputs a $\frac{\eta}{2}$-heavy-hitter with probability at least $1 - \beta$*

2. *$\mathcal{A}$ uses at most $O\left(\dfrac{\log^2 \frac{|\mathcal{D}| \log \frac{1}{\rho\beta}}{\eta} \log \frac{1}{\rho} + \log \frac{1}{\rho\beta}}{\eta^2 \rho^2} \log^3 \frac{1}{\rho} \right)$ samples from $\mathcal{D}$,*

where $|\mathcal{D}|$ is the maximum size subset supported by $\mathcal{D}$.

We note that it is easy to modify this result to remove the assumption that $\mathcal{D}$ has a heavy hitter (the algorithm instead outputs '$\perp$' in this case, or can test for the heaviest element), but the 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 11: List Heavy Hitters

Result: Replicably outputs a heavy hitter
Input: Distribution $\mathcal{D}$ over subsets of universe Ω (Sample Access)
Parameters:

- Replicability, confidence, and heaviness $\rho, \beta, \eta > 0$

- Sample sizes $t_1 = O\left(\dfrac{\log(\frac{|\mathcal{D}|}{\rho\beta})}{\eta} \right)$, $t_2 = O\left(\dfrac{\log^2 \frac{|\mathcal{D}| \log \frac{1}{\rho\beta}}{\eta} \log \frac{1}{\rho} + \log \frac{1}{\beta}}{\eta^2 \rho^4} \right)$

- Threshold accuracy $\tau \leq O(\frac{\rho\eta}{\log |\mathcal{D}|})$

Algorithm:

1. Sample t_1 subsets $C \sim \mathcal{A}$, and call their union T.

2. Sample an additional t_2 subsets $C \sim \mathcal{A}$, 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 $\hat{p}_t \geq v$

---

[16]We note that a similar result also appears implicitly in [GKM21, Theorem 20], albeit with worse sample complexity.

It is not hard to see this algorithm succeeds via the same analysis as for Theorem 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 $\mathcal{D}$.

Similar to the proof of Theorem 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

$$H_1^{(i)} = \{t \in T : \hat{p}(t, S_1) \leqslant v_i\}, \quad H_2^{(i)} = \{t \in T : \hat{p}(t, S_2) \leqslant v_i\}.$$

In the proof of Theorem 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 5.16, 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:

$$\mathbf{Pr}[\hat{p}(t, S_j) \leqslant v_i] \leqslant e^{-\Omega(j^2\tau^2|S_j|)},$$

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 5.13. $\qquad\square$

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 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:*

$$n(\rho, \alpha, \beta) \leqslant \tilde{O}\left(\frac{d^{12} \log^3(1/\beta)}{\alpha^2 \rho^2} \right)$$

We give the proof in Appendix B.

---

[17]Recall the problem can be solved replicably with $\mathrm{poly}(|\mathcal{D}|)$ dependence by [ILPS22] even in the infinite setting.

6.3.2 Replicable Realizable-to-Agnostic 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 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*

$$m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log^2(\Pi_H(n') \log \frac{1}{\rho\beta}) \log \frac{1}{\rho} + \log \frac{1}{\rho\beta}}{\alpha^2 \rho^2} \left(\alpha^2 n' + \log \frac{\Pi_H(n')}{\beta}\right) \log^3 \frac{1}{\rho}\right)$$

samples, where $\Pi_H(n)$ is the growth function of (X, H).

Since $\Pi_H(n) \leqslant 2^n$, 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 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 12: List Distribution Generator

Result: Outputs a list of good hypotheses
Input: Replicable learner $\mathcal{L}$ 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 $\mathcal{L}$ across all strings in r on all possible labelings of S_U to receive:

$$C_r(S_U) \coloneqq \{\mathcal{L}((S_U, h(S_U)); r_i) : h \in H, r_i \in \{r\}\}$$

3. Prune sub-optimal hypotheses from $C_r(S_U)$:

$$C_r^\alpha(S_U, S_L) \coloneqq \left\{h \in C_r(S_U) : R_{S_L}(h) \leqslant \min_{h' \in C_r(S_U)} (R_{S_L}(h')) + \alpha/2\right\}$$

return $C_r^\alpha(S_U, S_L)$

Algorithm 12 generates a sample from a distribution over families of hypotheses with near-optimal error. The accuracy of $\mathcal{A}$ 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 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:
$$\mathbf{Pr}[\forall h \in C_r^\alpha(S_U, S_L) : R(h) \leqslant OPT + \alpha] \geqslant 1 - \beta/2$$

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:
$$\mathbf{Pr}_{S_U}[\mathcal{L}((S_U, h_{OPT}(S_U)), r) = h_r] \geq 1/2.$$

Furthermore, since $\mathcal{L}$ 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 $\mathcal{L}((S_U, h_{OPT}(S_U)), r) \in C_r(S_U)$ 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 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. $\qquad\square$

6.3.3 Private Realizable-to-Agnostic Reduction

We are finally ready to prove our agnostic-to-realizable reduction for private learning. We restate the Theorem for ease of reading.

Theorem 6.10 (Theorem 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 $\mathcal{A}$ on n samples. We will convert $\mathcal{A}$ 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 $\mathcal{A}$ to a 0.01-replicable realizable learner $\mathcal{R}$

3. Convert $\mathcal{R}$ into an agnostic learner $\mathcal{R}_{agn}$

4. Convert $\mathcal{R}_{agn}$ back into an agnostic private learner $\mathcal{A}_{agn}$

5. Privately amplify correctness of $\mathcal{A}_{agn}$

Let's formalize this procedure. In the first step, we simply wish to convert $\mathcal{A}$ into a $(cm^{-1/2}, \mathrm{poly}(m^{-1}))$-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 2.9): draw $m = O(n^2)$ examples, construct a subset S by selecting m elements uniformly at random, and return $\mathcal{A}(S)$. For an appropriate choice of constants this is $(cm^{-1/2}, \mathrm{poly}(m^{-1}))$-DP. Accuracy is maintained since S is equidistributed with a standard size m sample.

Now that we have our $(cm^{-1/2}, \mathrm{poly}(m^{-1}))$-DP, $(\alpha, .01)$-accurate PAC learner, we invoke Corollary 3.18 (applying correlated sampling) to build the .01-replicable learner $\mathcal{R}$ 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

$$m' \leqslant O\left(n^2 + \frac{\log^3(\Pi_H(c'n^2))}{\alpha^2}\right)$$

samples for some constant $c' > 0$. Finally, we move back to the private regime via Theorem 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

$$m'' \leqslant O\left(\frac{m' \log \delta^{-1} \log \beta^{-1}}{\varepsilon}\right)$$

samples. To give the form of the result in the theorem statement, it is enough to choose sample size t satisfying the recurrence $t \geqslant \Omega(m' \log t \log \beta^{-1})$. Selecting $t = c_2 m' \log \beta^{-1} \log(m'\beta^{-1})$ for large enough $c_2 > 0$ then completes the proof. □

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 3.2 and 3.3).

6.4.1 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 $d \geqslant 0$. We apply our framework to the hypothesis class $Thresh_d$ consisting of thresholds over the domain $\{0, 1, \dots, d\}$. A threshold function f_z parameter-

ized by integer $0 \leqslant z \leqslant d$, is defined as follows.

$$f_z(x) = \begin{cases} 1 & \text{if } x > z \\ 0 & \text{if } x \leqslant z \end{cases} \tag{17}$$

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 6.11. *An algorithm solves the interior point problem over a totally ordered domain $\mathcal{X}$ with error probability β, if for all datasets $S \in \mathcal{X}^m$, if*

$$\mathbf{Pr}[\min_i S_i \leqslant \mathcal{A}(S) \leqslant \max_i S_i] \geqslant 1 - \beta.$$

Now we are ready to apply our reduction to obtain the improved sample complexity.

Theorem 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 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 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

$$m = \tilde{O}\left(\frac{\log^4(1/\rho)}{\rho^2\alpha^2}(\log^* d)^3 \log^2(1/\beta) \right).$$

$\square$

---

[18]We note that a similar approach (taking a differentially private algorithm for learning distributions under Kolomogorov 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].

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 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 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 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 5.3 (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 $O_{\rho,\alpha,\beta}(\text{PRDim}(H)^2)$ for finite hypothesis classes with finite probabilistic representation dimension (denoted by PRDim), and ρ-replicable (α, β)-PAC learners with sample complexity scaling as $\tilde{O}_{\rho,\alpha,\beta}(\text{LDim}(H)^{12})$ 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 B.1.

6.4.2 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, $d_{TV}(D, D') \leqslant \alpha$.

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 $\delta \leqslant \varepsilon$, 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 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 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 $\mathcal{A}$ 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 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

$$\mathbb{E}[\|\hat{d} - p\|_1] \leqslant \mathbb{E}[\|\hat{d} - \hat{v}\|_1] + \mathbb{E}[\|\hat{v} - p\|_1].$$

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 $\mathbb{E}[\|\hat{d} - \hat{v}\|_1] \leqslant \alpha$. Hence, we can write that

$$\mathbb{E}[\|\hat{d} - p\|_1] \leqslant \alpha + \mathbb{E}[\|v - p\|_1],$$

where in the second term on the right hand side, we have used again that $\hat{v}$ and v are identically distributed.

Finally, since $\mathbb{E}[\|v - p\|_1] \leqslant \mathbb{E}[\|v - d\|_1] + \mathbb{E}[\|d - p\|_1]$ (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 $\mathcal{A}$), we get that

$$\mathbb{E}[\|\hat{d} - p\|_1] \leqslant \alpha,$$

as required.

Replicability of this transformation follows from the fact that the transformation prior to projection onto the simplex is replicable (by Corollary 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 $\mathcal{A}$ 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. $\qquad\square$

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

$$\|\hat{\mu} - \mu\|_\Sigma = \|\Sigma^{-1/2}(\hat{\mu} - \mu)\|_2 \leqslant \alpha.$$

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.

---

[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

Theorem 6.15. *Fix $R > 0$, $0 < k < \ell$, and sufficiently small $\rho > 0$. Fix a distribution $D = N(\mu, \Sigma)$, where $\|\mu\|_\infty \leqslant R$, 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*

$$m = \tilde{O}\left(\frac{(d\log(1/\alpha) + \log(1/\beta))^2 \log^3(1/\rho)}{\alpha^2 \rho^2}\right).$$

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 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 $\mathcal{A}_1$ 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 $1 - 3\beta$,

$$
\begin{aligned}
\|\mu_{disc} - \mu\|_\Sigma &\leqslant \|\mu_{disc} - \hat{\mu}\|_\Sigma + \|\hat{\mu} - \mu\|_\Sigma \\
&\leqslant \|\Sigma^{-1/2}[\mu_{disc} - \hat{\mu}]\|_2 + \alpha \\
&\leqslant \frac{1}{\sqrt{\lambda_d}}\|\mu_{disc} - \hat{\mu}\|_2 + \alpha \\
&\leqslant \frac{\alpha'}{\sqrt{\lambda_d}}\sqrt{d} + \alpha \\
&\leqslant \frac{\alpha\sqrt{\lambda}}{\sqrt{d}\sqrt{\lambda_d}}\sqrt{d} + \alpha \\
&\leqslant \frac{\alpha\sqrt{8\lambda_d}}{\sqrt{d}\sqrt{\lambda_d}}\sqrt{d} + \alpha = O(\alpha)
\end{aligned}
$$

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 3.18 (with ρ set to $\rho/2$), and applying

our transformation we can convert this to a $\rho/2$-replicable algorithm $\mathcal{A}_2$, 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 $\mathcal{A}_1$ and then using its estimate in the algorithm $\mathcal{A}_2$ 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 $\mathcal{A}_1$ and $\mathcal{A}_2$). Note that m_2 asymptotically dominates m_1, so the sample complexity of this entire procedure is

$$m = \tilde{O}\left(\frac{(d\log(1/\alpha) + \log(1/\beta))^2\log^3(1/\rho)}{\alpha^2\rho^2}\right).$$

$\square$

6.4.3 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 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 $\mathcal{A}$ that can succesfully distinguish between Case H_0, where $\mathcal{A}$ receives m samples from $N(\mu, \Sigma)$ and Case H_1 where $\mathcal{A}$ receives m samples from any distribution $N(\mu', \Sigma)$, such that $\|\mu' - \mu\|_\Sigma \geqslant \alpha$, 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 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

$$m = \tilde{O}\left(\frac{d^{1/2}}{\alpha^2 \rho^2}\right).$$

□

Acknowledgements

References

[ABMS20] Noga Alon, Amos Beimel, Shay Moran, and Uri Stemmer, *Closure properties for private classification and online prediction*, Conference on Learning Theory, PMLR, 2020, pp. 119–152.

[AHHM21] Noga Alon, Steve Hanneke, Ron Holzman, and Shay Moran, *A theory of PAC learnability of partial concept classes*, 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022, IEEE, 2021, pp. 658–671.

[ALMM19] Noga Alon, Roi Livni, Maryanthe Malliaris, and Shay Moran, *Private pac learning implies finite littlestone dimension*, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019, pp. 852–860.

[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*.

[ASZ21] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang, *Differentially private assouad, fano, and le cam*, Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide (Vitaly Feldman, Katrina Ligett, and Sivan Sabato, eds.), Proceedings of Machine Learning Research, vol. 132, PMLR, 2021, pp. 48–78.

[BBG18] Borja Balle, Gilles Barthe, and Marco Gaboardi, *Privacy amplification by subsampling: Tight analyses via couplings and divergences*, Proceedings of the 32nd International Conference on Neural Information Processing Systems (Red Hook, NY, USA), NIPS'18, Curran Associates Inc., 2018, p. 6280–6290.

[BCS20] Mark Bun, Marco Leandro Carmosino, and Jessica Sorrell, *Efficient, noise-tolerant, and private learning via boosting*, Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria] (Jacob D. Abernethy and Shivani Agarwal, eds.), Proceedings of Machine Learning Research, vol. 125, PMLR, 2020, pp. 1031–1077.

[BDPSS09] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz, *Agnostic online learning.*, COLT, vol. 3, 2009, p. 1.

[BDRS18] Mark Bun, Cynthia Dwork, Guy N. Rothblum, and Thomas Steinke, *Composable and versatile privacy via truncated CDP*, Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018 (Ilias Diakonikolas, David Kempe, and Monika Henzinger, eds.), ACM, 2018, pp. 74–86.

[BE02] Olivier Bousquet and André Elisseeff, *Stability and generalization*, The Journal of Machine Learning Research **2** (2002), 499–526.

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth, *Learnability and the vapnik-chervonenkis dimension*, Journal of the ACM (JACM) **36** (1989), no. 4, 929–965.

[BF16] Raef Bassily and Yoav Freund, *Typicality-based stability and privacy*, CoRR **abs/1604.03336** (2016).

[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*, Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual (Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, eds.), 2021, pp. 7950–7964.

[BI91] Gyora M Benedek and Alon Itai, *Learnability with respect to fixed distributions*, Theoretical Computer Science **86** (1991), no. 2, 377–389.

[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), IEEE, 2020, pp. 389–402.

[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** (1996), no. 3, 434–452.

[BMA19] Raef Bassily, Shay Moran, and Noga Alon, *Limits of private learning with access to public data*, Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada (Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, eds.), 2019, pp. 10342–10352.

[BMN+18] Raef Bassily, Shay Moran, Ido Nachum, Jonathan Shafer, and Amir Yehudayoff, *Learners that use little information*, Algorithmic Learning Theory, ALT 2018, 7-9 April 2018, Lanzarote, Canary Islands, Spain (Firdaus Janoos, Mehryar Mohri, and Karthik Sridharan, eds.), Proceedings of Machine Learning Research, vol. 83, PMLR, 2018, pp. 25–55.

[BMNS19] Amos Beimel, Shay Moran, Kobbi Nissim, and Uri Stemmer, *Private center points and learning of halfspaces*, Conference on Learning Theory, PMLR, 2019, pp. 269–282.

[BNS13] Amos Beimel, Kobbi Nissim, and Uri Stemmer, *Characterizing the sample complexity of private learners*, Proceedings of the 4th conference on Innovations in Theoretical Computer Science, 2013, pp. 97–110.

[BNS+16a] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan R. Ullman, *Algorithmic stability for adaptive data analysis*, Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016 (Daniel Wichs and Yishay Mansour, eds.), ACM, 2016, pp. 1046–1059.

[BNS16b] Amos Beimel, Kobbi Nissim, and Uri Stemmer, *Private learning and sanitization: Pure vs. approximate differential privacy*, Theory Comput. **12** (2016), no. 1, 1–61.

[BNS16c] Mark Bun, Kobbi Nissim, and Uri Stemmer, *Simultaneous private learning of multiple concepts*, Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, 2016, pp. 369–380.

[BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan, *Differentially private release and learning of threshold functions*, IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015 (Venkatesan Guruswami, ed.), IEEE Computer Society, 2015, pp. 634–649.

[Bro97] Andrei Z. Broder, *On the resemblance and containment of documents*, Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings (Bruno Carpentieri, Alfredo De Santis, Ugo Vaccaro, and James A. Storer, eds.), IEEE, 1997, pp. 21–29.

[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** (2019), no. 1.

[BUV18] Mark Bun, Jonathan R. Ullman, and Salil P. Vadhan, *Fingerprinting codes and the price of approximate differential privacy*, SIAM J. Comput. **47** (2018), no. 5, 1888–1938.

[CKM⁺20] Clément L. Canonne, Gautam Kamath, Audra McMillan, Jonathan R. Ullman, and Lydia Zakynthinou, *Private identity testing for high-dimensional distributions*, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, eds.), 2020.

[CLN⁺16] Rachel Cummings, Katrina Ligett, Kobbi Nissim, Aaron Roth, and Zhiwei Steven Wu, *Adaptive learning with robust generalization guarantees*, Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016 (Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, eds.), JMLR Workshop and Conference Proceedings, vol. 49, JMLR.org, 2016, pp. 772–814.

[DFH⁺15a] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth, *Generalization in adaptive data analysis and holdout reuse*, Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada (Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, eds.), 2015, pp. 2350–2358.

[DFH⁺15b] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth, *Preserving statistical validity in adaptive data analysis*, Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015 (Rocco A. Servedio and Ronitt Rubinfeld, eds.), ACM, 2015, pp. 117–126.

[DHS15] Ilias Diakonikolas, Moritz Hardt, and Ludwig Schmidt, *Differentially private learning of structured discrete distributions*, Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada (Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, eds.), 2015, pp. 2566–2574.

[DMNS16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith, *Calibrating noise to sensitivity in private data analysis*, J. Priv. Confidentiality **7** (2016), no. 3, 17–51.

[DMY16] Ofir David, Shay Moran, and Amir Yehudayoff, *Supervised learning through the lens of compression*, Advances in Neural Information Processing Systems **29** (2016), 2784–2792.

[DR14] Cynthia Dwork and Aaron Roth, *The algorithmic foundations of differential privacy*, Found. Trends Theor. Comput. Sci. **9** (2014), no. 3-4, 211–407.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan, *Boosting and differential privacy*, 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA, IEEE Computer Society, 2010, pp. 51–60.

[DSS+15] Cynthia Dwork, Adam D. Smith, Thomas Steinke, Jonathan R. Ullman, and Salil P. Vadhan, *Robust traceability from trace amounts*, IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015 (Venkatesan Guruswami, ed.), IEEE Computer Society, 2015, pp. 650–669.

[DW79] L. Devroye and T. Wagner, *Distribution-free performance bounds for potential function rules*, IEEE Transactions on Information Theory **25** (1979), no. 5, 601–604.

[GGKM21] Badih Ghazi, Noah Golowich, Ravi Kumar, and Pasin Manurangsi, *Sample-efficient proper pac learning with approximate differential privacy*, Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, 2021, pp. 183–196.

[GKM21] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi, *User-level differentially private learning via correlated sampling*, Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual (Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, eds.), 2021, pp. 20172–20184.

[GM82] Shafi Goldwasser and Silvio Micali, *Probabilistic encryption and how to play mental poker keeping secret all partial information*, Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '82, Association for Computing Machinery, 1982, p. 365–377.

[Gol21] Noah Golowich, *Differentially private nonparametric regression under a growth condition*, Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA (Mikhail Belkin and Samory Kpotufe, eds.), Proceedings of Machine Learning Research, vol. 134, PMLR, 2021, pp. 2149–2192.

[Hau92] David Haussler, *Decision theoretic generalizations of the pac model for neural net and other learning applications*, Information and computation **100** (1992), no. 1, 78–150.

[HKLM22] Max Hopkins, Daniel M. Kane, Shachar Lovett, and Gaurav Mahajan, *Realizable learning is all you need*, Conference on Learning Theory, 2-5 July 2022, London, UK (Po-Ling Loh and Maxim Raginsky, eds.), Proceedings of Machine Learning Research, vol. 178, PMLR, 2022, pp. 3015–3069.

[Hol09] Thomas Holenstein, *Parallel repetition: Simplification and the no-signaling case*, Theory Comput. **5** (2009), no. 1, 141–172.

[ILPS22] Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell, *Reproducibility in learning*, STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022 (Stefano Leonardi and Anupam Gupta, eds.), ACM, 2022, pp. 818–831.

[JKT20] Young Hun Jung, Baekjin Kim, and Ambuj Tewari, *On the equivalence between online and private learnability beyond binary classification*, Proceedings of the 34th International Conference on Neural Information Processing Systems (Red Hook, NY, USA), NIPS'20, Curran Associates Inc., 2020.

[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*, 11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA (Thomas Vidick, ed.), LIPIcs, vol. 151, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 31:1–31:17.

[KKMN09] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas, *Releasing search queries and clicks privately*, Proceedings of the 18th international conference on World wide web, 2009, pp. 171–180.

[KKMV23] Alkis Kalavasis, Amin Karbasi, Shay Moran, and Grigoris Velegkas, *Statistical indistinguishability of learning algorithms*, Personal communication, January 2023.

[KLM$^+$20] Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer, *Privately learning thresholds: Closing the exponential gap*, Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria] (Jacob D. Abernethy and Shivani Agarwal, eds.), Proceedings of Machine Learning Research, vol. 125, PMLR, 2020, pp. 2263–2285.

[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** (2011), no. 3, 793–826.

[KS14] Shiva P. Kasiviswanathan and Adam Smith, *On the 'semantics' of differential privacy: A bayesian formulation*, Journal of Privacy and Confidentiality **6** (2014), no. 1.

[KT02] Jon M. Kleinberg and Éva Tardos, *Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields*, J. ACM **49** (2002), no. 5, 616–639.

[Lon01] Philip M Long, *On agnostic learning with {0,*, 1}-valued and real-valued hypotheses*, International Conference on Computational Learning Theory, Springer, 2001, pp. 289–302.

[LS19] Katrina Ligett and Moshe Shenfeld, *A necessary and sufficient stability notion for adaptive generalization*, Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada (Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, eds.), 2019, pp. 11481–11490.

[MHS19] Omar Montasser, Steve Hanneke, and Nathan Srebro, *Vc classes are adversarially robustly learnable, but only improperly*, Conference on Learning Theory, PMLR, 2019, pp. 2512–2530.

[MT07] Frank McSherry and Kunal Talwar, *Mechanism design via differential privacy*, Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (USA), FOCS '07, IEEE Computer Society, 2007, p. 94–103.

[Nar22] Shyam Narayanan, *Private high-dimensional hypothesis testing*, Conference on Learning Theory, 2-5 July 2022, London, UK (Po-Ling Loh and Maxim Raginsky, eds.), Proceedings of Machine Learning Research, vol. 178, PMLR, 2022, pp. 3979–4027.

[Nat19] National Academies of Sciences, Engineering, and Medicine, *Reproducibility and replicability in science*, The National Academies Press, Washington, DC, 2019.

[NK19] Vaishnavh Nagarajan and J. Zico Kolter, *Uniform convergence may be unable to explain generalization in deep learning*, 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).

[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*, The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy] (Silvia Chiappa and Roberto Calandra, eds.), Proceedings of Machine Learning Research, vol. 108, PMLR, 2020, pp. 2830–2840.

[RRST16] Ryan M. Rogers, Aaron Roth, Adam D. Smith, and Om Thakkar, *Max-information, differential privacy, and post-selection hypothesis testing*, IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA (Irit Dinur, ed.), IEEE Computer Society, 2016, pp. 487–494.

[RRT+16] Maxim Raginsky, Alexander Rakhlin, Matthew Tsao, Yihong Wu, and Aolin Xu, *Information-theoretic analysis of stability and bias of learning algorithms*, 2016 IEEE Information Theory Workshop, ITW 2016, Cambridge, United Kingdom, September 11-14, 2016, IEEE, 2016, pp. 26–30.

[RW78] W. H. Rogers and T. J. Wagner, *A Finite Sample Distribution-Free Performance Bound for Local Discrimination Rules*, The Annals of Statistics **6** (1978), no. 3, 506 – 514.

[RZ16] Daniel Russo and James Zou, *Controlling bias in adaptive data analysis using information theory*, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016 (Arthur Gretton and Christian C. Robert, eds.), JMLR Workshop and Conference Proceedings, vol. 51, JMLR.org, 2016, pp. 1232–1240.

[SBG21] Satchit Sivakumar, Mark Bun, and Marco Gaboardi, *Multiclass versus binary differentially private pac learning*, Advances in Neural Information Processing Systems **34** (2021), 22943–22954.

[SSSSS10] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan, *Learnability, stability and uniform convergence*, The Journal of Machine Learning Research **11** (2010), 2635–2670.

[SZ20] Thomas Steinke and Lydia Zakynthinou, *Reasoning about generalization via conditional mutual information*, Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria] (Jacob D. Abernethy and Shivani Agarwal, eds.), Proceedings of Machine Learning Research, vol. 125, PMLR, 2020, pp. 3437–3452.

[Val84] Leslie G Valiant, *A theory of the learnable*, Proceedings of the sixteenth annual ACM symposium on Theory of computing, ACM, 1984, pp. 436–445.

[VC74] Vladimir Vapnik and Alexey Chervonenkis, *Theory of pattern recognition*, 1974.

[XR17] Aolin Xu and Maxim Raginsky, *Information-theoretic analysis of generalization capability of learning algorithms*, Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA (Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, eds.), 2017, pp. 2524–2533.

[ZH19] Tijana Zrnic and Moritz Hardt, *Natural analysts in adaptive data analysis*, Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Kamalika Chaudhuri and Ruslan Salakhutdinov, eds.), Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 7703–7711.

A 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 13: 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 $m = m(\rho, \alpha, \beta) \leqslant O\left(\frac{\log(|H|/\beta\rho)}{\alpha^2\rho^2}\right)$

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 A.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 13 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:*

$$\mathbf{Pr}_{r,S}\big[\mathcal{A}(S) \in [OPT, OPT + \alpha/2]\big] \geqslant 1 - \beta.$$

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:

$$B_1 = \left[0, \frac{\alpha}{8}\right), \ \ldots, B_{\frac{1}{\alpha}} = \left[1 - \frac{\alpha}{8}, 1\right].$$

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 ROUND$(OPT_S + \alpha/4) \in [OPT, OPT + \frac{\alpha}{2}]$ with high probability as desired. $\qquad\square$

B 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 B.1. *Let (X, H) be a class with Littlestone dimension d. Then the sample complexity of realizably replicably learning (X, H) is at most:*

$$n(\rho, \alpha, \beta) \leqslant \tilde{O}\left(\frac{d^{12} \log^3(1/\beta)}{\alpha^2 \rho^2}\right)$$

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 $\mathcal{L}$ have risk at most $\alpha/2$

2. *Heavy Hitter*: There exists $h \in H$ output with probability $\Omega(1/d)$

3. *Size*: $\mathcal{L}$ outputs at most $\exp(d^2 + d \log \frac{d}{\alpha\beta})$ hypotheses

4. *Sample Complexity*: $\mathcal{L}$ uses at most $\tilde{O}(\frac{d^6 \log^2 \frac{1}{\beta}}{\alpha^2})$ samples.

Note that for $\beta \leqslant O(1/d)$, any heavy hitter of this distribution is a good hypothesis, so it is enough to replicably output such a heavy hitter. Applying Theorem 6.7, this can be done ρ-replicably and with probability at least $1 - \beta$ using

$$O\left(\frac{\log^2 \frac{|\mathcal{D}| \log \frac{1}{\rho\beta}}{\eta} \log \frac{1}{\rho} + \log \frac{1}{\rho\beta}}{\eta^2 \rho^2} \log^3 \frac{1}{\rho}\right) = \tilde{O}\left(\frac{d^6 \log \frac{1}{\rho} + d^2 \log \frac{1}{\rho\beta}}{\rho^2} \log^3 \frac{1}{\rho}\right)$$

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. $\qquad\square$

C Additional Properties of Replicability

C.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 C.1. *Say an algorithm $\mathcal{A}$ 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 $\mathcal{A}'$ that replicably uses at most $k \cdot \max_{i \in [k]}\{b_i\}$ bits of randomness.*

Here, by "replicably uses" we mean that algorithm $\mathcal{A}'$ uses the same positions in the random string for each subroutine in every run of the algorithm.

Proof. Have $\mathcal{A}'$ 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 $\mathcal{A}$. At most $k \cdot \max_{i \in [k]}\{b_i\}$ bits of randomness are used. $\qquad\square$

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 C.2. *Say an algorithm $\mathcal{A}$ 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 $\mathcal{A}'$ 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. $\square$

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 $\mathcal{A}$ 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.

C.2 Replicability across Two Close Distributions

Next, we prove a simple Lemma bounding the effect of distributional shift on replicability.

Lemma C.3 (Replicability under Distributional Shift). *Let D_1 and D_2 be two distributions over $\mathcal{X}$ with total variational distance $d_{TV}(D_1, D_2) = \delta$. Let $\rho \geqslant 0$, and let $\mathcal{A}$ be a ρ-replicable algorithm that uses a sample of size exactly m. Then*

$$\mathbf{Pr}_{S_1 \sim D_1^m, S_2 \sim D_2^m, r}[\mathcal{A}(S_1; r) = \mathcal{A}(S_2; r)] \geqslant (1 - \delta)^{2m}\rho.$$

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,

$$\begin{aligned}
\mathbf{Pr}_{S_1 \sim D_1^m, S_2 \sim D_2^m, r}[\mathcal{A}(S_1; r) = \mathcal{A}(S_2; r)] &= (1 - \delta)^{2m}\mathbf{Pr}_{S_1 \sim D^m, S_2 \sim D^m, r}[\mathcal{A}(S_1; r) = \mathcal{A}(S_2; r)] \\
&\quad + \ldots \\
&\quad + \delta^{2m}\mathbf{Pr}_{S_1 \sim D'^m, S_2 \sim D''^m, r}[\mathcal{A}(S_1; r) = \mathcal{A}(S_2; r)] \\
&\geqslant (1 - \delta)^{2m}\mathbf{Pr}_{S_1 \sim D^m, S_2 \sim D^m, r}[\mathcal{A}(S_1; r) = \mathcal{A}(S_2; r)] \\
&= (1 - \delta)^{2m}\rho.
\end{aligned}$$

$\square$

However, Lemma C.3 may not be tight for specific class of algorithms, functions, or distributions.

D 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

$\mathcal{X}$ input spaces

$\mathcal{Y}$ 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

$\mathcal{A}, \mathcal{B}$ algorithms

I interval

v threshold values

$\|$ concatenation of strings