

SAMPLING CORRECTORS*

CLÉMENT L. CANONNE[†], THEMIS GOULEAKIS[‡], AND RONITT RUBINFELD[§]

Abstract. In many situations, sample data is obtained from a noisy or imperfect source. In order to address such corruptions, this paper introduces the concept of a *sampling corrector*. Such algorithms use structure that the distribution is purported to have, in order to allow one to make “on-the-fly” corrections to samples drawn from probability distributions. These algorithms then act as filters between the noisy data and the end user. We show connections between sampling correctors, distribution learning algorithms, and distribution property testing algorithms. We show that these connections can be utilized to expand the applicability of known distribution learning and property testing algorithms as well as to achieve improved algorithms for those tasks. As a first step, we show how to design sampling correctors using proper learning algorithms. We then focus on the question of whether algorithms for sampling correctors can be more efficient in terms of sample complexity than learning algorithms for the analogous families of distributions. When correcting monotonicity, we show that this is indeed the case when also granted query access to the cumulative distribution function. We also obtain sampling correctors for monotonicity even without this stronger type of access, provided that the distribution be originally *very* close to monotone (namely, at a distance $O(1/\log^2 n)$). In addition to that, we consider a restricted error model that aims at capturing “missing data” corruptions. In this model, we show that distributions that are close to monotone have sampling correctors that are significantly more efficient than achievable by the learning approach. We consider the question of whether an additional source of independent random bits is required by sampling correctors to implement the correction process. We show that for correcting close-to-uniform distributions and close-to-monotone distributions, no additional source of random bits is required, as the samples from the input source itself can be used to produce this randomness.

Key words. randomized algorithms, probability distributions, property testing, learning

AMS subject classifications. 68W20, 68R01, 68Q87

DOI. 10.1137/16M1076666

1. Introduction. Data consisting of samples from distributions is notorious for reliability issues: sample data can be greatly affected by noise, calibration problems, or other faults in the sample recording process; portions of data may be lost; extraneous samples may be erroneously recorded. Such noise may be completely random or may have some underlying structure. To give a sense of the range of difficulties one might have with sample data, we mention some examples: a sensor network which tracks traffic data may have dead sensors which transmit no data at all, or other sensors that are defective and transmit arbitrary numbers. Sample data from surveys may suffer from response rates that are correlated with location or socioeconomic factors. Sample data from species distribution models are prone to geographic location errors [HBTB14].

*Received by the editors May 25, 2016; accepted for publication (in revised form) March 28, 2018; published electronically July 19, 2018. An extended abstract of this work appeared in the Proceedings of the 2016 Conference on Innovations in Theoretical Computer Science, 2016.

<http://www.siam.org/journals/sicomp/47-4/M107666.html>

Funding: The first author’s research was supported by a Motwani Postdoctoral Fellowship and in part by NSF grants CCF-1115703 and NSF CCF-1319788. The second author’s research was supported by NSF grants CCF-1420692, CCF-1217423, and CCF-1065125. The third author’s research was supported by ISF grant 1536/14 and NSF grants CCF-1420692, CCF-1217423, and CCF-1065125.

[†]Stanford University, Stanford, CA 94305 (ccanonne@cs.stanford.edu).

[‡]CSAIL, MIT, Cambridge, MA 02139 (tgoule@mit.edu).

[§]CSAIL, MIT, Cambridge, MA 02139 and the Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, 6997801 Israel (ronitt@csail.mit.edu).

Statisticians have grappled with defining a methodology for working with distributions in the presence of noise by *correcting* the samples. If, for example, you know that the uncorrupted distribution is Gaussian, then it would be natural to correct the samples of the distribution to the nearest Gaussian. The challenge in defining this methodology is, how do you correct the samples if you do not know much about the original uncorrupted distribution? To analyze distributions with noise in a principled way, approaches have included *imputation* [LR02, Sch97, STP07] for the case of missing or incomplete data and *outlier detection and removal* [Haw80, Bar78, IH93] to handle “extreme points” deviating significantly from the underlying distribution. More generally, the question of coping with the *sampling bias* inherent to many strategies (such as opportunity sampling) used in studying rare events or species, or with inaccuracies in the reported data, is a key challenge in many of the natural and social sciences (see, e.g., [SL82, SV03, PMC08]). While these problems are usually dealt with by drawing on additional knowledge or by using specific modeling assumptions, no general procedure is known that addresses them in a systematic fashion.

In this work, we propose a methodology which is based on using *known structural properties* of the distribution to design *sampling correctors* which “correct” the sample data. While assuming these structural properties is in itself a type of modeling, it is in general much weaker than postulating a strict form of the data (e.g., that it follows a linear model perturbed by Gaussian noise). Examples of structural properties which might be used to correct samples include the property of being bimodal, a mixture of several Gaussians, a mixture of piecewise-polynomial distributions, or an independent joint distribution. Within this methodology, the main question is: how best can one output samples of a distribution in which on one hand, the structural properties are restored, and on the other hand, the corrected distribution is close to the original distribution? We show that this task is intimately connected to distribution learning tasks, but we also give instances in which such tasks can be performed strictly more efficiently.

1.1. Our model. We introduce two (related) notions of algorithms to correct distributions: *sampling correctors* and *sampling improvers*. Although the precise definitions are deferred to section 3, we describe and state informally what we mean by these. In what follows, \mathcal{X} is a finite domain, \mathcal{P} is any fixed property of distributions, i.e., a subset of distributions, over \mathcal{X} , and distances between distributions are measured according to their *total variation distance*.¹

A *sampling corrector* for \mathcal{P} is a randomized algorithm which gets samples from a distribution D guaranteed to be ε -close to having property \mathcal{P} and outputs a sample from a “corrected distribution” \tilde{D} which, with high probability, (a) *has* the property and (b) is still close to the original distribution D (i.e., within distance ε_1). The *sample complexity* of such a corrector is the number of samples it needs to obtain from D in order to output one from \tilde{D} .

To make things concrete, we give a simple example of correcting independence of distributions over a product space $[n] \times [m]$. For each pair of samples (x, y) and (x', y') from a distribution D which is ε -close to independent, output *one* sample (x, y') . As x and y' are independent, the resulting distribution clearly has the property, and it can be shown that if D was indeed ε -close to independent, then the distribution of (x, y') will indeed be 3ε -close to D [SV98]. (Whether this sample complexity can be reduced further to $q < 2$, even on average, is an open question.)

¹The total variation distance is defined as $d_{\text{TV}}(D_1, D_2) \stackrel{\text{def}}{=} \max_{S \subseteq \mathcal{X}} (D_1(S) - D_2(S)) = \frac{1}{2} \sum_{x \in \mathcal{X}} |D_1(x) - D_2(x)|$.

Note that in some settings it may be too much to ask for complete correction (or may even not be the most desirable option). For this reason, we also consider the weaker notion of *sampling improvers*, which is similar to a sampling corrector but is only required to transform the distribution into a new distribution which is *closer* to having the property \mathcal{P} .

One naive way to solve these problems, the “learning approach,” is to approximate the probability mass function of D and find a candidate $\tilde{D} \in \mathcal{P}$. Since we assume we have a complete description of \tilde{D} , we can then output samples according to \tilde{D} without further access to D . In general, such an approach can be very inefficient in terms of time complexity. However, if there is an *efficient* agnostic proper learning algorithm² for \mathcal{P} , we show that this approach can lead to efficient sampling correctors. For example, we use such an approach to give sampling correctors for the class of monotone distributions.

In our model, we wish to optimize the following two parameters of our correcting algorithms. The first parameter is the number of samples of D needed to output samples of \tilde{D} . The second parameter is the number of *additional* truly random bits needed for outputting samples of \tilde{D} . Note that in the above learning approach, the dependence on each of these parameters could be quite large. Although these parameters are not independent of each other (if D is of high enough entropy, then it can be used to simulate truly random bits), they can be thought of as complementary, as one typically will aim at a tradeoff between the two. Furthermore, a parsimonious use of extra random bits may be crucial for some applications, while in others the correction of the data itself is the key factor; for this reason, we track each of the parameters separately. For any property \mathcal{P} , the main question is whether one can achieve improved complexity in terms of these parameters over the use of the naive (agnostic) learning approach for \mathcal{P} .

1.2. Our results. Throughout this paper, we will focus on two particular properties of interest, namely *uniformity* and *monotonicity*. The first one, arguably one of the most natural and illustrative properties to be considered, is nonetheless deeply challenging in the setting of randomness scarcity. As for the second, not only does it provide insight in the workings of sampling correctors as well as nontrivial connections and algorithmic results, but it is also one of the most-studied classes of distributions in the statistics and probability literature, with a body of work covering several decades (see, e.g., [Gre56, Bir87, BKR04, DDS14] or [DDS+13] for a detailed list of references). Moreover, recent work on distribution testing [DDS+13, CDGR17] shows strong connections between monotonicity and a wide range of other properties, such as, for instance, log-concavity, monotone hazard risk, and Poisson binomial distributions (PBDs). This gives evidence that the study of monotone distributions may have direct implications for correction of many of these “shape-constrained properties.”

Sampling correctors, learning algorithms, and property testing algorithms. We begin by showing implications of the existence of sampling correctors for the existence of various types of learning and property testing algorithms in other models. We

²Recall that a *learning algorithm* for a class of distributions \mathcal{C} is an algorithm which gets independent samples from an unknown distribution $D \in \mathcal{C}$, and on input ε must, with high probability, output a hypothesis which is ε -close to D in total variation distance. If the hypotheses the algorithm produces are guaranteed to belong to \mathcal{C} as well, we call it a *proper learning algorithm*. Finally, if the—not-necessarily proper—algorithm is able to learn distributions that are only *close* to \mathcal{C} , returning a hypothesis at a distance at most $\text{OPT} + \varepsilon$ from D —where OPT is the distance from D to the class, it is said to be *agnostic*. For a formal definition of these concepts, the reader is referred to section 5.2 and Appendix B.

first show in Theorem 5.1 that learning algorithms for a distribution class imply sampling correctors for distributions in this class (under *any* property to correct) with the same sample complexity, though not necessarily the same running time dependency. However, when efficient agnostic proper learning algorithms for a distribution class exist, we show that there are efficient sampling correctors for the same class. In [Bir87, CDSS14] efficient algorithms for agnostic learning of concise representations for several families of distributions are given, including distributions that are monotone, k -histograms, Poisson binomial, and sums of k independent random variables. Not all of these algorithms are proper.

Next, we show in Theorem 5.4 that the existence of (a) an efficient learning algorithm, as, e.g., in [ILR12, CDSS13, DDS12, DDO+13], and (b) an efficient sampling corrector for a class of distributions implies an efficient *agnostic* learning algorithm for the same class of distributions. It is well known that agnostic learning can be much harder than nonagnostic learning, as in the latter the algorithm is able to leverage structural properties of the class \mathcal{C} . Thus, by the above result, we also get that any agnostic learning lower bounds can be used to obtain sampling corrector lower bounds.

Our third result in this section, Theorem 5.7, shows that an efficient property tester, an efficient distance estimator (which computes an additive estimate of the distance between two distributions), and an efficient sampling corrector for a distribution class imply a tolerant property tester with complexity equal to the complexity of correcting the number of samples required to run both the tester and estimator.³ As tolerant property testing can be much more difficult than property testing [GR00, BFR+10, Pan08, VV11], this gives a general-purpose way of getting both upper bounds on tolerant property testing and lower bounds on sampling correctors.

We describe how these results can be employed in section 5, where we give specific applications in achieving improved property testers for various properties.

Is sampling correction easier than learning? We next turn to the question of whether there are natural examples of sampling correctors whose query complexity is asymptotically smaller than that of distribution learning algorithms for the same class. While the sample complexity of learning monotone distributions is known to be $\Omega(\log n)$ [Bir87] (this lower bound on the sample and query complexity holds even when the algorithm is allowed both to make queries to the cumulative distribution function and to access samples of the distribution), we present in section 6.2 an oblivious sampling corrector for monotone distributions whose sample complexity is $O(1)$ and that corrects error that is smaller than $\varepsilon \leq O(1/\log^2 n)$. This is done by first implicitly approximating the distribution by a histogram on only a small number of intervals, using ingredients from [Bir87]. This (very close) approximation can then be combined, still in an oblivious way, with a carefully chosen slowly decreasing distribution, so that the resulting mixture is guaranteed to be not only monotone but also close to the original distribution.

It is an open question whether there exist sampling correctors for monotone distributions with sample complexity $o((\log n)/\varepsilon^3)$ that can correct arbitrary error $\varepsilon \in (0, 1)$, thus beating the sample complexity of the learning approach. (We note however that a logarithmic dependence on n is inherent when $\varepsilon = \omega(1/\log n)$, as pointed out to us by Valiant [Val15].)

³Recall that the difference between testing and tolerant testing lies in that the former asks to distinguish whether an unknown distribution *has* a property, or is far from it, while the latter requires to decide whether the distribution is *close* to the property versus far from it. (See Appendix B for the rigorous definition.)

Assuming a stronger type of access to the unknown distribution, namely, query access to its cumulative distribution function (cdf) as in [BDKR05, CR14], we describe in section 6.3 a sampling corrector for monotonicity with (expected) query complexity $O(\sqrt{\log n})$ which works for arbitrary $\varepsilon \in (0, 1)$. At a high level, our algorithm combines the “succinct histogram” technique mentioned above with a two-level bucketing approach to correct the distribution first at a very coarse level only (on “superbuckets”) and defers the finer corrections (within a given superbucket) to be made on-the-go at query time. The challenge in this last part is that one must ensure that all of these disjoint local corrections are consistent with each other—and crucially, *with all future sample corrections*. To achieve this, we use a boundary correction subroutine which fixes potential violations between two neighboring superbuckets by evening out the boundary differences. To make it possible, we use rejection sampling to allocate adaptively an extra “budget” to each superbucket that this subroutine can use for corrections.

Restricted error models. Since many of the sampling correction problems are difficult to solve in general, we suggest error models for which more efficient sampling correction algorithms may exist. A first class of error models, which we refer to as *missing data errors*, is introduced in section 7 and defined as follows: given a distribution over $[n]$, all samples in some interval $[i, j]$ for $1 < i < j < n$ are deleted. Such errors could correspond to samples from a sensor network where one of the sensors ran out of power, emails mistakenly deleted by a spam filter, or samples from a study in which some of the paperwork got lost. Whenever the input distribution D , whose distance from monotonicity is $\varepsilon \in (0, 1)$, falls under this model, we give a sampling improver that is able to find a distribution both ε_2 -close to monotone and $O(\varepsilon)$ -close to the original using $\tilde{O}(1/\varepsilon_2^3)$ samples. The improver works in two stages. In the “preprocessing stage,” we detect the location of the missing interval (when the missing weight is sufficiently large) and then estimate its missing weight, using a “learning through testing” approach from [DDS14] to keep the sample complexity under control. In the second stage, we give a procedure by which the algorithm can use its knowledge of the estimated missing interval to correct the distribution by rejection sampling.

Randomness scarcity. We then consider the case where only a limited amount of randomness (other than the input distribution) is available and optimizing its uses, possibly at the cost of worse parameters and/or sample complexity of our sampling improvers, is crucial. This captures situations where generating the random bits the algorithm use is either expensive⁴ (as in the case of physical implementations relying on devices, such as Geiger counters or Zener diodes) or undesirable (e.g., when we want the output distribution to be a deterministic function of the input data, for the sake of reproducibility or parallelization). We focus on this setting in section 8 and provide sampling correctors and improvers for uniformity that use samples *only* from the input distribution. For example, we give a sampling improver that, given access to distribution ε -close to uniform, grants access to a distribution ε_2 -close to uniform distribution and has *constant* sample complexity $O_{\varepsilon, \varepsilon_2}(1)$. We achieve this by exploiting the fact that the uniform distribution is not only an absorbing element for convolution in Abelian groups but also an *attractive fixed point* with high convergence rate. That is, by convolving a distribution with itself (i.e., summing independent samples modulo the order of the group) one gets very quickly close to uniform. Combining this idea with a different type of improvement (based on a von

⁴On this topic, see for instance the discussion in [KR94, IZ89] and references therein.

Neumann-type “trick”) allows us to obtain an essentially optimal tradeoff between closeness to uniform and to the original distribution.

1.3. Open problems.

Correcting vs. learning. A main direction of interest would be to obtain more examples of properties for which correcting is strictly more efficient than (agnostic or nonagnostic) learning. Such examples would be insightful even if they are more efficient only in terms of the number of samples required from the original distribution, without considering the additional randomness requirements for generating the distribution. More specifically, one may ask whether there exists a sampling corrector for monotonicity of distributions (i.e., one that beats the learning bound from Lemma 6.1) for all $\epsilon < 1$ which uses at most $o((\log n)/\epsilon^3)$ samples from the original distribution per sample output of the corrected distribution. Other properties of interest, among many, include log-concavity of distributions, having a piecewise-constant density (i.e., being a k -histogram for some fixed value k), or being a PBD.

The power of additional queries. Following the line of work pursued in [CFG13, CRS14, CR14] (in the setting of distribution testing), it is natural in many situations to consider additional types of queries to the input distribution, e.g., either *conditional queries* (getting a sample conditioned on a specific subset of the domain) or *cumulative queries* (granting query access to the cumulative distribution function, besides the usual sampling). By providing algorithms with this extended access to the underlying probability distribution, can one obtain faster sampling correctors for specific properties, as we do in section 6.3 in the case of monotonicity?

Confidence boosting. Suppose that there exists, for some property \mathcal{P} , a sampling improver \mathcal{A} that only guarantees a success probability⁵ of $2/3$. Using \mathcal{A} as a black-box can one design a sampling improver \mathcal{A}' which succeeds with probability $1 - \delta$ for any δ ?

More precisely, let \mathcal{A} be a batch improver for \mathcal{P} which, when queried, makes $q(\epsilon_1, \epsilon_2)$ queries and provides $t \geq 1$ samples, with success probability at least $2/3$. Having black-box access to \mathcal{A} , can we obtain a batch improver \mathcal{A}' which on input $\delta > 0$ provides $t' \geq 1$ samples, with success probability at least $1 - \delta$? If so, what is the best t' one can achieve, and what is the minimum query complexity of \mathcal{A}' one can get (as a function of $q(\cdot, \cdot)$, t' , and δ)?

This is known for property testing (by running the testing algorithm independently $O(\log(1/\delta))$ times and taking the majority vote), as well as for learning (again, by running the learning algorithm many times, and then doing hypothesis testing, e.g., à la [DK14, Theorem 19]). However, these approaches do not straightforwardly generalize to sampling improvers or correctors, respectively, because the output is not a single bit, and as we only obtain a sequence of samples (instead of an actual, fully specified hypothesis distribution).

1.4. Previous work. Dealing with noisy or incomplete datasets has been a challenge in statistics and data sciences, and many methods have been proposed to handle them. One of the most widely used, *multiple imputation* (one of many variants of the general paradigm of *imputation*) was first introduced by Rubin [Rub87] and consists of the creation of several complete datasets from an incomplete one. Specifically, one first obtains these new datasets by filling in the missing values randomly according to

⁵We note that the case of interest here is of batch sampling improvers; indeed, in order to generate a single draw, a sampling improver acts in a nontrivial way only if the parameter ϵ is greater than its failure probability δ . If not, a draw from the original distribution already satisfies the requirements.

a maximum likelihood (ML) distribution computed from the observations and a modeling assumption made on the data. The parameters of this model are then updated using the new datasets, and the ML distribution is computed again. This resembles the expectation-maximization (EM) algorithm, a heuristic which can also be used for similar problems, as, e.g., in [DLR77]. In cases where the EM algorithm does converge to the ML distribution, after a few iterations one can get both accurate parameter estimates and the right distribution to sample data from. Assuming the assumptions chosen to model the data did indeed reflect its true distribution and that the number of these new datasets was large enough, this can be shown to yield statistically accurate and unbiased results [Sch97, LR02].

From a theoretical computer science perspective, the problem of local correction of data has received much attention in the contexts of self-correcting programs, locally correctable codes, and local filters for graphs and functions over $[n]^d$ (some examples include [BLR90, Yek10, ACCL08, SS10, BGJ+12, JR11]). To the best of our knowledge, this is the first work to address the correction of data from distributions. (We observe that Chakraborty, García-Soriano, and Matsliah consider in [CGM11] a different question, although of a similar distributional flavor; namely, given query access to a Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ which is close to a k -junta f^* , they show how to approximately generate uniform Probably Approximately Correct style samples of the form $\langle x, g^*(x) \rangle$, where $x \in \{0, 1\}^k$ and g^* is the function underlying f^* . They then describe how to apply this “noisy sampler” primitive to test whether a function is close to being a junta.)

In this work, we show that the problem of estimating distances between distributions is related. There has been much work on this topic, but we note the following result: [DDS+13] show how to estimate the total variation distance between k -modal probability distributions.⁶ The authors give a reduction of their problem into one with logarithmic size, using a result by Birgé on monotone distributions [Bir87]. In particular, one can partition the domain $\mathcal{X} = [n]$ into $\log n/\varepsilon$ intervals in an oblivious way, such that the “flattening” of any monotone distribution according to that interval is $O(\varepsilon)$ -close to the original one. We use similar ideas in order to obtain some of the results in the present paper.

Another related field in statistics is that of *robust statistics*, which is concerned with estimating the parameters of a model in spite of a fraction of the data being corrupted (equivalently, under some model misspecification). This type of question is similar in spirit to our setting, in that it aims at overcoming noisy or corrupted data; however, the focus there is on *learning* characteristics of the purported model, instead of “removing” the noise (which could be seen as a less demanding goal, and a possible approach towards the learning task itself). Well-studied in statistics since the seminal work of Tukey [Tuk60], robust statistics have recently been studied from a computational and algorithmic viewpoint; we refer the reader to [HR09, HRRS11] for surveys and overviews of the field itself and to [LRV16, DKK+16, CSV17, DKK+17, BDLS17, DKK+18] for recent advances from a theoretical computer science perspective.

Finally, it is instructive to compare the goal of our model of distribution sampling correctors to that of extractors; in spite of many similarities, the two have essential differences, and the results are in many cases incomparable. We defer this discussion to section 8.1.

⁶A probability distribution D is k -modal if there exists a partition of $[n]$ in k intervals such that D is monotone (increasing or decreasing) on each.

2. Preliminaries. Hereafter, we write $[n]$ for the set $\{1, \dots, n\}$ and \log for the logarithm in base 2. A *probability distribution* over a finite domain \mathcal{X} is a nonnegative function $D: \mathcal{X} \rightarrow [0, 1]$ such that $\sum_{x \in \mathcal{X}} D(x) = 1$; we denote by $\mathcal{U}_{\mathcal{X}}$ the uniform distribution on \mathcal{X} . Moreover, given a distribution D over \mathcal{X} and a set $S \subseteq \mathcal{X}$, we write $D(S)$ for the total probability weight $\sum_{x \in S} D(x)$ assigned to S by D .

Previous tools from probability. As previously mentioned, in this work we will be concerned with the total variation distance between distributions. Of interest for the analysis of some of our algorithms, and assuming \mathcal{X} is totally ordered (in our case, $\mathcal{X} = [n]$), one can also define the *Kolmogorov distance* between D_1 and D_2 as

$$(1) \quad d_K(D_1, D_2) \stackrel{\text{def}}{=} \max_{x \in \mathcal{X}} |F_1(x) - F_2(x)|,$$

where F_1 and F_2 are the respective cdfs of D_1 and D_2 . Thus, the Kolmogorov distance is the ℓ_∞ distance between the cdf, and $d_K(D_1, D_2) \leq d_{\text{TV}}(D_1, D_2) \in [0, 1]$.

We first state the following theorem, which guarantees that for any two distributions D_1, D_2 , applying any (possibly randomized) function to both D_1 and D_2 can never increase their total variation distance.

Fact 2.1 (data processing inequality for total variation distance). Let D_1, D_2 be two distributions over a domain Ω . Fix any randomized function⁷ F on Ω , and let $F(D_1)$ be the distribution such that a draw from $F(D_1)$ is obtained by drawing independently x from D_1 and f from F and then outputting $f(x)$ (likewise for $F(D_2)$). Then we have

$$d_{\text{TV}}(F(D_1), F(D_2)) \leq d_{\text{TV}}(D_1, D_2).$$

Finally, we recall below a fundamental fact from probability theory that will be useful to us, the *Dvoretzky–Kiefer–Wolfowitz (DKW) inequality*. Informally, this result says that one can learn the cumulative distribution function of a distribution up to an additive error ε in ℓ_∞ distance by taking only $O(1/\varepsilon^2)$ samples from it.

THEOREM 2.2 ([DKW56, Mas90]). *Let D be a distribution over $[n]$. Given m independent samples x_1, \dots, x_m from D , define the empirical distribution \hat{D} as follows:*

$$\hat{D}(i) \stackrel{\text{def}}{=} \frac{|\{j \in [m] : x_j = i\}|}{m}, \quad i \in [n].$$

Then, for all $\varepsilon > 0$, $\Pr[d_K(D, \hat{D}) > \varepsilon] \leq 2e^{-2m\varepsilon^2}$, where the probability is taken over the samples.

In particular, setting $m = \Theta(\frac{\log(1/\delta)}{\varepsilon^2})$ we get that $d_K(D, \hat{D}) \leq \varepsilon$ with probability at least $1 - \delta$.

Flattenings. For a distribution D and a partition of $[n]$ into intervals $\mathcal{I} = (I_1, \dots, I_\ell)$, we define the *flattening of D with relation to \mathcal{I}* as the distribution $\Psi_{\mathcal{I}}(D)$, where $\Psi_{\mathcal{I}}(D)(i) = D(I_k)/|I_k|$ for all $k \in [\ell]$ and $i \in I_k$. A straightforward computation (see Appendix C) shows that such flattening cannot increase the distance between two distributions, i.e.,

$$(2) \quad d_{\text{TV}}(\Psi_{\mathcal{I}}(D_1), \Psi_{\mathcal{I}}(D_2)) \leq d_{\text{TV}}(D_1, D_2).$$

⁷Which can be seen as a distribution over functions over Ω .

Monotone distributions. We say that a distribution D on $[n]$ is *monotone* (non-increasing) if its probability mass function is nonincreasing, that is, if $D(1) \geq \dots \geq D(n)$. When dealing with monotone distributions, it will be useful to consider the *Birgé decomposition*, which is a way to approximate any monotone distribution D by a histogram, where the latter is supported by logarithmically many intervals *which crucially do not depend on D itself*.

DEFINITION 2.3 (Birgé decomposition). *Given a parameter $\alpha > 0$, the corresponding (oblivious) Birgé decomposition of $[n]$ is the partition $\mathcal{I}_\alpha = (I_1, \dots, I_\ell)$, where $\ell = \Theta(\frac{\ln(\alpha n + 1)}{\alpha}) = \Theta(\frac{\log n}{\alpha})$ and $|I_k| = \lfloor (1 + \alpha)^k \rfloor$, $1 \leq k \leq \ell$.*

For a distribution D and parameter α , define $\Phi_\alpha(D)$ to be the flattened distribution with relation to the oblivious decomposition \mathcal{I}_α ; that is, $\Phi_\alpha(D) = \Psi_{\mathcal{I}_\alpha}(D)$. The next theorem states that every monotone distribution can be well approximated by its flattening on the Birgé decomposition’s intervals.

THEOREM 2.4 ([Bir87, DDS+13]). *If D is monotone, then $d_{TV}(D, \Phi_\alpha(D)) \leq \alpha$.*

As a corollary, one can extend the theorem to distributions only promised to be close to monotone.

COROLLARY 2.5. *Suppose that D is ε -close to monotone, and let $\alpha > 0$. Then $d_{TV}(D, \Phi_\alpha(D)) \leq 2\varepsilon + \alpha$. Furthermore, $\Phi_\alpha(D)$ is also ε -close to monotone.*

Access to the distributions. While we will mostly be concerned in this work with the standard model of access to the probability distributions, where the algorithm is provided with independent samples from an unknown distribution D , the concepts we introduce and some of our results apply to some other types of access as well. One in particular, the cumulative dual access model, grants the algorithms the ability to query the value of the cdf of D , in addition to regular sampling.⁸ (We observe, as in [CR14], that this type of query access is for instance justified when the distribution originates from a sorted dataset, in which case such queries can be implemented with only a logarithmic overhead.)

Unless explicitly specified otherwise, our algorithms only assume standard sampling access; the formal definitions of the two models mentioned above can be found in Appendix B.

3. Our model: Definitions. In this section, we state the precise definitions of sampling correctors, improvers, and batch sampling improvers. To get an intuition, the reader may think, for instance, of the parameter ε_1 below as being 2ε and the error probability δ as $1/3$. Although all definitions are presented in terms of the total variation distance, analogous definitions in terms of other distances can also be made.

DEFINITION 3.1 (sampling corrector). *Fix a given property \mathcal{P} of distributions on \mathcal{X} . An $(\varepsilon, \varepsilon_1)$ -sampling corrector for \mathcal{P} is a randomized algorithm which is given parameters $\varepsilon, \varepsilon_1 \in (0, 1]$ such that $\varepsilon_1 \geq \varepsilon$ and $\delta \in [0, 1]$, as well as sampling access to a distribution D . Under the promise that $d_{TV}(D, \mathcal{P}) \leq \varepsilon$, the algorithm must provide, with probability at least $1 - \delta$ over the samples it draws and its internal randomness, sampling access to a distribution \tilde{D} such that*

- (i) \tilde{D} is close to D : $d_{TV}(\tilde{D}, D) \leq \varepsilon_1$;
- (ii) \tilde{D} has the property $\tilde{D} \in \mathcal{P}$.

⁸See also [Can15] for a summary and comparison of the different existing access models.

In other terms, with high probability the algorithm will simulate exactly a sampling oracle for \tilde{D} . The query complexity $q = q(\varepsilon, \varepsilon_1, \delta, \mathcal{X})$ of the algorithm is the number of samples from D it takes per query in the worst case.

One can define a more general notion, which allows the algorithm to only get closer to the desired property and convert some type of access ORACLE_1 into some other type of access ORACLE_2 (e.g., from sampling to evaluation access).

DEFINITION 3.2 (sampling improver (general definition)). *Fix a given property \mathcal{P} over distributions on \mathcal{X} . A sampling improver for \mathcal{P} (from ORACLE_1 to ORACLE_2) is a randomized algorithm which, given parameter $\varepsilon \in (0, 1]$ and ORACLE_1 access to a distribution D with the promise that $d_{\text{TV}}(D, \mathcal{P}) \leq \varepsilon$ as well as parameters $\varepsilon_1, \varepsilon_2 \in [0, 1]$ satisfying $\varepsilon_1 + \varepsilon_2 \geq \varepsilon$, provides, with probability at least $1 - \delta$ over the answers from ORACLE_1 and its internal randomness, ORACLE_2 access to a distribution \tilde{D} such that*

$$\text{(Close to } D) \quad d_{\text{TV}}(\tilde{D}, D) \leq \varepsilon_1,$$

$$\text{(Close to } \mathcal{P}) \quad d_{\text{TV}}(\tilde{D}, \mathcal{P}) \leq \varepsilon_2.$$

In other terms, with high probability the algorithm will simulate exactly ORACLE_2 access to \tilde{D} . The query complexity $q = q(\varepsilon, \varepsilon_1, \varepsilon_2, \delta, \mathcal{X})$ of the algorithm is the number of queries it makes to ORACLE_1 in the worst case.

Finally, one may ask for such an improver to provide *many* samples from the (same) improved distribution,⁹ where “many” is a number committed in advance. We refer to such an algorithm as a *batch sampling improver* (or, similarly, batch sampling corrector).

DEFINITION 3.3 (batch sampling improver). *For \mathcal{P} , D , ε , $\varepsilon_1, \varepsilon_2 \in [0, 1]$ as above, and parameter $m \in \mathbb{N}$, a batch sampling improver for \mathcal{P} (from ORACLE_1 to ORACLE_2) is a sampling improver which provides, with probability at least $1 - \delta$, ORACLE_2 access to \tilde{D} for as many as m queries, in between which it is allowed to maintain some internal state ensuring consistency. The query complexity q of the algorithm is now allowed to depend on m as well; i.e., $q = q(\varepsilon, \varepsilon_1, \varepsilon_2, m, \delta, \mathcal{X})$.*

Note that, in particular, when providing sampling access to \tilde{D} the batch improver must guarantee independence of the m samples. When ε_2 is set to 0 in the above definition, we will refer to the algorithm as a batch sampling corrector, with query complexity $q(\varepsilon, \varepsilon_1, m, \delta, \mathcal{X})$.

Remark 3.4 (on parameters of interest). We observe that the regime of interest of our correctors and improvers is when the number of corrected samples to output is at least of the order $\Omega(1/\varepsilon)$. Indeed, if fewer samples are required, then the assumption that the distribution D be ε -close to having the property implies that—with high probability—a small number of samples from D will be indistinguishable from the closest distribution having the property. (So that, intuitively, they are already “as good as it gets,” and need not be corrected.)

Remark 3.5 (on testing lower bounds). A similar observation holds for properties \mathcal{P} that are known to be *hard to test*, that is, for which some lower bound of $q(n, \varepsilon)$

⁹Indeed, observe that as sampling correctors and improvers are randomized algorithms with access to their “own” coins, there is no guarantee that fixing the input distribution D would lead to the same output distribution \tilde{D} . This is particularly important when providing other types of access (e.g., evaluation queries) to \tilde{D} than only sampling.

samples holds to decide whether a given distribution satisfies \mathcal{P} , or is ε -far from it. In light of such a lower bound, one may wonder whether there is something to be gained in correcting $m < q(n, \varepsilon)$ samples, instead of simply using m samples from the original distribution altogether. However, such a result only states that there exists *some* worst-case instance D^* that is at distance ε from the property \mathcal{P} , yet requires this many samples to be distinguished from it, so that any algorithm relying on samples from distributions satisfying \mathcal{P} could be fed $q(n, \varepsilon) - 1$ samples from this particular D^* without complaining. Yet, for “typical” distributions that are ε -close to \mathcal{P} , far fewer samples are required to reveal their deviation from it; for many, as few as $O(1/\varepsilon)$ suffice. Thus, an algorithm that expects to get, say, $q(n, \varepsilon)^{99}$ samples from an honest-to-goodness distribution from \mathcal{P} , but instead is provided with samples from one that is merely ε -close to it, may break down very quickly. Our corrector, in this very regime of $o(q(n, \varepsilon))$ samples, guarantees this will not happen.

We conclude this section by introducing a relaxation of the notion of sampling corrector, where instead of asking that the unknown distribution be close to the class it is corrected for, we instead decouple the two. For instance, one may require the unknown distribution to be close to a binomial distribution but only correct it to be unimodal. This leads to the following definition of a *nonproper corrector*.

DEFINITION 3.6 (nonproper sampling corrector). *Fix two given properties $\mathcal{P}, \mathcal{P}'$ of distributions on \mathcal{X} . An $(\varepsilon, \varepsilon_1)$ -nonproper sampling corrector for \mathcal{P}' assuming \mathcal{P} is a randomized algorithm which is given parameters $\varepsilon, \varepsilon_1 \in (0, 1]$ such that $\varepsilon_1 \geq \varepsilon$ and $\delta \in [0, 1]$, as well as sampling access to a distribution D . Under the promise that $d_{\text{TV}}(D, \mathcal{P}) \leq \varepsilon$, the algorithm must provide, with probability at least $1 - \delta$ over the samples it draws and its internal randomness, sampling access to a distribution \tilde{D} such that*

- (i) \tilde{D} is close to D : $d_{\text{TV}}(\tilde{D}, D) \leq \varepsilon_1$;
- (ii) \tilde{D} has the (target) property $\tilde{D} \in \mathcal{P}'$.

In other terms, with high probability the algorithm will simulate exactly a sampling oracle for \tilde{D} . The query complexity $q = q(\varepsilon, \varepsilon_1, \delta, \mathcal{X})$ of the algorithm is the number of samples from D it takes per query in the worst case.

Note that if there exists D close to \mathcal{P} such that every $D' \in \mathcal{P}'$ is far from D , this may not be achievable. Hence, the above definition requires that some relation between \mathcal{P} and \mathcal{P}' hold; for instance, that any neighborhood of a distribution from \mathcal{P} intersects \mathcal{P}' . Similarly, we extend this definition to nonproper improvers and batch improvers.

4. A warmup: Nonproper correcting of histograms. To illustrate these ideas, we start with a toy example: nonproper correcting of *regular histograms*. Recall that a distribution D over $[n]$ is said to be a *k-histogram* if its probability mass function is piecewise-constant with at most k “pieces”; that is, if there exists a partition $\mathcal{I} = (I_1, \dots, I_k)$ of $[n]$ into k intervals such that D is constant on each I_j .

Letting \mathcal{H}_k denote the class of all k -histograms over $[n]$, we start with the following question: given samples from a distribution close to \mathcal{H}_k , can we efficiently provide sample access to a corrected distribution $\tilde{D} \in \mathcal{H}_\ell$, for some $\ell = \ell(k, \varepsilon)$? That is, is there a nonproper corrector for \mathcal{H}_ℓ assuming \mathcal{H}_k ?

In this short section, we show how to design such a corrector, under some additional assumption on the min-entropy of the unknown distribution to correct. Namely, we will require the following definition: given some constant $c \geq 1$, we say that a distribution D is *c-regular* if $D(i) \leq \frac{c}{n}$ for all $i \in [n]$, i.e., $\|D\|_\infty \leq \frac{c}{n}$.

PROPOSITION 4.1 (correcting regular histograms). *Fix any constant $c > 0$. For any ε , $\varepsilon_1 \geq 4\varepsilon$ and $\varepsilon_2 = 0$ as in the definition, there exist $\ell = O(k/\varepsilon)$ and a non-proper sampling corrector for \mathcal{H}_ℓ assuming \mathcal{H}_k with sample complexity $O(1)$, under the assumption that the unknown distribution is c -regular.*

Proof. The algorithm works as follows: setting $K \stackrel{\text{def}}{=} \frac{ck}{\varepsilon}$, it first divides the domain into $K \leq L \leq K + 1$ intervals I_1, \dots, I_L of size less than or equal to $\lfloor \frac{n}{K} \rfloor$. Then, the corrected distribution is the flattening \bar{D} of D on these intervals; to output a sample from the L -histogram \bar{D} , the algorithm draws a sample $s \sim D$, checks which of the I_i 's this sample s belongs to, and then outputs s' drawn uniformly from this interval. The sample complexity is clearly constant, as outputting one sample of \bar{D} only requires one from D , and being an L -histogram, $\bar{D} \in \mathcal{H}_\ell$ for $\ell \leq \frac{ck}{\varepsilon} + 1$.

We now turn to proving that $d_{\text{TV}}(D, \bar{D}) \leq 4\varepsilon$. Denote by H the closest k -histogram to D , i.e., $H \in \mathcal{H}_k$ such that $\alpha \stackrel{\text{def}}{=} d_{\text{TV}}(D, H) = d_{\text{TV}}(D, \mathcal{H}_k)$, and let B be the union of the (at most k) intervals among I_1, \dots, I_L where H is not constant. Since D is c -regular, we do have $D(B) \leq k \cdot \frac{c}{n} \cdot \frac{n}{K} = \varepsilon$. Then, since H is α -close to D we get $H(B) \leq \varepsilon + \alpha$.

Now, let \bar{D} (resp. \bar{H}) be the L -histogram obtained by flattening D (resp. H) on I_1, \dots, I_L . By the data processing inequality (Fact 2.1), we obtain

$$d_{\text{TV}}(\bar{D}, \bar{H}) \leq d_{\text{TV}}(D, H).$$

Therefore, by the triangle inequality,

$$d_{\text{TV}}(D, \bar{D}) \leq d_{\text{TV}}(D, H) + d_{\text{TV}}(H, \bar{H}) + d_{\text{TV}}(\bar{H}, \bar{D}) \leq 2d_{\text{TV}}(D, H) + d_{\text{TV}}(H, \bar{H}).$$

Furthermore, as H and \bar{H} can only differ on B , and since the flattening operation preserve the probability weight on each interval of \mathcal{I} , we obtain

$$\begin{aligned} d_{\text{TV}}(H, \bar{H}) &= \frac{1}{2} \|H - \bar{H}\|_1 = \frac{1}{2} \sum_{i \in B} |H(i) - \bar{H}(i)| \leq \frac{1}{2} (H(B) + \bar{H}(B)) \\ &= H(B) \leq \varepsilon + \alpha \end{aligned}$$

which, once plugged back in the previous expression, yields

$$d_{\text{TV}}(D, \bar{D}) \leq 2d_{\text{TV}}(D, H) + \varepsilon + \alpha = 3\alpha + \varepsilon \leq 4\varepsilon$$

since $\alpha \leq \varepsilon$ by assumption. \square

5. Connections to learning and testing. In this section, we draw connections between sampling improvers and other areas, namely testing and learning. These connections shed light on the relation between our model and these other lines of work and provide a way to derive new algorithms and impossibility results for both testing and learning problems. (For the formal definition of the testing and learning notions used in this section, the reader is referred to Appendix B and the relevant subsections.)

5.1. From learning to correcting. As a first observation, it is not difficult to see that, under the assumption that the unknown distribution D belongs to some specific class \mathcal{C} , correcting (or improving) a property \mathcal{P} requires at most as many samples as learning the class \mathcal{C} ; that is, *learning (a class of distributions) is at least as hard as correcting (distributions of this class)*. Here, \mathcal{P} and \mathcal{C} need not be related.

Indeed, assuming there exists a learning algorithm \mathcal{L} for \mathcal{C} , it then suffices to run \mathcal{L} on the unknown distribution $D \in \mathcal{C}$ to learn (with high probability) a hypothesis \hat{D} such that D and \hat{D} are at most at distance $\frac{\varepsilon_1 - \varepsilon}{2}$. In particular, \hat{D} is at most $\frac{\varepsilon_1 + \varepsilon}{2}$ -far from \mathcal{P} . One can then (e.g., by exhaustive search) find a distribution \tilde{D} in \mathcal{P} which is closest to \hat{D} (and therefore at most ε_1 -far from D) and use it to produce as many “corrected samples” as wanted.

THEOREM 5.1. *Let \mathcal{C} a class of probability distributions over \mathcal{X} . Suppose there exists a learning algorithm \mathcal{L} for \mathcal{C} with sample complexity $q_{\mathcal{L}}$. Then, for any property \mathcal{P} of distributions, there exists a (not necessarily computationally efficient) sampling corrector for \mathcal{P} with sample complexity $q(\varepsilon, \varepsilon_1, \delta) = q_{\mathcal{L}}(\frac{\varepsilon_1 - \varepsilon}{2}, \delta)$ under the promise that $D \in \mathcal{C}$.*

Furthermore, if the (efficient) learning algorithm \mathcal{L} has the additional guarantee that its hypothesis class is a subset of \mathcal{P} (i.e., the hypotheses it produces always belong to \mathcal{P}) and that the hypotheses it contains allow efficient generation of samples, then we immediately obtain a computationally efficient sampling corrector; indeed, in this case $\hat{D} \in \mathcal{P}$ already. Furthermore, as mentioned in the introduction, when efficient agnostic proper learning algorithms for distribution classes exist, then there are efficient sampling correctors for the same classes. It is, however, worth pointing out that this correcting-by-learning approach is quite inefficient with regard to the amount of extra randomness needed; indeed, every sample generated from \tilde{D} requires fresh new random bits.

To illustrate this theorem, we give two easy corollaries. The first follows from Chan et al., who showed in [CDSS13] that monotone hazard risk distributions can be learned to accuracy ε using $\tilde{O}(\log n / \varepsilon^4)$ samples; moreover, the hypothesis obtained is a $O(\log(n/\varepsilon)/\varepsilon^2)$ -histogram.

COROLLARY 5.2. *Let \mathcal{C} be the class of monotone hazard risk distributions over $[n]$, and let \mathcal{P} be the property of being a histogram with (at most) \sqrt{n} pieces. Then, under the promise that $D \in \mathcal{C}$ and as long as $\varepsilon = \tilde{\Omega}(1/\sqrt{n})$, there is a sampling corrector for \mathcal{P} with sample complexity $\tilde{O}(\frac{\log n}{(\varepsilon_1 - \varepsilon)^4})$.*

Our next example however demonstrates that this learning approach is not always optimal.

COROLLARY 5.3. *Let \mathcal{C} be the class of monotone distributions over $[n]$, and let \mathcal{P} be the property of being a histogram with (at most) \sqrt{n} pieces. Then, under the promise that $D \in \mathcal{C}$ and as long as $\varepsilon = \tilde{\Omega}(1/\sqrt{n})$, there is a sampling corrector for \mathcal{P} with sample complexity $O(\frac{\log n}{(\varepsilon_1 - \varepsilon)^3})$.*

Indeed, for learning, monotone distributions $\Theta(\log n / \varepsilon^3)$ samples are known to be necessary and sufficient [Bir87]. Yet one can also correct the distribution by simulating samples directly from its flattening on the corresponding Birgé decomposition (as per Definition 2.3), and every sample from this correction-by-simulation costs exactly one sample from the original distribution.

5.2. From correcting to agnostic learning. Let \mathcal{C} and \mathcal{H} be two classes of probability distributions over \mathcal{X} . Recall that a (semi-)agnostic learner for \mathcal{C} (using hypothesis class \mathcal{H}) is a learning algorithm \mathcal{A} which, given sample access to an arbitrary distribution D and parameter ε , outputs a hypothesis $\hat{D} \in \mathcal{H}$ such that, with high probability, \hat{D} does “as much as well as the best approximation from \mathcal{C} ,

$$d_{\text{TV}}(D, \hat{D}) \leq c \cdot \text{OPT}_{\mathcal{C}, D} + O(\varepsilon),$$

where $\text{OPT}_{\mathcal{C},D} \stackrel{\text{def}}{=} \inf_{D_{\mathcal{C}} \in \mathcal{C}} d_{\text{TV}}(D_{\mathcal{C}}, D)$ and $c \geq 1$ is some absolute constant (if $c = 1$, the learner is said to be agnostic).

We first describe how to combine a (nonagnostic) learning algorithm with a sampling corrector in order to obtain an agnostic learner, under the strong assumption that a (rough) estimate of OPT is known. Then, we explain how to get rid of this extra requirement, using machinery from the distribution learning literature (namely, an efficient *hypothesis selection* procedure).

THEOREM 5.4. *Let \mathcal{C} be as above. Suppose there exists a learning algorithm \mathcal{L} for \mathcal{C} with sample complexity $q_{\mathcal{L}}$ and a batch sampling corrector \mathcal{A} for \mathcal{C} with sample complexity $q_{\mathcal{A}}$. Suppose further that a constant-factor estimate $\widehat{\text{OPT}}$ of $\text{OPT}_{\mathcal{C},D}$ is known (up to a multiplicative c).*

Then, there exists a semiagnostic learner for \mathcal{C} with sample complexity $q(\varepsilon, \delta) = q_{\mathcal{A}}(\widehat{\text{OPT}}, \widehat{\text{OPT}} + \varepsilon, q_{\mathcal{L}}(\varepsilon, \frac{\delta}{2}), \frac{\delta}{2})$ (where the constant in front of $\text{OPT}_{\mathcal{C},D}$ is c).

Proof. Let c be the constant such $\text{OPT}_{\mathcal{C},D} \leq \widehat{\text{OPT}} \leq c \cdot \text{OPT}_{\mathcal{C},D}$. The agnostic learner \mathcal{L}' for \mathcal{P} , on input $\varepsilon \in (0, 1]$, works as follows:

- Run \mathcal{A} on D with parameters $(\widehat{\text{OPT}}, \widehat{\text{OPT}} + \varepsilon, \frac{\delta}{2})$ to get $q_{\mathcal{L}}(\varepsilon, \frac{\delta}{2})$ samples distributed according to some distribution \tilde{D} .
- Run \mathcal{L} on these samples, with parameters $\varepsilon, \frac{\delta}{2}$, and output its hypothesis \hat{D} .

We hereafter condition on both algorithms succeeding (which, by a union bound, happens with probability at least $1 - \delta$). Since D is $\widehat{\text{OPT}}$ -close to \mathcal{C} , and therefore by correctness of the sampling corrector we have both $\tilde{D} \in \mathcal{C}$ and $d_{\text{TV}}(D, \tilde{D}) \leq \widehat{\text{OPT}} + \varepsilon$. Hence, the output \hat{D} of the learning algorithm satisfies $d_{\text{TV}}(\tilde{D}, \hat{D}) \leq \varepsilon$, which implies

$$(3) \quad d_{\text{TV}}(D, \hat{D}) \leq \widehat{\text{OPT}} + 2\varepsilon \leq c \cdot \text{OPT}_{\mathcal{C},D} + 2\varepsilon$$

for some absolute constant c , as claimed (using the assumption on $\widehat{\text{OPT}}$). \square

It is worth noting that in the case the learning algorithm is *proper* (meaning the hypotheses it outputs belong to the target class \mathcal{C} ; that is, $\mathcal{H} \subseteq \mathcal{C}$), then so is the agnostic learner obtained with Theorem 5.4. This turns out to be a very strong guarantee; specifically, getting (computationally efficient) proper agnostic learning algorithms remains a challenge for many classes of interest—see, e.g., [DDS12], which mentions efficient proper learning of PBDs as an open problem.

We stress that the above can be viewed as a *generic* framework to obtain efficient agnostic learning results from known efficient learning algorithms. For the sake of illustration, let us consider the simple case of binomial distributions: it is known, for instance as a consequence of the aforementioned results on PBDs, that learning such distributions can be performed with $\tilde{O}(1/\varepsilon^2)$ samples (and that $\Omega(1/\varepsilon^2)$ are required). Our theorem then provides a simple way to obtain agnostic learning of binomial distributions with sample complexity $\tilde{O}(1/\varepsilon^2)$, namely, by designing an efficient sampling corrector for this class with sample complexity $\text{poly}(\log \frac{1}{\varepsilon}, \log \frac{1}{\varepsilon_1})$.

COROLLARY 5.5. *Suppose there exists a batch sampling corrector \mathcal{A} for the class \mathcal{B} of binomial distributions over $[n]$, with sample complexity*

$$q_{\mathcal{A}}(\varepsilon, \varepsilon_1, m, \delta) = \text{polylog} \left(\frac{1}{\varepsilon}, \frac{1}{\varepsilon_1}, m, \frac{1}{\delta} \right).$$

Then, there exists a semiagnostic learner for \mathcal{B} , which, given access to an unknown distribution D promised to be ε -close to some binomial distribution, takes $\tilde{O}(\frac{1}{\varepsilon^2})$ samples

from D and outputs a distribution $\hat{B} \in \mathcal{B}$ such that

$$d_{\text{TV}}(D, \hat{B}) \leq 3\varepsilon$$

with probability at least $2/3$.

To the best of our knowledge, an agnostic learning algorithm for the class of binomial distributions with sample complexity $\tilde{O}(1/\varepsilon^2)$ is not explicitly known, although the results of [CDSS14] do imply a $\tilde{O}(1/\varepsilon^3)$ upper bound and a modification of [DDS12] (to make their algorithm agnostic) seems to yield one. The above suggests an approach which would lead to the (essentially optimal) sample complexity. (Since publication of our work, we have learned that [ADLS15] provides such a result unconditionally.)

5.2.1. Removing the assumption on knowing $\widehat{\text{OPT}}$. In the absence of such an estimate $\widehat{\text{OPT}}$ within a constant factor of $\text{OPT}_{\mathcal{C},D}$ given as input, one can apply the following strategy, inspired by [CDSX14, Theorem 6]. In the first stage, we try to repeatedly “guess” a good $\widehat{\text{OPT}}$ and run the agnostic learner of Theorem 5.4 with this value to obtain a hypothesis. After this stage, we have generated a succinct list \mathcal{H} of hypotheses, one for each $\widehat{\text{OPT}}$ that we tried; the second stage is then to run a hypothesis selection procedure to pick the best $h \in \mathcal{H}$; as long as one of the guesses was good, this h will be an accurate hypothesis.

More precisely, suppose we run the agnostic learner of Theorem 5.4 a total of $\log(1/\varepsilon)$ times, setting at the k^{th} iteration $\widehat{\text{OPT}}_k \stackrel{\text{def}}{=} 2^k\varepsilon$ and $\delta' \stackrel{\text{def}}{=} \delta/(2\log(1/\varepsilon))$. For the first k such that $2^{k-1}\varepsilon \leq \text{OPT}_{\mathcal{C},D} < 2^k\varepsilon$, $\widehat{\text{OPT}}_k$ is in $[\text{OPT}_{\mathcal{C},D}, 2 \cdot \text{OPT}_{\mathcal{C},D}]$. Therefore, by a union bound on all runs of the learner at least one of the hypotheses \hat{D}_k will have the agnostic learning guarantee we want to achieve, i.e., will satisfy (3), with $c = 2$.

Conditioned on this being the case, it remains to determine *which* hypothesis achieves the guarantee of being $(2\text{OPT} + O(\varepsilon))$ -close to the distribution D . This is where we apply a hypothesis selection algorithm—a “tournament” procedure, as in, e.g., [DL01, DDS12, DK14, AJOS14]—to our $N = \log(1/\varepsilon)$ candidates, with accuracy parameter ε and failure probability $\delta/2$. This algorithm has the following guarantee.

PROPOSITION 5.6 ([DL01, Chapter 7]). *There exists a procedure `TOURNAMENT` that, given sample access to an unknown distribution D and both sample and evaluation access to N hypotheses H_1, \dots, H_N , has the following behavior. `TOURNAMENT` makes a total of $O(\log N \log(1/\delta)/\varepsilon^2)$ queries to D, H_1, \dots, H_N , runs in time $O(N^2)$, and outputs a hypothesis H_i such that, with probability at least $1 - \delta$,*

$$d_{\text{TV}}(D, H_i) \leq 3 \min_{j \in [N]} d_{\text{TV}}(D, H_j) + O(\varepsilon).$$

We note that the quadratic running time can be brought down to near-linear, as shown in [DK14, AJOS14] (with the same sample complexity, although at the price of a worse semiagnostic constant). This improved running time, however, is not crucial for our applications.

Summary. Using this result in the approach outlined above, we get with probability at least $1 - \delta$ that we will obtain a hypothesis \hat{D}_{k^*} doing “almost as well as the best D_k ”; that is,

$$d_{\text{TV}}(D, \hat{D}_{k^*}) \leq 6 \cdot \text{OPT}_{\mathcal{C},D} + O(\varepsilon).$$

The overall sample complexity is

$$\sum_{k=1}^{\log(1/\varepsilon)} q_{\mathcal{A}}\left(2^k\varepsilon, (2^k + 1)\varepsilon, q_{\mathcal{L}}\left(\varepsilon, \frac{\delta}{4\log(1/\varepsilon)}\right), \frac{\delta}{4\log(1/\varepsilon)}\right) + \tilde{O}\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right),$$

where the first term comes from the $\log(1/\varepsilon)$ runs of the learner from Theorem 5.4, and the second is the overhead due to the hypothesis selection tournament.

5.3. From correcting to tolerant testing. We observe that the existence of sampling correctors for a given property \mathcal{P} , along with an efficient distance estimation procedure, allows one to convert any distribution testing algorithm into a tolerant distribution testing algorithm. This is similar to the connection between “local reconstructors” and tolerant testing of graphs described in [Bra08, Theorem 3.1] and [CGR13, Theorem 3.1]. That is, if a property \mathcal{P} has both a distance estimator and a sampling corrector, then one can perform *tolerant* testing of \mathcal{P} in the time required to generate enough corrected samples for both the estimator and a (nontolerant) tester.

We first state our theorem in all generality, before instantiating it in several corollaries. For the sake of clarity, the reader may wish to focus on these on a first pass.

THEOREM 5.7. *Let \mathcal{C} be a class of distributions and $\mathcal{P} \subseteq \mathcal{C}$ a property. Suppose there exist an $(\varepsilon, \varepsilon_1)$ -batch sampling corrector \mathcal{A} for \mathcal{P} with complexity $q_{\mathcal{A}}$ and a distance estimator \mathcal{E} for \mathcal{C} with complexity $q_{\mathcal{E}}$ —that is, given sample access to $D_1, D_2 \in \mathcal{C}$ and parameters ε, δ , \mathcal{E} draws $q_{\mathcal{E}}(\varepsilon, \delta)$ samples from D_1, D_2 and outputs a value \hat{d} such that $|\hat{d} - d_{\text{TV}}(D_1, D_2)| \leq \varepsilon$ with probability at least $1 - \delta$.*

Then, from any property tester \mathcal{T} for \mathcal{P} with sample complexity $q_{\mathcal{T}}$, one can get a tolerant tester \mathcal{T}' with query complexity

$$q(\varepsilon', \varepsilon, \delta) = q_{\mathcal{A}}\left(\varepsilon', \Theta(\varepsilon), q_{\mathcal{E}}\left(\frac{\varepsilon - \varepsilon'}{4}, \frac{\delta}{3}\right) + q_{\mathcal{T}}\left(\frac{\varepsilon - \varepsilon'}{4}, \frac{\delta}{3}\right), \frac{\delta}{3}\right).$$

Proof. The tolerant tester \mathcal{T}' for \mathcal{P} , on input $0 \leq \varepsilon' < \varepsilon \leq 1$, works as follows, setting $\beta \stackrel{\text{def}}{=} \frac{\varepsilon - \varepsilon'}{4}$ and $\varepsilon_1 \stackrel{\text{def}}{=} \varepsilon' + \beta$: run \mathcal{A} on D with parameters $(\varepsilon', \varepsilon_1, \delta/3)$ to get $q_{\mathcal{E}}(\beta, \delta/3) + q_{\mathcal{T}}(\beta, \delta/3)$ samples distributed according to some distribution \tilde{D} . Using these samples,

1. estimate $d_{\text{TV}}(D, \tilde{D})$ to within an additive β , and REJECT if this estimate is more than $\varepsilon_1 + \beta = \frac{\varepsilon + \varepsilon'}{2}$;
2. otherwise, run \mathcal{T} on \tilde{D} with parameter β , and accept if and only if \mathcal{T} outputs ACCEPT.

We hereafter condition on all algorithms succeeding (which, by a union bound, happens with probability at least $1 - \delta$).

If D is ε' -close to \mathcal{P} , then the corrector ensures that \tilde{D} is ε_1 -close to D , so the estimate of $d_{\text{TV}}(D, \tilde{D})$ is at most $\varepsilon_1 + \beta$; Step 1 thus passes, and as $\tilde{D} \in \mathcal{P}$ the tester outputs ACCEPT in Step 2.

On the other hand, if D is ε -far from \mathcal{P} , then either (a) $d_{\text{TV}}(D, \tilde{D}) > \varepsilon_1 + 2\beta$ (in which case we output REJECT in Step 1, since the estimate exceeds $\varepsilon_1 + \beta$), or (b) $d_{\text{TV}}(\tilde{D}, \mathcal{P}) > \varepsilon - (\varepsilon_1 + 2\beta) = \beta$, in which case \mathcal{T} outputs REJECT in Step 2. \square

Remark 5.8. Only asking that the distance estimation procedure \mathcal{E} be specific to the class \mathcal{C} is not innocent; indeed, it is known ([VV11]) that for *general* distributions, distance estimation has sample complexity $n^{1-o(1)}$. However, the task becomes

significantly easier for certain classes of distributions and, for instance, can be performed with only $\tilde{O}(k \log n)$ samples, if the distributions are guaranteed to be k -modal [DDS+13]. This observation can be leveraged in cases when one knows that the distribution has a specific property but it does not quite satisfy a second property, e.g., is known to be k -modal but not known to be, say, log-concave.

The reduction above can be useful both as a black-box way to derive upper bounds for tolerant testing, as well as to prove lower bounds for either testing or distance estimation. For the first use, we give two applications of our theorem to provide tolerant monotonicity testers for k -modal distributions. The first is a conditional result, showing that the existence of good monotonicity correctors yield tolerant testers. The second, while unconditional, only guarantees a weaker form of tolerance (guaranteeing acceptance only of distributions that are very close to monotone); and relies on a corrector we describe in section 6.2. As we detail shortly after stating these two results, even this weak tolerance improves upon the one provided by currently known testing algorithms.

COROLLARY 5.9. *Suppose there exists an $(\varepsilon, \varepsilon_1)$ -batch sampling corrector for monotonicity with complexity q . Then, for any $k = O(\log n / \log \log n)$, there exists an algorithm that distinguishes whether a k -modal distribution is (a) ε -close to monotone or (b) 5ε -far from monotone with success probability $2/3$ and sample complexity*

$$q\left(\varepsilon, 2\varepsilon, C \frac{k \log n}{\varepsilon^4 \log \log n}, \frac{1}{9}\right),$$

where C is an absolute constant.

Proof. We combine the distance estimator of [DDS+13] with the monotonicity tester of [DDS14, section 3.4], which both apply to the class of k -modal distributions. As their respective sample complexity is, for distance parameter α and failure probability δ , $O\left(\left(\frac{k^2}{\alpha^4} + \frac{k \log n}{\alpha^4 \log(k \log n)}\right) \log \frac{1}{\delta}\right)$ and $O\left(\frac{k}{\alpha^2} \log \frac{1}{\delta}\right)$, the choice of parameters ($\delta = 1/3$, ε and 5ε) and the assumption on k yield

$$O\left(\frac{k}{\varepsilon^2}\right) + O\left(\frac{k^2}{\varepsilon^4} + \frac{k \log n}{\varepsilon^4 \log(k \log n)}\right) = O\left(\frac{k \log n}{\varepsilon^4 \log(k \log n)}\right),$$

and we obtain by Theorem 5.7 a tolerant tester with sample complexity

$$q\left(\varepsilon, 2\varepsilon, O\left(\frac{k \log n}{\varepsilon^4 \log(k \log n)}\right), \frac{1}{9}\right),$$

as claimed. □

Another application of this theorem, but this time taking advantage of a result from section 6.1, allows us to derive an *explicit* tolerant tester for monotonicity of k -modal distributions.

COROLLARY 5.10. *For any $k \geq 1$, there exists an algorithm that distinguishes whether a k -modal distribution is (a) $O(\varepsilon^3 / \log^2 n)$ -close to monotone or (b) ε -far from monotone with success probability $2/3$ and sample complexity*

$$O\left(\frac{1}{\varepsilon^4} \frac{k \log n}{\log(k \log n)} + \frac{k^2}{\varepsilon^4}\right).$$

In particular, for $k = O(\log n / \log \log n)$ this yields a (weakly) tolerant tester with sample complexity $O\left(\frac{1}{\varepsilon^4} \frac{k \log n}{\log \log n}\right)$.

Proof. We again use the distance estimator of [DDS+13] and the monotonicity tester of [DDS14], which both apply to the class of k -modal distributions, this time with the monotonicity corrector we describe in Corollary 6.5, which works for any ε_1 and $\varepsilon = O(\varepsilon_1^3/\log^2 n)$ and has constant-rate sample complexity (that is, it takes $O(q)$ samples from the original distribution to output q samples). Similarly to Corollary 5.9, the sample complexity is a straightforward application of Theorem 5.7. \square

Note that, to the best of our knowledge, no tolerant tester for monotonicity of k -modal distributions was previously known, though using the (regular) $O(k/\varepsilon^2)$ -sample tester of [DDS14] and standard arguments, one can achieve a weak tolerance on the order of $O(\varepsilon^2/k)$. While the sample complexity obtained in Corollary 5.10 is worse by a polylog(n) factor, it has better tolerance for $k = \Omega(\log^2 n/\varepsilon)$.

6. Sample complexity of correcting monotonicity. In this section, we focus on the sample complexity aspect of correcting, considering the specific example of monotonicity correction. As a first result, we show in section 6.1 how to design a simple batch corrector for monotonicity which, after a preprocessing step costing logarithmically many samples, is able to answer an arbitrary number of queries. This corrector follows the “learning approach” described in section 5.1 and in particular provides a very efficient way to amortize the cost of making many queries to a corrected distribution.

A natural question is then whether one can “beat” this approach and correct the distribution without approximating it as a whole beforehand. Section 6.2 answers it by the affirmative; namely, we show that one can correct distributions that are guaranteed to be $(1/\log^2 n)$ -close to monotone in a completely *oblivious* fashion, with a nonadaptive approach that does not require us to learn anything about the distribution.

Finally, we give in section 6.3 a corrector for monotonicity with no restriction on the range of parameters but assuming a stronger type of query access to the original distribution. Specifically, our algorithm leverages the ability to make *cdf queries* to the distribution D , in order to generate independent samples from a corrected \tilde{D} . This sampling corrector also outperforms the one from section 6.1, making only $O(\sqrt{\log n})$ queries per sample on expectation.

A parenthesis: Nonproper correcting. We note that it is easy to obtain a nonproper corrector for $k(n, \varepsilon)$ -histograms assuming monotonicity with *constant* sample complexity, for $k(n, \varepsilon) = \Theta(\frac{\log n}{\varepsilon})$. Indeed, this follows from the oblivious Birgé decomposition (see Definition 2.3) we shall be using many times throughout this section, which ensures that flattening a monotone distribution yields a $k(n, \varepsilon)$ -histogram that remains close to the original distribution.

6.1. A natural approach: Correcting by learning. Our first corrector works in a straightforward fashion: it *learns* a good approximation of the distribution to correct, which is also concisely represented. It then uses this approximation to build a sufficiently good monotone distribution M' “offline,” by searching for the closest monotone distribution, which in this case can be achieved via linear programming. Any query made to the corrector is then answered according to the latter distribution, at no additional cost.

LEMMA 6.1 (correcting by learning). *Fix any constant $c > 0$. For any ε , $\varepsilon_1 \geq (3 + c)\varepsilon$ and $\varepsilon_2 = 0$ as in the definition, any type of oracle ORACLE and any number of queries m , there exists a sampling corrector for monotonicity from sampling to ORACLE with sample complexity $O(\log n/\varepsilon^3)$.*

Proof. Consider the Birgé decomposition $\mathcal{I}_\alpha = (I_1, \dots, I_\ell)$ with parameter $\alpha \stackrel{\text{def}}{=} \frac{c\varepsilon}{3}$ which partitions the domain $[n]$ into $O(\frac{\log n}{\varepsilon})$ intervals. By Corollary 2.5 and the learning result of [Bir87], we can learn with $(O(\frac{\log n}{\varepsilon^3}))$ samples a $O(\frac{\log n}{\varepsilon})$ -histogram \bar{D} such that

$$(4) \quad d_{\text{TV}}(D, \bar{D}) \leq 2\varepsilon + \alpha.$$

Also, let M be the closest monotone distribution to D . From (2), we get the following: letting \mathcal{M} denote the set of monotone distributions,

$$(5) \quad d_{\text{TV}}(\bar{D}, \mathcal{M}) = d_{\text{TV}}(\Phi_\alpha(D), \mathcal{M}) \leq d_{\text{TV}}(\Phi_\alpha(D), \Phi_\alpha(M)) \leq d_{\text{TV}}(D, M) \leq \varepsilon,$$

where the first inequality follows from the fact that $\Phi_\varepsilon(M)$ is monotone. Thus, \bar{D} is ε -close to monotone, which implies that \bar{D}' is $(\varepsilon + \alpha)$ -close to monotone. Furthermore, it is easy to see that, without loss of generality, one can assume the closest monotone distribution \bar{D}' to be piecewise constant with relation to the same partition (e.g., using again (2)). It is therefore sufficient to find such a piecewise constant distribution; to do so, consider the following linear program which finds exactly this: a monotone M' , closest to \bar{D}' and piecewise constant on \mathcal{I}_α :

$$\begin{aligned} &\text{minimize } \sum_{j=1}^{\ell} \left| x_j - \frac{\bar{D}'(I_j)}{|I_j|} \right| \cdot |I_j| \\ &\text{subject to } 1 \geq x_1 \geq x_2 \geq \dots \geq x_\ell \geq 0 \\ &\qquad \sum_{j=1}^{\ell} x_j |I_j| = 1. \end{aligned}$$

This linear program has $O(\frac{\log n}{\varepsilon})$ variables, and so it can be solved in time $\text{poly}(\log n, \frac{1}{\varepsilon})$.

After finding a solution $(x_j)_{j \in [\ell]}$ to this linear program,¹⁰ we define the distribution $M': [n] \rightarrow [0, 1]$ as follows: $M'(i) = x_{\text{ind}(i)}$, where $\text{ind}(i)$ is the index of the interval of \mathcal{I}_α which i belongs to. This implies that

$$d_{\text{TV}}(\bar{D}', M') \leq \varepsilon + \alpha,$$

and by the triangle inequality we finally get

$$d_{\text{TV}}(D, M^*) \leq d_{\text{TV}}(D, \bar{D}) + d_{\text{TV}}(\bar{D}, \bar{D}') + d_{\text{TV}}(\bar{D}', M') \leq 3\varepsilon + 3\alpha = (3 + c)\varepsilon. \quad \square$$

6.2. Oblivious correcting of distributions which are very close to monotone. We now turn to our second monotonicity corrector, which achieves constant sample complexity for distributions already $(1/\log^2 n)$ -close to monotone. Note that this is a very strong assumption, as if one draws less than $\log^2 n$ samples one does not expect to see any difference between such a distribution D and its closest monotone distribution. Still, our construction actually yields a stronger guarantee; namely, given *evaluation (query)* access to D , it can answer evaluation queries to the corrected distribution as well. See Remark 6.6 for a more detailed statement.

¹⁰To see why a good solution always exists, consider the closest monotone distribution to \bar{D} and apply Φ_α to it. This distribution satisfies all the constraints.

The high-level idea is to treat the distribution as a k -histogram on the Birgé decomposition (for $k = O(\log n)$), thus “implicitly approximating” it, and to correct this histogram by adding a certain amount of probability weight to every interval, so that each gets slightly more than the next one. By choosing these quantities carefully, this ensures that *any* violation of monotonicity gets corrected in the process, without ever having to find out *where* they actually occur.

We start by stating the general correcting approach for general k -histograms satisfying a certain property (namely, the ratio between two consecutive intervals is constant).

LEMMA 6.2. *Let $\mathcal{I} = (I_1, \dots, I_k)$ be a decomposition of $[n]$ in consecutive intervals such that $|I_{j+1}|/|I_j| = 1 + c$ for all j , and let D be a k -histogram distribution on \mathcal{I} that is ε -close to monotone. Then, there is a monotone distribution \tilde{D} which can be sampled from in constant time given oracle access to D , such that $d_{\text{TV}}(D, \tilde{D}) = O(\varepsilon k^2)$. Further, \tilde{D} is also a k -histogram distribution on \mathcal{I} .*

Proof. We will argue that no interval can have significantly more total weight than the previous one, as it would otherwise contradict the bound on the closeness to monotonicity. This bound on the “jump” between two consecutive intervals enables us to define a new distribution \hat{D} which is a mixture of D with an arithmetically decreasing k -histogram (which only depends on ε and k); it can be shown that for the proper choice of parameters, \hat{D} is now monotone.

We start with the following claim, which leverages the distance to monotonicity in order to give a bound on the total violation between two consecutive intervals of the partition.

Claim 6.3. Let D be a k -histogram distribution on \mathcal{I} that is ε -close to monotone. Then, for any $j \in \{1, \dots, k-1\}$,

$$(6) \quad D(I_{j+1}) \leq (1+c)D(I_j) + \varepsilon(2+c).$$

Proof. First, observe that without loss of generality, one can assume the monotone distribution closest to D to be a k -histogram on \mathcal{I} as well (e.g., by a direct application of Fact 2.1 to the flattening on \mathcal{I} of the monotone distribution closest to D). Assume there exists an index $j \in \{1, \dots, k-1\}$ contradicting (6); then,

$$\frac{D(I_{j+1})}{|I_{j+1}|} > (1+c)\frac{D(I_j)}{|I_j|} \cdot \frac{|I_j|}{|I_{j+1}|} + \varepsilon \frac{2+c}{|I_{j+1}|} = \frac{D(I_j)}{|I_j|} + \varepsilon \frac{2+c}{|I_{j+1}|}.$$

But any monotone distribution M which is a k -histogram on \mathcal{I} must satisfy $\frac{M(I_{j+1})}{|I_{j+1}|} \leq \frac{M(I_j)}{|I_j|}$; so that at least $\varepsilon(2+c)$ total weight has to be “redistributed” to fix this violation. Indeed, it is not hard to see¹¹ that the minimum amount of probability weight to “move” in order to do so is at least what is needed to uniformize D on I_j and I_{j+1} . This latter process yields a distribution D' which puts weight $(D(I_j) + D(I_{j+1})) / ((2+c)|I_j|)$ on each element of $I_j \cup I_{j+1}$, and the total variation distance between D and D' (a lower bound on its distance to monotonicity) is then

$$d_{\text{TV}}(D, D') = \frac{D(I_{j+1}) + D(I_j)}{2+c} - D(I_j) = \frac{D(I_{j+1}) - (1+c)D(I_j)}{2+c} > \frac{\varepsilon(2+c)}{2+c} = \varepsilon$$

¹¹E.g., by writing the ℓ_1 cost as the sum of the weight added/removed from “outside” the two buckets and the weight moved between the two buckets in order to satisfy the monotonicity condition and minimizing this function.

which is a contradiction. \square

This suggests immediately the following correcting scheme: to output samples according to \tilde{D} , k -histogram on \mathcal{I} defined by

$$\begin{aligned} \tilde{D}(I_k) &= \lambda(D(I_k)), \\ \tilde{D}(I_{k-1}) &= \lambda(D(I_{k-1}) + (2 + c)\varepsilon), \\ &\vdots \\ \tilde{D}(I_{k-j}) &= \lambda(D(I_{k-j}) + j(2 + c)\varepsilon), \end{aligned}$$

that is,

$$\tilde{D}(I_j) = \lambda \left(D(I_j) + \varepsilon \sum_{i=j}^{k-1} \left(1 + \frac{|I_{j+1}|}{|I_j|} \right) \right), \quad 1 \leq j \leq k.$$

where the normalizing factor is $\lambda \stackrel{\text{def}}{=} (1 + \varepsilon(2 + c)\frac{k(k-1)}{2})^{-1}$. As, by claim 6.3, adding weight decreasing by $(2 + c)\varepsilon$ at each step fixes any pair of adjacent intervals whose average weights are not monotone, \tilde{D}/λ is a nonincreasing nonnegative function. The normalization by λ preserving the monotonicity, \tilde{D} is indeed a monotone distribution, as claimed.

It only remains to bound $d_{\text{TV}}(D, \tilde{D})$:

$$\begin{aligned} 2d_{\text{TV}}(D, \tilde{D}) &= \sum_{j=1}^k \left| D(I_j) - \tilde{D}(I_j) \right| = \sum_{j=1}^k \left| (1 - \lambda)D(I_j) - \lambda\varepsilon \sum_{i=j}^{k-1} (2 + c) \right| \\ &\leq (1 - \lambda) \sum_{j=1}^k D(I_j) + \lambda\varepsilon \sum_{j=1}^k \sum_{i=j}^{k-1} (2 + c) = 1 - \frac{1 - \varepsilon(2 + c)\frac{k(k-1)}{2}}{1 + \varepsilon(2 + c)\frac{k(k-1)}{2}}. \end{aligned}$$

Finally, note that \tilde{D} is a mixture of D (with weight λ) and an explicit arithmetically nonincreasing distribution; sampling from \tilde{D} is thus straightforward and needs at most one sample from D for each draw. \square

Remark 6.4. The above scheme can be easily adapted to the case where the ratio between consecutive intervals is not always the same but is instead $|I_{j+1}| / |I_j| = 1 + c_j$ for some known $c_j \in [C_1, C_2]$; the result then depends on the ratio $C_2/C_1 = \Theta(1)$ as well.

As a direct corollary, this describes how to correct distributions which are promised to be (very) close to monotone, in a completely *oblivious* fashion; that is, the behavior of the corrector does not depend on what the input distribution is; furthermore, the probability of failure is null (i.e., $\delta = 0$).

COROLLARY 6.5 (oblivious correcting of monotonicity). *For every $\varepsilon' \in (0, 1)$, there exists an (oblivious) sampling corrector for monotonicity, with parameters $\varepsilon = O(\varepsilon'^3 / \log^2 n)$, $\varepsilon_1 = \varepsilon'$ and sample complexity $O(1)$.*

Proof. We will apply Lemma 6.2 for $k = O(\log n / \varepsilon')$ and \mathcal{I} being the corresponding Birgé decomposition (with parameter $\varepsilon'/2$). The idea is then to work with the flattening \tilde{D} of D ; since D is ε -close to monotone, it is also $(\varepsilon'/2)$ -close, and \tilde{D} is both

$(\varepsilon'/2)$ -close to D and ε -close to monotone. Applying the correcting scheme with our value of k and c set to ε' , the corrected distribution \tilde{D} is monotone, and

$$d_{\text{TV}}(\bar{D}, \tilde{D}) \leq 1 - \frac{1 - \varepsilon(2 + \varepsilon')^{\frac{k(k-1)}{2}}}{1 + \varepsilon(2 + \varepsilon')^{\frac{k(k-1)}{2}}} \leq \frac{\varepsilon'}{2},$$

where the last inequality derives from the fact that $k^2\varepsilon = O(\varepsilon')$. This in turn implies by a triangle inequality that \tilde{D} is ε' -close to D . Finally, observe that, as stated in the lemma, \tilde{D} can be easily simulated given access to D , using either 0 or 1 draw; indeed, \tilde{D} is a mixture with known weights of an explicit distribution and \bar{D} , and access to the latter can be obtained from D . \square

Remark 6.6. An interesting feature of the above construction is that it does not *only* yield a $O(1)$ -query corrector from sampling to sampling; it similarly implies a corrector from ORACLE to ORACLE with query complexity $O(1)$, for ORACLE being (for instance) an evaluation or cumulative dual oracle (compare Appendix B). This follows from the fact that the corrected distribution \tilde{D} is of the form $\tilde{D} = \lambda D + (1 - \lambda)P$, where both λ and P are fully known.

6.3. Correcting with cumulative dual access. In this section we prove the following result, which shows that correcting monotonicity with $o(\log n)$ queries (on expectation) is possible when one allows a stronger type of access to the original distribution. In particular, recall that in the cumulative dual model (as defined in Appendix B) the algorithm is allowed to make, in addition to the usual draws from the distribution, *evaluation queries* to its cumulative distribution function.¹²

THEOREM 6.7. *For any $\varepsilon \in (0, 1]$, any number of queries m and $\varepsilon_1 = O(\varepsilon)$ as in the definition, there exists a sampling corrector for monotonicity from cumulative dual to SAMP with expected sample complexity $O(\sqrt{m \log n / \varepsilon})$.*

In particular, since learning distributions in the cumulative dual model are easily seen to have query complexity $\Theta(\log n / \varepsilon)$ (e.g., by considering the lower bound instance of [Bir87]), the above corrector beats the learning approach as long as $m = o(\log n / \varepsilon)$.

Remark 6.8. One may look at this ability to correct up to $o(\log n / \varepsilon)$ samples cautiously, as it is well known that the lower bound for testing monotonicity of distributions is $\Omega(\sqrt{n} / \varepsilon^2)$ already [BKR04]. However, this lower bound only establishes a *worst-case* indistinguishability as pointed out in Remark 3.5, for many “typical” distributions that are ε -close to monotone, as few as $O(1/\varepsilon)$ samples would be sufficient to detect the discrepancy from monotone (and compromise the correctness of any algorithm relying on these uncorrected samples).

6.3.1. Overview and discussion. A natural idea would be to first group the elements into consecutive intervals (the “buckets”) and correct this distribution (now a histogram over these buckets) at two levels. That is, start by correcting it optimally at a coarse level (the “superbuckets,” each of them being a group of consecutive buckets); then, every time a sample has to be generated, draw a superbucket from this coarse distribution and correct at a finer level *inside this superbucket* before outputting a sample from the corrected local distribution (i.e., conditional on the superbucket that was drawn and corrected). While this approach seems tantalizing, the main difficulty

¹²We remark that our algorithm will in fact only use this latter type of access and will not rely on its ability to draw samples from D .

with it lies in the possible boundary violations between superbuckets; that is, even if the average weights of the superbuckets are nonincreasing and the distribution over buckets is nondecreasing inside each superbucket, it might still be the case that there are local violations between adjacent superbuckets (i.e., the boundaries are bad). A simple illustration is the sequence $\langle .5, .1, .3, .1 \rangle$, where the first superbucket is $(.5, .1)$ and the second $(.3, .1)$. The average weight is decreasing, and the sequence is locally decreasing inside each superbucket; yet overall the sequence is not monotone.

Thus, we have to consider three kinds of violations:

- (i) global superbucket violations: the average weight of the superbuckets is not monotone;
- (ii) local bucket violations: the distribution of the buckets inside some superbucket is not monotone;
- (iii) superbucket boundary violations: the probability of the last bucket of a superbucket is lower than the probability of the first bucket of the next superbucket.

The ideas underlying our sampling corrector (which is granted both sampling and cumulative query access to the distribution, as defined in the cumulative dual access model) are quite simple; after reducing via standard techniques the problem to that of correcting a *histogram* supported of logarithmically many intervals (the Birgé decomposition), we group these ℓ intervals in K superbuckets, each containing L consecutive intervals from that histogram (buckets). (As a guiding remark, our overall goal is to output samples from a corrected distribution using $o(\ell)$ queries, as otherwise we would already use enough queries to actually learn the distribution.) This two-level approach will allow us to keep most of the corrected distribution implicit, only figuring out (and paying queries for that) the portions from which we will end up outputting samples.

By performing K queries, we can exactly learn the coarse distribution on superbuckets, and correct it for monotonicity (optimally, e.g., by a linear program ensuring the average weights of the superbuckets are monotone), solving the issues of type (i). In order to fix the boundary violations (iii) on-the-go, the idea is to allocate to each superbucket an extra *budget* of probability weight that *can* be used for these boundary corrections. Importantly, if this budget is not entirely used the sampling process restarts from the beginning with a probability corresponding with the remaining budget. This effectively ends up simulating a distribution where each superbucket was assigned an extra weight matching exactly what was needed for the correction, *without having to figure out all these quantities beforehand* (as this would cost too many queries).

Essentially, each superbucket is selected according to its “potential weight,” which includes both the actual probability weight it has and the extra budget it is allowed to use for corrections. Whenever a superbucket S_i is selected this way, we first perform optimal local corrections of type (ii) on both it and the previous superbucket S_{i-1} , making a cdf query at every boundary point between buckets in order to get the weights of all $2L$ buckets they contain, and then computing the optimal fix; at this point, the distribution is monotone inside S_i (and inside S_{i-1}). After this, we turn to the possible boundary violations of type (iii) between S_{i-1} and S_i , by “pouring” some of the weight from S_i ’s budget to fill “valleys” in the last part of S_{i-1} . Once this water-filling has ended,¹³ we may not have used all of S_i ’s budget (but as we shall

¹³We borrow this graphic analogy with the process of pouring water from [ACCL08], which employs it in a different context (in order to bound the running time of an algorithm by a potential-based argument.).

see we make sure we never run out of it); the remaining portion is thus redistributed to the whole distribution by restarting the sampling process from the beginning with the corresponding probability. Note that as soon as we know the weights of all $2L$ Birgé buckets, no more cdf queries are needed to proceed.

6.3.2. Preliminary steps (preprocessing).

First step: Reducing D to a histogram. Given cumulative dual access (i.e., granting both SAMP and cdf query access) to an unknown distribution D over $[n]$ which is ε -close to monotone, we can simulate cumulative dual access to its Birgé flattening $D^{(1)} \stackrel{\text{def}}{=} \Phi_\varepsilon(D)$, also ε -close to monotone and 3ε -close to D (by Corollary 2.5). For this reason, we hereafter work with $D^{(1)}$ instead of D , as it has the advantage of being an ℓ -histogram for $\ell = O(\log n/\varepsilon)$. Because of this first reduction, it becomes sufficient to perform cdf queries on the buckets (and not the individual elements of $[n]$), which altogether entirely define $D^{(1)}$.

Second step: Global correcting of the superbuckets. By making K cdf queries, we can figure out exactly the quantities $D^{(1)}(S_1), \dots, D^{(1)}(S_K)$. By running a linear program, we can re-weight them to obtain a distribution $D^{(2)}$ such that (a) the averages $\frac{D^{(2)}(S_j)}{|S_j|}$ are nonincreasing, (b) the conditional distributions of $D^{(1)}$ and $D^{(2)}$ on each superbucket are identical ($D^{(2)}_{S_j} = D^{(1)}_{S_j}$ for all $j \in [K]$), and (c) $\sum_j |D^{(2)}(S_j) - D^{(1)}(S_j)|$ is minimized.

Third step: Allocating budgets to superbuckets. For reasons that will become clear in the subsequent “water-filling” step, we want to give each superbucket S_j a budget b_j of extra weight *added to its first bucket* $S_{j,1}$ that can be used for local corrections when needed—if it uses only part of this budget during the local correction, it will need to “give back” the surplus. To do so, define $D^{(3)}$ as the distribution such that the following are true:

- $D^{(3)}(S_j) = \lambda^{(3)}(D^{(2)}(S_j) + b_j)$, $j \in [K]$ (where $b_j \stackrel{\text{def}}{=} D^{(2)}(S_j)/(1 + \varepsilon)$ for $j \in [K]$, and $\lambda^{(3)} \stackrel{\text{def}}{=} (1 + \sum_j b_j)^{-1}$ is a normalization factor). Note that $\sum_j b_j = 1/(1 + \varepsilon) \in [1/2, 1]$, so that $\lambda^{(3)} \in [1, 2]$.
- The conditional distribution on $S_j \setminus S_{j,1}$ satisfies $D^{(3)}_{S_j \setminus S_{j,1}} = D^{(2)}_{S_j \setminus S_{j,1}}$ for all $j \in [K]$.

That is, $D^{(3)}$ is a version of $D^{(2)}$ where each superbucket is reweighted, but “locally” looks the same inside each superbucket *except for the first bucket of each superbucket, which received the additional “budget weight.”* Observe that since the size $|S_j|$ of the superbuckets is multiplicatively increasing by an $(1 + \varepsilon)$ factor (as a consequence of Birgé bucketing), the averages $D^{(3)}(S_j)/|S_j|$ will remain nonincreasing. That is, the average changes by less for big values of j 's than for small values, as the budget is spread over more elements.

Remark 6.9. $D^{(3)}$ is uniquely determined by ε , n and D and can be explicitly computed using K cdf queries.

6.3.3. Sampling steps (correcting while sampling). Before going further, we describe a procedure that will be necessary for our fourth step, as it will be the core subroutine allowing us to perform local corrections *between superbuckets*.

Water-filling. Partition each superbucket S_i into range H_i , M_i , and L_i , where (assuming the buckets in S_i are monotone)

- $m_i = D^{(3)}(S_i)/|S_i|$ is the initial value of the average value of superbucket S_i (this does not change throughout the procedure);
- H_i are the (leftmost) elements whose value is greater than m_i (these elements may move to M_i or stay in H_i);
- M_i are the (middle) elements whose value is equal to m_i (these elements stay in M_i);
- L_i are the (rightmost) elements whose value is less than m_i (these elements may move to M_i or stay in L_i);
- \min_i is the minimum probability value in superbucket S_i (this updates throughout the procedure);
- \max_i is the maximum probability value in superbucket S_i (this updates throughout the procedure).

Let $e_i \stackrel{\text{def}}{=} \sum_{x \in H_i} (p(x) - m_i)$ be the *surplus* (so that if $e_i = 0$ then $H_i = \emptyset$, and the superbucket is said to be *dry*) and $d_i \stackrel{\text{def}}{=} \sum_{x \in L_i} (m_i - p(x))$ be the *deficit* (if $d_i = 0$ then $L_i = \emptyset$, and the superbucket is said to be *full*).

Sampling procedure. Recall that we now start and work with $D^{(3)}$, as obtained in section 6.3.2.

- Draw a superbucket S_{i+1} according to the distribution $D^{(3)}(S_1), D^{(3)}(S_K)$ on $[K]$.
- If $S_{i+1} \neq S_1$ (we did not land in the first superbucket), then do the following:
 - Obtain (via cdf queries, if they were not previously known) the $2L$ values $D^{(3)}(S_{i,j}), D^{(3)}(S_{i+1,j})$ ($j \in [L]$) of the buckets in superbuckets S_i, S_{i+1} .
 - Correct them (separately for each of the two superbuckets) optimally for monotonicity, e.g., via linear programming (if that was not done in a prior stage of sample generation), ignoring the extra budget b_i and b_{i+1} on the first bucket of each superbucket. Compute H_i, M_i, L_i , and $H_{i+1}, M_{i+1}, L_{i+1}$.

Algorithm 1. Procedure *water-fill*.

- 1: Take an infinitesimal amount ∂p from the top of the max, leftmost buckets of H_{i+1} , in superbucket S_{i+1} (this would be from the first bucket and any other buckets that have the same probability).
 - 2: Pour ∂p into superbucket S_i (this would land in the min, rightmost buckets of L_i , in superbucket S_i and spread to the left, to buckets that have the same probability, just like water).
-

Algorithm 2. Procedure *front-fill*.

- 1: **while** the surplus e_{i+1} is greater than the extra budget b_{i+1} allocated in section 6.3.2 **do**
 - 2: Take an infinitesimal amount ∂p from the top of the max, leftmost elements of H_{i+1} , in superbucket S_{i+1} (this would be from the first bucket and any other buckets that have the same probability).
 - 3: Pour ∂p into the very first bucket of the domain, $S_{1,1}$.
 - 4: **end while**
 - 5: **return** the total amount f_i of weight poured into $S_{1,1}$.
-

Algorithm 3. Procedure water-boundary-correction.

Require: Superbucket index $j = i + 1$, with initial weight $D^{(3)}(S_{i+1})$.

- 1: Move weight from the surplus of H_{i+1} into L_i using water-fill until
 - (a) $\max_{i+1} \leq \min_i$; or
 - (b) $L_i = \emptyset$ (S_i is full)—i.e., $\min_i = m_i$; or
 - (c) $H_{i+1} = \emptyset$ (S_{i+1} is dry)—this can only happen if $e_{i+1} < d_i$ \triangleright This should never happen because of the “budget allocation” step.
- 2: Note that the distribution might not yet be monotone on $S_i \cup S_{i+1}$ if one of the last two conditions is reached first. If this is the case, then do further correction:
 - (a) if $L_i = \emptyset$ then do front-fill until $\max_{i+1} \leq \min_i$ (this will happen before $H_{i+1} = \emptyset$)
 - (b) if $H_{i+1} = \emptyset$ then abort and **return** FAIL \triangleright This should never happen because of the “budget allocation” step.
- 3: **return** the list B_1, \dots, B_s of buckets in $T_i \stackrel{\text{def}}{=} L_i \cup S_{i+1}$, along with the weights w_1, \dots, w_s they have from w after the redistribution and the portion ε_i of the budget that was not used and the portion f_i that was moved by front-fill (so that $\lambda^{(3)}\varepsilon_i + f_i + \sum_{t=1}^s w_t = D^{(3)}(S_{i+1})$).

- Call water-boundary-correction on $(i + 1)$ using the extra budget only if S_{i+1} becomes dry and not counting it while trying to satisfy condition (a).¹⁴
- If $S_{i+1} = S_1$ (we landed in the first superbucket), we proceed similarly as per the steps above, except for the water-boundary-correction. That is, we only correct locally S_1 for monotonicity.
- During the execution of water-boundary-correction, the water-filling procedure may have used some of the initial “allocated budget” b_{i+1} to pour into L_i . Let $\varepsilon_i \in [0, b_{i+1}]$ be the amount of the budget remaining (not used).
 - With probability $p_i \stackrel{\text{def}}{=} \lambda^{(3)}\varepsilon_i / D^{(3)}(S_{i+1})$, restart the sampling process from the beginning (this is the “budget redistribution step,” which ensures the correction only uses *what it needs* for each superbucket).
 - With probability $q_i \stackrel{\text{def}}{=} f_i / D^{(3)}(S_{i+1})$, where f_i is the weight moved by the procedure front-fill, output from the very first bucket of the domain.
 - With the remaining probability, output a sample from the new (conditional) distribution on the buckets in $T_i \stackrel{\text{def}}{=} L_i \cup S_{i+1}$. This is the conditional distribution defined on T_i by the weights w_1, \dots, w_s , as returned by water-boundary-correction.

Note that the distribution we output from if we initially select the superbucket S_{i+1} is supported on $L_i \cup S_{i+1}$. Moreover, conditioning on $M_{i+1} \cup L_{i+1}$ we get exactly the conditional distribution $D_{M_{i+1} \cup L_{i+1}}^{(3)}$. (This ensures that from each bucket there

¹⁴At this point, the new distribution $D^{(4)}$ (which is at least partly implicit, as it is only known at a very coarse level over superbuckets and locally for some buckets inside $S_i \cup S_{i+1}$) obtained is monotone over the superbuckets (water-boundary-correction does not violate the invariant that the distribution over superbuckets is monotone), is monotone inside both S_i and S_{i+1} , and furthermore is monotone over $S_i \cup S_{i+1}$. Even more important, the fact that $\min_i \geq \max_{i+1}$ will ensure applying the same process in the future, e.g., to S_{i+2} , will remain consistent with regard to monotonicity.

is a unique superbucket that has to be picked initially for the bucket’s weight to be modified.) Observe that as defined above, buckets from $L_i \subseteq S_i$ can be outputted from either because superbucket S_i was picked or because S_{i+1} was drawn and some of its weight was reassigned to L_i by water-boundary-correction. The probability of outputting any bucket in L_i is then the sum of the probabilities of the two types of events.

6.3.4. Analysis. The first observation is that the distribution of any sample output by the sampling process described above is not only consistent but completely determined by n , ε , and D .

Claim 6.10. The process described in sections 6.3.2 and 6.3.3 uniquely defines a distribution \tilde{D} , which is a function of D , n , and $\varepsilon \in (0, 1)$ only.

Claim 6.11. The expected number of queries necessary to output m samples from \tilde{D} is upper bounded by $K + 4mL\varepsilon$.

Proof. The number of queries for the preliminary stage is K ; after this, generating a sample requires X queries, where X is a random variable satisfying

$$X \leq 2L + RX',$$

where X, X' are independent and identically distributed, and R is a Bernoulli random variable independent of X' and with parameter Δ (itself a random variable depending on X ; Δ takes value p_i when the first draw selects superbucket $i + 1$), corresponding to the probability of restarting the sampling process from the beginning. It follows that

$$\mathbb{E}[X] \leq 2L + \mathbb{E}[R] \mathbb{E}[X] = 2L + \mathbb{E}[\Delta] \mathbb{E}[X].$$

Using the fact that $\mathbb{E}[\Delta] = \sum_{i \in [K]} D^{(3)}(S_{i+1})p_i = \sum_{i \in [K]} D^{(3)}(S_{i+1}) \frac{\lambda^{(3)} \varepsilon_i}{D^{(3)}(S_{i+1})} \leq \lambda^{(3)} \sum_{i \in [K]} b_i \in [1/3, 1/2]$ and rearranging, we get $\mathbb{E}[X] \leq 4L$. \square

LEMMA 6.12. *If D is a distribution on $[n]$ satisfying $d_{TV}(D, \mathcal{M}) \leq \varepsilon$, then the distribution \tilde{D} defined above is monotone.*

Proof. Observe that as the average weights of the superbuckets in $D^{(2)}$ are nonincreasing, the definition of $D^{(3)}$ along with the fact that the lengths of the superbuckets are (multiplicatively) increasing implies that the average weights of the superbuckets in $D^{(3)}$ are also nonincreasing. In more detail, fix $1 \leq i \leq K - 1$; we have

$$\frac{D^{(2)}(S_i)}{|S_i|} \geq \frac{D^{(2)}(S_{i+1})}{(1 + \varepsilon)|S_i|},$$

using the fact that $|S_j| = (1 + \varepsilon)|S_{j-1}|$. From there, we get that

$$(1 + \varepsilon)D^{(2)}(S_i) \geq D^{(2)}(S_{i+1}),$$

or equivalently

$$\begin{aligned} \frac{b_i + D^{(2)}(S_i)}{|S_i|} &= \frac{D^{(2)}(S_i) + (1 + \varepsilon)D^{(2)}(S_i)}{(1 + \varepsilon)|S_i|} \geq \frac{D^{(2)}(S_{i+1}) + (1 + \varepsilon)D^{(2)}(S_{i+1})}{(1 + \varepsilon)^2 |S_i|} \\ &= \frac{b_{i+1} + D^{(2)}(S_{i+1})}{|S_{i+1}|} \end{aligned}$$

showing that before renormalization (and therefore after as well) the average weights of the superbuckets in $D^{(3)}$ are indeed nonincreasing. Rephrased, this means that the sequence of m_i 's, for $i \in [K]$, is monotone.

Moreover, notice that by construction the distribution \tilde{D} is monotone within each superbucket; indeed, it is explicitly made so one superbucket at a time, in the third step of the sampling procedure. After a superbucket has been made monotone this way, it can only be changed by water-filling which by design can never introduce new violations; the weight is always moved to the left, with the values m_i 's acting as boundary conditions to stop the water-filling process and prevent new violations, or moved to the first element of the domain.

It only remains to argue that monotonicity is not violated at the boundary of two consecutive superbuckets. But since the **water-boundary-correction**, if it does not abort, guarantees that the distribution is monotone between consecutive buckets as well (as $m_{i+1} \leq \max_{i+1} \leq \min_i \leq m_i$), it is sufficient to show that **water-boundary-correction** never returns FAIL. This is ensured by the budget allocation step, which by providing H_{i+1} with up to an additional b_{i+1} to spread into L_i guarantees it will become dry. Indeed, if this happened then it would mean that correcting this particular violation (before the budget allocation, which only affects the first elements of the superbuckets) in $D^{(2)}$ required to move more than b_{i+1} weight, contradicting the fact that the average weights of the superbuckets in $D^{(2)}$ were nonincreasing. In more detail, the maximum amount of weight to “pour” in order to fill L_i is in the case where H_{i+1} is empty (i.e., the distribution on S_{i+1} is already uniform) but L_i is (almost) all of S_i (i.e., all the weight in S_i is in the first bucket). To correct this with our water-filling procedure, one would have to pour $|S_i| \cdot \frac{D^{(2)}(S_{i+1})}{|S_{i+1}|} = \frac{D^{(2)}(S_{i+1})}{1+\varepsilon}$ weight in L_i , which is exactly our choice of value for b_{i+1} . \square

LEMMA 6.13. *If D is a distribution on $[n]$ satisfying $d_{\text{TV}}(D, \mathcal{M}) \leq \varepsilon$, then $d_{\text{TV}}(D, \tilde{D}) = O(\varepsilon)$.*

Proof. We will bound separately the distances D to $D^{(1)}$, $D^{(1)}$ to $D^{(2)}$, and $D^{(2)}$ to \tilde{D} and conclude by the triangle inequality.

- First of all, the distance $d_{\text{TV}}(D, D^{(1)})$ is at most 3ε , by properties of the Birgé decomposition (and as $d_{\text{TV}}(D, \mathcal{M}) \leq \varepsilon$).
- We now turn to $d_{\text{TV}}(D^{(1)}, D^{(2)})$, showing that it is at most ε ; in order to do so, we introduce D' , the piecewise-constant distribution obtained by flattening $D^{(1)}$ on each of the K superbuckets (so that $D'(S_j) = D^{(1)}(S_j)$ for all j). It is not hard to see, e.g., by the data processing inequality for total variation distance, that D' is also ε -close to monotone, and additionally that the closest monotone distribution M' can also be assumed to be constant on each superbucket.

Consider now the transformation that reweights in D' each superbucket S_j by a factor $\alpha_j > 0$ to obtain M' ; it is straightforward to see from section 6.3.2 that this transformation maps $D^{(1)}$ to $D^{(2)}$. Therefore,

$$\begin{aligned} 2d_{\text{TV}}(D^{(1)}, D^{(2)}) &= \sum_{j \in [K]} \sum_{x \in S_j} \left| D^{(1)}(x) - D^{(2)}(x) \right| \\ &= \sum_{j \in [K]} \sum_{x \in S_j} \left| D^{(1)}(x) - \alpha_j D^{(1)}(x) \right| \\ &= \sum_{j \in [K]} \sum_{x \in S_j} D^{(1)}(x) \cdot |1 - \alpha_j| = \sum_{j \in [K]} D^{(1)}(S_j) \cdot |1 - \alpha_j| \end{aligned}$$

$$\begin{aligned}
 &= \sum_{j \in [K]} \sum_{x \in S_j} D^{(1)}(x) \cdot |1 - \alpha_j| \\
 &= \sum_{j \in [K]} D^{(1)}(S_j) \cdot \left| 1 - \frac{M'(S_j)}{D'(S_j)} \right| \\
 &= \sum_{j \in [K]} |D'(S_j) - M'(S_j)| = 2d_{\text{TV}}(D', M') \leq 2\varepsilon.
 \end{aligned}$$

- To bound $d_{\text{TV}}(D^{(2)}, \tilde{D})$, first consider the distribution D'' obtained by correcting optimally $D^{(2)}$ for monotonicity *inside each superbucket separately*. That is, D'' is the distribution satisfying monotonicity on each S_j (separately) and $D''(S_j) = D^{(2)}(S_j)$ for each $j \in [K]$ and minimizing

$$\sum_{j \in [K]} \sum_{i \in [L]} \left| D''(S_{j,i}) - D^{(2)}(S_{j,i}) \right|$$

(or, equivalently, minimizing $\sum_{i \in [L]} |D''(S_{j,i}) - D^{(2)}(S_{j,i})|$ for all $j \in [K]$). The first step is to prove that D'' is close to $D^{(2)}$: recall first that by the triangle inequality, our previous argument implies that $D^{(2)}$ is (2ε) -close to monotone. Therefore, the (related) optimization problem asking to find a non-negative function P that minimizes the same objective, but under the different constraints “ P is monotone on $[n]$ and $P([n]) = D^{(2)}([n])$,” has a solution P whose total variation distance from $D^{(2)}$ is at most 2ε .

But P can be used to obtain P' , solution to the original problem, by reweighting each superbucket S_j the following way:

$$P'(x) \stackrel{\text{def}}{=} P(x) \cdot \frac{D^{(2)}(S_j)}{P(S_j)}, \quad x \in S_j.$$

Clearly, P' satisfies the constraints of the first optimization problem; moreover,

$$\begin{aligned}
 2d_{\text{TV}}(P', D^{(2)}) &= \sum_{j \in [K]} \sum_{x \in S_j} \left| P'(x) - D^{(2)}(x) \right| \\
 &= \sum_{j \in [K]} \sum_{x \in S_j} \left| P(x) \frac{D^{(2)}(S_j)}{P(S_j)} - D^{(2)}(x) \right| \\
 &\leq \sum_{j \in [K]} \sum_{x \in S_j} \left| P(x) - D^{(2)}(x) \right| + \sum_{j \in [K]} \sum_{x \in S_j} P(x) \left| \frac{D^{(2)}(S_j)}{P(S_j)} - 1 \right| \\
 &= 2d_{\text{TV}}(P, D^{(2)}) + \sum_{j \in [K]} \left| D^{(2)}(S_j) - P(S_j) \right| \leq 4d_{\text{TV}}(P, D^{(2)}) \\
 &\leq 8\varepsilon,
 \end{aligned}$$

where we used the fact that

$$\begin{aligned}
 \sum_{j \in [K]} \left| D^{(2)}(S_j) - P(S_j) \right| &= \sum_{j \in [K]} \left| \sum_{x \in S_j} (D^{(2)}(x) - P(x)) \right| \\
 &\leq \sum_{j \in [K]} \sum_{x \in S_j} \left| D^{(2)}(x) - P(x) \right|.
 \end{aligned}$$

As $d_{\text{TV}}(P', D^{(2)})$ is an upper bound on the optimal value of the optimization problem, we get $d_{\text{TV}}(D'', D^{(2)}) \leq 4\varepsilon$.

The next and last step is to bound $d_{\text{TV}}(D'', \tilde{D})$, and show that it is $O(\varepsilon)$ as well. To see why this will allow us to conclude, note that D'' is the intermediate distribution that the sampling process we follow would define, if there was neither extra budget allocated nor **water-boundary-correction**. Put differently, \tilde{D} is derived from D'' by adding the “right amount of extra budget $b'_j \in [0, b_j]$ ” to S_j , then pouring it to S_{j-1} by water-filling and front-filling, and normalizing afterwards by $(1 + \sum_{j \in [K]} b'_j)^{-1}$.

Writing \tilde{D}'' for the result of the transformation above before the last renormalization step, we can bound $d_{\text{TV}}(D'', \tilde{D})$ by

$$\begin{aligned} 2d_{\text{TV}}(D'', \tilde{D}) &= \|D'' - \tilde{D}\|_1 \leq \|D'' - \tilde{D}''\|_1 + \|\tilde{D}'' - \tilde{D}\|_1 \\ &\leq \sum_{j \in [K]} b'_j + \sum_{j \in [K]} f_j + \sum_{x \in [n]} \left| \left(1 + \sum_{j \in [K]} b'_j \right) \tilde{D}(x) - \tilde{D}(x) \right| \\ &\leq \sum_{j \in [K]} b'_j + \sum_{j \in [K]} f_j + \left| \left(1 + \sum_{j \in [K]} b'_j \right) - 1 \right| \\ &= 2 \sum_{j \in [K]} b'_j + \sum_{j \in [K]} f_j \end{aligned}$$

where $f_j \geq 0$ is defined as the amount of weight moved from H_j to the first element of the domain during the execution of **water-boundary-correction**, if **front-fill** is called, and the bound on $\|D'' - \tilde{D}''\|_1$ comes from the fact that \tilde{D}'' pointwise dominates D'' and has a total additional $\sum_{j \in [K]} b'_j$ weight.

It then suffices to bound the quantities $\sum_{j \in [K]} f_j$ and $\sum_{j \in [K]} b'_j$, using for this the fact that by the triangle inequality D'' is itself (6ε) -close to monotone. The at most K intervals where D'' violates monotonicity (which are fixed by using the b'_j 's) are disjoint and centered at the boundaries between consecutive superbuckets; i.e., each of them is in a interval $V_j \subseteq L_{j-1} \cup H_j \subsetneq S_{j-1} \cup S_j$. Because of this disjointness, each transformation of D'' into a monotone distribution must add weight in $V_j \cap L_{j-1}$ or subtract some from $V_j \cap H_j$ to remove the corresponding violation. By definition of b'_j (as minimum amount of additional weight to bring to L_{j-1} when spreading weight from H_j to L_{j-1}), this implies that any such transformation has to “pay” at least $b'_j/2$ (in total variation distance) to fix violation V_j . From the bound on $d_{\text{TV}}(D'', \mathcal{M})$, we then get $\sum_{j \in [K]} b'_j \leq 12\varepsilon$. A similar argument shows that $\sum_{j \in [K]} f_j \leq 12\varepsilon$ as well, which in turn yields $d_{\text{TV}}(D'', \tilde{D}) \leq 18\varepsilon$.

- Putting these bounds together, we obtain

$$\begin{aligned} d_{\text{TV}}(D, \tilde{D}) &\leq d_{\text{TV}}(D, D^{(1)}) + d_{\text{TV}}(D^{(1)}, D^{(2)}) + d_{\text{TV}}(D^{(2)}, D'') \\ &\quad + d_{\text{TV}}(D'', \tilde{D}) \\ &\leq 3\varepsilon + \varepsilon + 4\varepsilon + 18\varepsilon = 26\varepsilon. \quad \square \end{aligned}$$

We are finally in the position of proving the main result of the section.

Proof of Theorem 6.7. The theorem follows from Claim 6.10, Claim 6.11, Lemma 6.12, and Lemma 6.13, setting $K = mL = \sqrt{m\ell}$ (where $\ell = O(\log n/\varepsilon)$ as defined in the Birgé decomposition). \square

7. Constrained error models. In the previous sections, no assumption was made on the form of the error, only on the amount. In this section, we suggest a model of errors capturing the deletion of a whole “chunk” of the distribution. We refer to this model as the *missing data model*, where we assume that some ε probability is removed by taking out all the weight of an arbitrary interval $[i, j]$ for $1 \leq i < j \leq n$ and redistributing it on the rest of the domain as per rejection sampling.¹⁵ We show that one can design sampling improvers for monotone distributions with arbitrarily large amounts of error. Hereafter, D will denote the original (monotone) distribution (before the deletion error occurred), and $D' = D^{(i,j)}$ the resulting (faulty) one, to which the sampling improver has access. Our sampling improver follows what could be called the “learning-just-enough” approach: instead of attempting to approximate the *whole* unaltered original distribution, it only tries to learn the values of i, j and then generates samples “on-the-fly.” At a high level, the algorithm works by (i) detecting the location of the missing interval (drawing a large (but still independent of n) number of samples), then (ii) estimating the weight of this interval under the original, unaltered distribution, and finally (iii) filling this gap uniformly by moving the right amount of probability weight from the end of the domain. To perform the first stage, we shall follow a paradigm first appeared in [DDS14] and utilize testing as a subroutine to detect when enough learning has been done.

THEOREM 7.1. *For the class of distributions following the missing data error model, there exists a batch sampling improver MISSING-DATA-IMPROVER for monotonicity that, on input $\varepsilon, q, \delta, \alpha$, achieves parameters $\varepsilon_1 = O(\varepsilon)$ and any $\varepsilon_2 < \varepsilon$, and has sample complexity $\tilde{O}(\frac{1}{\varepsilon_2^3} \log \frac{1}{\delta})$ independent of ε .*

The detailed proof of our approach, as well as the description of MISSING-DATA-IMPROVER, are given in the next subsection.

7.1. Proof of Theorem 7.1. Before describing further the way to implement our three-stage approach, we will need the following lemmata. The first examines the influence of adding or removing probability weight ε from a distribution, as is the case in the missing data model.

LEMMA 7.2. *Let D be a distribution over $[n]$ and $\varepsilon > 0$. Suppose $D' \stackrel{\text{def}}{=} (1 + \varepsilon)D - \varepsilon D_1$ for some distribution D_1 . Then $d_{\text{TV}}(D, D') \leq \varepsilon$.*

The proof follows from a simple application of the triangle inequality to the ℓ_1 distance between D and D' . We note that the same bound applies if $D' = (1 - \varepsilon)D + \varepsilon D_1$.

The next two lemmata show that the distance to monotonicity of distributions falling into this error model can be bounded in terms of the probability weight right after the missing interval.

LEMMA 7.3. *Let D be a monotone distribution and $D' = D^{(i,j)}$ be the faulty distribution. If $D([j + 1, 2j - i + 1]) > \varepsilon$, then D' is $\varepsilon/2$ -far from monotone.*

Proof. Let $L \stackrel{\text{def}}{=} j - i$ be the length of the interval where the deletion occurred. Since the interval $[j + 1, 2j - i + 1]$ has the same length as $[i, j]$ and weight $p > \varepsilon$,

¹⁵ That is, if D was the original distribution, the faulty one $D^{(i,j)}$ is formally defined as $(1 + \varepsilon)\mathbb{1}_{[n] \setminus [i,j]} \cdot D - \varepsilon \cdot \mathcal{U}_{[i,j]}$, where $\varepsilon = D([i, j])$.

the average weight of an element is at least $\frac{\varepsilon}{D}$. Every monotone distribution M should also be monotone on the interval $[i, 2j - i + 1]$; therefore, one must have $M([i, j]) \geq M([j + 1, 2j - i + 1])$. Let $q \stackrel{\text{def}}{=} M([i, j])$. As $D'([i, j]) = 0$, we get that $2d_{\text{TV}}(D', \hat{D}) \geq q$. On one hand, if $q < p$, then at least $q - p$ weight must have been “removed” from $[i, 2j - i + 1]$ to achieve monotonicity, and all together $2d_{\text{TV}}(D', M) \geq q + (p - q) = p$. On the other hand, if $q \geq p$ we directly get $2d_{\text{TV}}(D', M) \geq q \geq p$. In both cases,

$$d_{\text{TV}}(D', M) \geq p/2 \geq \varepsilon/2,$$

and D' is $\varepsilon/2$ -far from monotone. \square

LEMMA 7.4. *Let D be a monotone distribution and $D' = D^{(i,j)}$ as above. If $D'([j + 1, 2j - i + 1]) < \varepsilon/2$, then D' is ε -close to monotone.*

Proof. We will constructively define a monotone distribution M which will be ε -close to D' . Let $p \stackrel{\text{def}}{=} D'([j + 1, 2j - i + 1]) < \varepsilon/2$. According to the missing data model, D' should be monotone on the intervals $[1, i - 1]$ and $[j + 1, n]$. In particular, the probability weight of the last element of $[j + 1, 2j - i + 1]$ should be below the average weight of the interval; i.e., for all $k \geq 2j - i + 1$ one has $D'(k) \leq D'(2j - i + 1) < \frac{p}{j - i + 1}$.

So, if we let the distribution M (that we are constructing) be uniform on the interval $[j + 1, 2j - i + 1]$ and have also total weight p there, monotonicity will not be violated at the right endpoint of the interval; and the ℓ_1 distance between D' and M in that interval will be at most $2p$. “Taking” another p probability weight from the very end of the domain and moving it to the interval $[i, j]$ (where it is then uniformly spread) to finish the construction of M adds at most another $2p$ to the ℓ_1 distance. Therefore, $2d_{\text{TV}}(D', M) \leq 2p + 2p < 2\varepsilon$, and M is monotone as claimed. \square

The sampling improver is described in Algorithm 4.

Implementing (i): Detecting the gap.

LEMMA 7.5 (Lemma (i)). *There exists an algorithm that, on input $\alpha \in (0, 1/3)$ and $\delta \in (0, 1)$, takes $\tilde{O}(\frac{1}{\alpha^6} \log \frac{1}{\delta})$ samples from $D' = D^{i,j}$ and outputs either two elements $a, b \in [n]$ or close such that the following holds. With probability at least $1 - \delta$,*

- if it outputs elements a, b , then (a) $[i, j] \subseteq [a, b]$ and (b) $D'([a, b]) \leq 3\alpha^2$;
- if it outputs close, then D' is α^2 -close to monotone.

Proof. Inspired by techniques from [DDS14], we first partition the domain into $t = O(1/\alpha^2)$ intervals I_1, \dots, I_t of roughly equal weight as follows. By taking $O(\frac{1}{\alpha^6} \log \frac{1}{\delta})$ samples, the DKW inequality ensures that with probability at least $1 - \delta/2$ we obtain an approximation \hat{D} of D' , close up to $\alpha^3/5$ in Kolmogorov distance. We hereafter assume this holds. For our partitioning to succeed, we first have to take care of the “big elements,” which by assumption on D' (which originates from a monotone distribution) must all be at the beginning. In more detail, let

$$r \stackrel{\text{def}}{=} \max \left\{ x \in [n] : \hat{D}(x) \geq \frac{4\alpha^3}{5} \right\}$$

and $B \stackrel{\text{def}}{=} \{1, \dots, r\}$ be the set of potentially big elements. Note that if $D'(x) \geq \alpha^3$, then necessarily $x \in B$. This leaves us with two cases, depending on whether the missing data interval is amidst the big elements or in the tail part of the support.

Algorithm 4. MISSING-DATA-IMPROVER.

Require: $\varepsilon, \varepsilon_2 < \varepsilon, \delta \in (0, 1)$ and $q \geq 1$, sample access to D' .

```

1: Start ▷ PREPROCESSING
2:   Draw  $m \stackrel{\text{def}}{=} \tilde{\Theta}\left(\frac{1}{\varepsilon_2} \log \frac{1}{\delta}\right)$  samples from  $D' = D^{i,j}$ .
3:   Run the algorithm of Lemma 7.5 on them to get an estimate  $(a, b)$  of the
   unknown  $(i, j)$  or the value close.
4:   Run the algorithm of Lemma 7.6 on them to get an estimate  $\gamma$  of  $D'([b +$ 
    $1, 2b - a + 1])$  and values  $c, \gamma'$  such that  $|D'([c, n]) - \gamma'| \leq \varepsilon_2^{3/2}$ .
5: End
6: Start ▷ GENERATING
7:   for  $i$  from 1 to  $q$  do
8:     Draw  $s_i$  from  $D'$ .
9:     if the second step of PREPROCESSING returned close or  $\gamma < 5\varepsilon_2^{3/2}$  then
10:      return  $s_i$  ▷ The distribution is already  $\varepsilon_2$ -close to monotone; do not
      change it.
11:    end if
12:    if  $s_i \in [c, n]$  then ▷ Move  $\gamma$  weight from the end to  $[a, b]$ 
13:      With probability  $\gamma/\gamma'$ , return a uniform sample from  $[a, b]$ 
14:      Otherwise, return  $s_i$ 
15:    else if  $s_i \in [b + 1, 2b - a + 1]$  then
16:      return a uniform sample from  $[b + 1, 2b - a + 1]$ 
17:    else
18:      return  $s_i$  ▷ Do not change the part of  $D'$  that need not be changed.
19:    end if
20:  end for
21: End

```

- If $[i, j] \subseteq B$: it is then straightforward to *exactly* find i, j and output them as a, b . Indeed all elements $x \in B$ have, by monotonicity, either $D'(x) \geq D'(r) \geq \frac{3\alpha^3}{5}$ or $D'(x) = 0$ (the latter if and only if $x \in [i, j]$). Thus, one can distinguish between $x \in [i, j]$ (for which $\hat{D}(x) \leq \alpha^3/5$) and $x \notin [i, j]$ (in which case $\hat{D}(x) \geq 2\alpha^3/5$).
- If $[i, j] \not\subseteq B$: then, as $r \notin [i, j]$ (since $D'(r) > 0$), it must be the case that $[i, j] \subseteq \bar{B} = \{r + 1, \dots, n\}$. Moreover, every point $x \in \bar{B}$ is “light”: $D'(x) < \alpha^3$ and $\hat{D}(x) < \frac{4\alpha^3}{5}$. We iteratively define $I_1, \dots, I_t \subseteq \bar{B}$, where $I_i = [r_i + 1, r_{i+1}]$: $r_1 \stackrel{\text{def}}{=} r + 1, r_{t+1} \stackrel{\text{def}}{=} n$, and for $1 \leq i \leq t - 1$

$$r_{i+1} \stackrel{\text{def}}{=} \min \left\{ s > r_i : \hat{D}([r_i + 1, x]) \geq \alpha^2 \right\} .$$

This guarantees that, for all $i \in [t]$, $D'(I_i) \in [\alpha^2 - \frac{2\alpha^3}{5}, \alpha^2 + \frac{4\alpha^3}{5} + \frac{2\alpha^3}{5}] \subset [\alpha^2 - \frac{3\alpha^3}{2}, \alpha^2 + \frac{3\alpha^3}{2}]$. (And in turn that $t = O(1/\alpha^2)$ as claimed.) Observing that the definition of the missing data error model implies D' is 2-modal, we can now use the monotonicity tester of [DDS14, section 3.4]. This algorithm takes only $O(\frac{k}{\varepsilon} \log \frac{1}{\delta})$ samples (crucially, no dependence on n) to distinguish with probability at least $1 - \delta$ whether a k -modal distribution is monotone versus ε -far from it.

We iteratively apply this tester with parameters $k = 2, \varepsilon = \alpha^2/4$, and $\delta' = O(\delta/t)$ to each of the at most t prefixes of the form $P_\ell \stackrel{\text{def}}{=} \cup_{i=1}^\ell I_i$; a union

bound ensures that with probability at least $1 - \delta/2$ all tests are correct. Conditioning on this, we are able to detect the first interval I_{ℓ^*} which either contains or falls after j (if no such interval is found, then the input distribution is already α^2 -close to monotone, and we output `close`).

In more detail, suppose first no run of the tester rejects (so that `close` is outputted). Then, by Lemma 7.3, we must have $D([j+1, 2j-i+1]) \leq 2 \cdot \alpha^2/4 = \alpha^2/2$, and Lemma 7.4 guarantees D' is then α^2 -close to monotone. Suppose now that it rejects on some prefix P_{ℓ^*} (and accepted for all $\ell < \ell^*$). As D' is nonincreasing on $[1, j]$, we must have $[i, j] \subset P_{\ell^*}$. Moreover, the tester will by Lemma 7.3 reject as soon as an interval $[j+1, s] \subseteq [j+1, 2j-i+1]$ of weight $\alpha^2/2$ is added to the current prefix. This implies, as each I_ℓ has weight at least $\alpha^2/2$, that $[i, j] \subseteq I_{\ell^*-1} \cup \ell^* = [a, b]$.

Finally, observe that the above can be performed with $O(\frac{1}{\alpha^2} \cdot \frac{1}{\alpha^4} \cdot \log t) = \tilde{O}(\frac{1}{\alpha^6} \log \frac{1}{\delta})$ samples, as claimed (where the first $1/\alpha^2$ factor comes from doing rejection sampling to run the tester with domain P_ℓ only, which by construction is guaranteed to have weight $\Omega(1/\alpha^2)$). The overall probability of failure is at most $\delta/2 + \delta/2 = \delta$, as claimed. \square

Implementing (ii): estimating the missing weight. Conditioning on the output a, b of Lemma 7.5 being correct, the next lemma explains how to get a good estimate of the total weight we should *put back* in $[a, b]$ in order to fix the deletion error.

LEMMA 7.6. *Given D' , α as above, $\delta \in (0, 1)$, and a, b such that $[i, j] \subseteq [a, b]$ and $D'([a, b]) \leq 3\alpha^2$, there exists an algorithm which takes $O(\frac{1}{\alpha^6} \log \frac{1}{\delta})$ samples from D' and outputs values γ, γ' and c such that the following holds with probability at least $1 - \delta$:*

- (i) $|D'([b+1, 2b-a+1]) - \gamma| \leq \alpha^3$;
- (ii) $|D'([c, n]) - \gamma'| \leq \alpha^3$ and $\gamma' \geq \gamma$;
- (iii) $D'([c, n]) \geq D'([b+1, 2b-a+1]) - 2\alpha^3$ and $D'([c+1, n]) < D'([b+1, 2b-a+1]) + 2\alpha^3$;
- (iv) $\gamma \leq 2\varepsilon + 4\alpha^3$.

Proof. Again by invoking the DKW inequality, we can obtain (with probability at least $1 - \delta$) an approximation \hat{D} of D' , close up to $\alpha^3/2$ in Kolmogorov distance. This provides us with an estimate γ of $D'([b+1, 2b-a+1])$ satisfying the first item (as, for any interval $[r, s]$, $\hat{D}([r, s])$ is within an additive $\alpha^3/2$ of $D'([r, s])$). Then, setting

$$c \stackrel{\text{def}}{=} \max \left\{ x \in [n] : \hat{D}([x, n]) \geq \gamma \right\}$$

and $\gamma' \stackrel{\text{def}}{=} \hat{D}([c, n])$, items (ii) and (iii) follow. The last bound of (iv) derives from an argument identical to that of Lemma 7.3 and the promise that D' is ε -close to monotone; indeed, one must then have $D'([b+1, 2b-a+1]) \leq D'([a, b]) + 2\varepsilon \leq 2\varepsilon + 3\alpha^2$, which with (i) concludes the argument. \square

To finish the proof of Theorem 7.1, we apply the above lemmata with $\alpha \stackrel{\text{def}}{=} \Theta(\sqrt{\varepsilon_2})$ and need to show that the algorithm generates samples from a distribution that is $\varepsilon_2 = O(\alpha^2)$ -close to monotone. This is done by bounding the error encountered (due to approximation errors) in the following parts of the algorithm: when estimating the weight γ of an interval of equal length adjacent to the interval $[a, b]$, uniformizing its weight on that interval, and estimating the last γ -quantile of the distribution, in order to move the weight needed to fill the gap from there. If we could have perfect

estimates of the gap $([a, b] = [i, j])$, the missing weight γ , and the point c such that $D'([c, n]) = \gamma$, the corrected distribution would be monotone, as the probability mass function in both the gap and the next interval would be at the same “level” (that is, $\frac{\gamma}{b-a+1}$).

By choice of m , with probability at least $1 - \delta$ the two subroutines of the PREPROCESSING stage (from Lemmas 7.5 and 7.6) behave as expected. We hereafter condition on this being the case. For convenience, we write $I = [a, b]$, $J = [b + 1, 2b - a + 1]$ and $K = [c, n]$, where a, b, c and γ, γ' are the outcome of the preprocessing phase.

If the test in Line 9 passes. If the preprocessing stage returned either close or a value $\gamma < 5\varepsilon_2^{3/2} = 5\alpha^3$, then we claim that D' is already $O(\alpha^2)$ -close to monotone. The first case is by correctness of Lemma 7.5; as for the second, observe that it implies $D'(J) < 6\alpha^3$. Thus, “putting back” (from the tail of the support) weight at most $6\alpha^3$ in $[i, j]$ would be sufficient to correct the violation of monotonicity, which yields an $O(\alpha^3)$ upper bound on the distance of D' to monotone.

Otherwise. This implies in particular that $\gamma \geq 5\alpha^3$, and thus $D'(J) \geq 4\alpha^3$. By Lemma 7.6 (iii), it is then also the case that $D'(K) \geq 2\alpha^3$. Then, denoting by \tilde{D} the corrected distribution, we have

$$\tilde{D}(x) = \begin{cases} D'(x) + \frac{\gamma}{\gamma'} \cdot \frac{D'(K)}{|I|} & \text{if } x \in I, \\ \frac{D'(J)}{|J|} & \text{if } x \in J, \\ D'(x) \cdot (1 - \frac{\gamma}{\gamma'}) & \text{if } x \in K, \\ D'(x) & \text{otherwise.} \end{cases}$$

Distance to D' . From the expression above, we get that

$$2d_{\text{TV}}(\tilde{D}, D') \leq \frac{\gamma}{\gamma'} D'(K) + 2D'(J) + \frac{\gamma}{\gamma'} D'(K) = 2 \left(\frac{\gamma}{\gamma'} D'(K) + D'(J) \right).$$

From Lemma 7.6, we also know that $D'(J) \leq \gamma + \alpha^3$, $D'(K) \leq \gamma' + \alpha^3$, and $\gamma/\gamma' \leq 1$, so that

$$d_{\text{TV}}(\tilde{D}, D') \leq \frac{\gamma}{\gamma'} (\gamma' + \alpha^3) + \gamma + \alpha^3 \leq 2(\gamma + \alpha^3) \leq 4\varepsilon + 10\alpha^3 = O(\varepsilon)$$

(where, for the last inequality, we used Lemma 7.6 (iv) and finally the fact that $\varepsilon_2 \leq \varepsilon$).

Distance to monotone. Consider the distributions M defined as

$$M(x) = \begin{cases} D'(x) + \frac{D'(J)}{|I|} & \text{if } x \in I, \\ \frac{D'(J)}{|J|} & \text{if } x \in J, \\ D'(x) \cdot \left(1 - \frac{D'(J)}{D'(K)}\right) & \text{if } x \in K, \\ D'(x) & \text{otherwise.} \end{cases}$$

We first claim that M is $O(\alpha^2)$ -close to monotone. Indeed, M is monotone on $[a, n]$ by construction (and as D' was monotone on $[b, n]$). The only possible violations of monotonicity are on $[1, b]$, due to the approximation of (i, j) by (a, b) —that is, it is possible for the interval $[a, i]$ to now have too much weight, with $M(a - 1) < M(a)$. But as we have $D'([a, b]) \leq 3\alpha^2$, the total extra weight of this “violating bump” is $O(\alpha^2)$.

Moreover, the distance between M and \tilde{D} can be upper bounded by their difference on J and K :

$$2d_{\text{TV}}(\tilde{D}, M) \leq 2 \left| D'(J) - \frac{\gamma}{\gamma'} D'(K) \right| \leq 2\alpha^3 \frac{1 + \frac{\alpha^3}{D'(K)}}{1 - \frac{\alpha^3}{D'(K)}} \leq 6\alpha^3,$$

where we used the fact that $\frac{\gamma}{\gamma'} \in \left[\frac{D'(J) - \alpha^3}{D'(K) + \alpha^3}, \frac{D'(J) + \alpha^3}{D'(K) - \alpha^3} \right]$ and that $D'(K) \geq 2\alpha^3$. By the triangle inequality, \tilde{D} is then itself $O(\alpha^2)$ -close to monotone. This concludes the proof of Theorem 7.1.

8. Focusing on randomness scarcity.

8.1. Correcting uniformity. In order to illustrate the challenges and main aspects of this section, we shall focus on what is arguably the most natural property of interest, “being uniform” (i.e., $\mathcal{P} = \{\mathcal{U}_n\}$). As a first observation, we note that when one is interested in correcting uniformity on an arbitrary domain \mathcal{X} , allowing arbitrary amounts of additional randomness makes the task almost trivial; by using roughly $\log |\mathcal{X}|$ random bits per query, it is possible to interpolate arbitrarily between D and the uniform distribution. One can naturally ask whether the same can be achieved *while using no—or very little—additional randomness besides the draws from the sampling oracle itself*. As we show below, this is possible, at the price of a slightly worse query complexity. We hereafter focus once again on the case $\mathcal{X} = [n]$ and give constructions which achieve different tradeoffs between the level of correction (of D to uniform), the fidelity to the original data (closeness to D) and the sample complexity. We then show how to combine these constructions to achieve reasonable performance in terms of all the above parameters. In section 8.1.1, we turn to the related problem of correcting uniformity on an (unknown) subgroup of the domain and extend our results to this setting. Finally, we discuss the differences and relations with extractors in section 8.2.

High-level ideas. The first algorithm we describe (Theorem 8.1) is a sampling corrector based on a “von Neumann-type” approach: by seeing very crudely the distribution D as a distribution over two points (the first and second half of the support $[n]$), one can leverage the closeness of D to uniform to obtain with overwhelming probability a sequence of uniform random bits and use them to generate a uniform element of $[n]$. The drawback of this approach lies in the number of samples required from D , namely, $\Theta(\log n)$.

The second approach we consider relies on viewing $[n]$ as the Abelian group \mathbb{Z}_n and leverages crucial properties of the convolution of distributions. Using a robust version of the fact that the uniform distribution is the absorbing element for this operation, we are able to argue that taking a *constant* number of samples from D and outputting their sum obeys a distribution \tilde{D} exponentially closer to uniform (Theorem 8.2). This result, however efficient in terms of getting closer to uniform, does not guarantee anything nontrivial about the distance \tilde{D} to the input distribution D . More precisely, starting from D which is at a distance ε from uniform, it is possible to end up with \tilde{D} at a distance ε' from uniform, but $\varepsilon + \Omega(\varepsilon')$ from D (see Claim A.4 for more details). In other terms, this improver does get us closer to uniform, but somehow can *overshoot* in the process, getting too far from the input distribution.

The third improver we describe (in Theorem 8.3) yields slightly different parameters; it essentially enables one to get “midway” between D and the uniform distribution, and to sample from a distribution \tilde{D} (almost) $(\varepsilon/2)$ -close to *both* the input and the uniform distributions. It achieves this by combining both previous ideas: using D to generate a (roughly) unbiased coin toss and deciding based on the outcome whether to output a sample from D or from the improver of Theorem 8.2.

Finally, by “bootstrapping” the hybrid approach described above, one can provide sampling access to an improved \hat{D} both arbitrarily close to uniform *and* (almost) optimally close to the original distribution D (up to an additive $O(\varepsilon^3)$), as described in Theorem 8.4. Note that this is at a price of an extra $\log(1/\varepsilon_2)$ factor in the sample complexity, compared to Theorem 8.2: in a sense, the price of “staying faithful to the input data.”

THEOREM 8.1 (von Neumann sampling corrector). *For any $\varepsilon < 0.49$ (and $\varepsilon_1 = \varepsilon$) as in the definition, there exists a sampling corrector for uniformity with query complexity $O(\log n(\log \log n + \log(1/\delta)))$ (where δ is the probability of failure per sample).*

Proof. Let D be a distribution over $[n]$ such that $d_{TV}(D, \mathcal{U}) \leq \varepsilon < 1/2 - c$ for some absolute constant $c < 1/2$ (e.g., $c = 0.49$), and let S_0, S_1 denote, respectively, the sets $\{1, \dots, n/2\}$ and $\{n/2 + 1, \dots, n\}$. The high-level idea is to see a draw from D as a (biased) coin toss, depending on whether the sample lands in S_0 or S_1 ; by applying von Neumann’s method, we then can retrieve a truly uniform bit at a time (with high probability). Repeating this $\log n$ times will yield a uniform draw from $[n]$. More precisely, it is immediate by definition of the total variation distance that $|D(S_0) - D(S_1)| \leq 2\varepsilon$, so in particular (setting $p \stackrel{\text{def}}{=} D(S_0)$) we have access to a Bernoulli random variable with parameter $p \in [\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon]$.

To generate *one* uniform random bit (with probability of failure at most $\delta' = \delta/\log n$), it is sufficient to take in the worst case $m \stackrel{\text{def}}{=} \lceil (\log \frac{1}{1-c})^{-1} \log \frac{2}{\delta'} \rceil$ samples and stop as soon as a sequence S_0S_1 or S_1S_0 is seen (giving resp. a bit 0 or 1). If it does not happen, then the corrector VN-IMPROVER _{n} outputs FAIL; the probability of failure is therefore

$$\Pr[\text{VN-IMPROVER}_n \text{ outputs FAIL}] = p^m + (1-p)^m \leq 2 \cdot (1-c)^m \leq \delta' = \frac{\delta}{\log n}.$$

By a union bound over the $\log n$ bits to extract, VN-IMPROVER _{n} indeed outputs a uniform random number $s \in [n]$ with probability at least $1 - \delta$, using at most $m \log n = (\log n \log \frac{\log n}{\delta})$ samples—and, in expectation, only $O((\log n)/p) = O(\log n)$. \square

As previously mentioned, we hereafter work modulo n , equating $[n]$ to the Abelian group $(\mathbb{Z}_n, +)$. This convenient (and equivalent) view will allow us to use properties of convolutions of distributions over Abelian groups,¹⁶ in particular the fact that the uniform distribution on \mathbb{Z}_n is (roughly speaking) an attractive fixed point for this operation. In particular, taking D to be the (unknown) distribution promised to be ε -close to uniform, Fact A.3 guarantees that by drawing two independent samples $x, y \sim D$ and computing $z = x + y \pmod n$, the distribution of z is $(2\varepsilon^2)$ -close to the uniform distribution on $\{0, \dots, n - 1\}$. This key observation is the basis for our next result.

THEOREM 8.2 (convolution improver). *For any $\varepsilon < \frac{1}{\sqrt{2}}$, ε_2 , and $\varepsilon_1 = \varepsilon + \varepsilon_2$ as in the definition, there exists a sampling improver for uniformity with query complexity $(\frac{\log \frac{1}{\varepsilon_2}}{\log \frac{1}{\varepsilon}})$.*

Proof. Extending by induction the observation above to a sum of finitely many independent samples, we get that by drawing $k \stackrel{\text{def}}{=} \frac{\log \frac{1}{\varepsilon_2} - 1}{\log \frac{1}{\varepsilon} - 1}$ independent elements s_1, \dots, s_k from D and computing

¹⁶For more detail on this topic, the reader is referred to Appendix A.

$$s = \left(\sum_{\ell=1}^k s_{\ell} \pmod{n} \right) + 1 \in [n];$$

the distribution \tilde{D} of s is $(\frac{1}{2}(2\varepsilon)^k)$ -close to uniform, and by choice of k , $(\frac{1}{2}(2\varepsilon)^k) = \varepsilon_2$. As $d_{\text{TV}}(D, \tilde{D}) \leq d_{\text{TV}}(D, \mathcal{U}) + d_{\text{TV}}(\mathcal{U}, \tilde{D}) \leq \varepsilon + \varepsilon_2$, the vacuous bound on the distance between D and \tilde{D} is as stated. \square

This triggers a natural question, namely, can this “vacuous bound” be improved? That is, setting $\varepsilon \stackrel{\text{def}}{=} d_{\text{TV}}(D, \mathcal{U})$ and $D^{(k)} \stackrel{\text{def}}{=} D * \dots * D$ (k -fold convolution), what can be said about $d_{\text{TV}}(D, D^{(k)})$ as a function of ε and k ? Trivially, the triangle inequality asserts that

$$\varepsilon - 2^{k-1}\varepsilon^k \leq d_{\text{TV}}(D, D^{(k)}) \leq \varepsilon + 2^{k-1}\varepsilon^k;$$

but can the right-hand side be tightened further? For instance, one might hope to achieve ε . Unfortunately, this is not the case; even for $k = 2$, one cannot get better than $\varepsilon + \Omega(\varepsilon^2)$ as an upper bound. Indeed, one can show that for $\varepsilon \in (0, \frac{1}{2})$, there exists a distribution D on \mathbb{Z}_n such that $d_{\text{TV}}(D, \mathcal{U}) = \varepsilon$, yet $d_{\text{TV}}(D, D * D) = \varepsilon + \frac{3}{4}\varepsilon^2 + O(\varepsilon^3)$ (see Claim A.4 in the Appendix A).

THEOREM 8.3 (hybrid improver). *For any $\varepsilon \leq \frac{1}{2}$, $\varepsilon_1 = \frac{\varepsilon}{2} + 2\varepsilon^3 + \varepsilon'$ and $\varepsilon_2 = \frac{\varepsilon}{2} + \varepsilon'$, there exists a sampling improver for uniformity with query complexity $O(\frac{\log \frac{1}{\varepsilon'}}{\log \frac{1}{\varepsilon}})$.*

Proof. Let D be a distribution over $[n]$ such that $d_{\text{TV}}(D, \mathcal{U}) = \varepsilon$, and write d_0 (resp., d_1) for $D(\{1, \dots, n/2\})$ (resp., $D(\{n/2+1, \dots, n\})$). By definition, $|d_0 - d_1| \leq 2\varepsilon$. Define the Bernoulli random variable X by taking two independent samples s_1, s_2 from D and setting X to 0 if both land in the same half of the support (both in $\{1, \dots, n/2\}$ or both in $\{n/2+1, \dots, n\}$). It follows that $p_0 \stackrel{\text{def}}{=} \Pr[X = 0] = d_0^2 + d_1^2$ and $p_1 \stackrel{\text{def}}{=} \Pr[X = 1] = 2d_0d_1$, i.e., $0 \leq p_0 - p_1 = (d_1 - d_0)^2 \leq 4\varepsilon^2$. In other terms, $X \sim \text{Bern}(p_0)$ with $\frac{1}{2} \leq p_0 \leq \frac{1}{2} + 2\varepsilon^2$.

Consider now the distribution

$$\tilde{D} \stackrel{\text{def}}{=} (1 - p_0)D + p_0D^{(k)},$$

where $D^{(k)} = \overbrace{D * \dots * D}^{k \text{ times}}$ as in Theorem 8.2. Observe that getting a sample from \tilde{D} only requires at most $k + 2$ queries¹⁷ to the oracle for D . Moreover,

$$\begin{aligned} d_{\text{TV}}(\tilde{D}, \mathcal{U}) &\leq (1 - p_0)d_{\text{TV}}(D, \mathcal{U}) + p_0d_{\text{TV}}(D^{(k)}, \mathcal{U}) \leq (1 - p_0)\varepsilon + p_02^{k-1}\varepsilon^k \\ &\leq \frac{\varepsilon}{2} + \left(\frac{1}{4} + \varepsilon^2\right)(2\varepsilon)^k \leq \frac{\varepsilon}{2} + \frac{1}{2}(2\varepsilon)^k \end{aligned}$$

while

$$\begin{aligned} d_{\text{TV}}(\tilde{D}, D) &\leq p_0d_{\text{TV}}(D^{(k)}, D) \leq p_0(\varepsilon + 2^{k-1}\varepsilon^k) \leq \left(\frac{1}{2} + 2\varepsilon^2\right)(\varepsilon + 2^{k-1}\varepsilon^k) \\ &\leq \frac{\varepsilon}{2} + 2\varepsilon^3 + \frac{1}{2}(2\varepsilon)^k \end{aligned}$$

¹⁷More precisely, 3 with probability $1 - p_0$, and $k + 2$ with probability p_0 , for an expected number $(k - 1)p_0 + 3 \simeq k/2$.

(recalling for the rightmost step of each inequality that $\varepsilon \leq \frac{1}{2}$). Taking $k = 3$, one obtains, with a sample complexity at most 5, a distribution \tilde{D} satisfying

$$d_{\text{TV}}(\tilde{D}, \mathcal{U}) \leq \frac{\varepsilon}{2} + 4\varepsilon^3, \quad d_{\text{TV}}(\tilde{D}, D) \leq \frac{\varepsilon}{2} + 6\varepsilon^3.$$

(Note that assuming $\varepsilon < 1/4$, one can get the more convenient—yet looser—bounds $d_{\text{TV}}(\tilde{D}, \mathcal{U}) \leq \frac{21}{32}\varepsilon < \frac{2\varepsilon}{3}$, $d_{\text{TV}}(\tilde{D}, D) \leq \frac{97\varepsilon}{128} < \frac{4\varepsilon}{5}$.) \square

THEOREM 8.4 (bootstrapping improver). *For any $\varepsilon \leq \frac{1}{2}$, $0 < \varepsilon_2 < \varepsilon$, and $\varepsilon_1 = \varepsilon - \varepsilon_2 + O(\varepsilon^3)$, there exists a sampling improver for uniformity with query complexity $O(\frac{\log^2 \frac{1}{\varepsilon_2}}{\log \frac{1}{\varepsilon}})$.*

Proof. We show how to obtain such a guarantee—note that the constant 27 in the $O(\varepsilon^3)$ is not tight and can be reduced at the price of a more cumbersome analysis. Let $\alpha > 0$ be a parameter (to be determined later) satisfying $\alpha < \varepsilon^2$, and k be the number of bootstrapping steps—i.e., the number of time one recursively applies the construction of Theorem 8.3 with α . We write D_j for the distribution obtained after the j th recursive step, so that $D_0 = D$ and $\hat{D} = D_k$ and let u_j (resp., v_j) denote an upper bound on $d_{\text{TV}}(D_j, \mathcal{U})$ (resp., $d_{\text{TV}}(D_j, D)$). Note that by the guarantee of Theorem 8.3 and applying a triangle inequality for v_j , one gets the following recurrence relations for $(u_j)_{0 \leq j \leq k}$ and $(v_j)_{0 \leq j \leq k}$:

$$\begin{aligned} u_0 &= \varepsilon, & u_{j+1} &= \frac{1}{2}u_j + \alpha, \\ v_0 &= 0, & v_{j+1} &= \left(\frac{1}{2}u_j + 2u_j^3 + \alpha\right) + v_j. \end{aligned}$$

Solving this recurrence for u_k gives

$$(7) \quad u_k = \frac{\varepsilon}{2^k} + 2\left(1 - \frac{1}{2^k}\right)\alpha < \frac{\varepsilon}{2^k} + 2\alpha$$

while one gets an upper bound on v_k by writing

$$\begin{aligned} v_k &= v_k - v_0 = \sum_{j=0}^{k-1} (v_{j+1} - v_j) = k\alpha + \frac{1}{2} \sum_{j=0}^{k-1} u_j + 2 \sum_{j=0}^{k-1} u_j^3 \\ &= 2k\alpha + \left(1 - \frac{1}{2^k}\right)\varepsilon - 2\left(1 - \frac{1}{2^k}\right)\alpha + 2 \sum_{j=0}^{k-1} u_j^3 \\ &< \left(1 - \frac{1}{2^k}\right)\varepsilon + 2 \underbrace{\left(k - 1 + \frac{1}{2^k}\right)\alpha}_{\leq k\alpha} + (3\varepsilon^3 + 16\varepsilon^2\alpha + 48\varepsilon\alpha^2 + 16k\alpha^3), \end{aligned}$$

where we used the expression (7) for u_j . Since $\alpha < \varepsilon^2 \leq \frac{1}{4}$, we can bound the rightmost terms as $16k\alpha^3 \leq k\alpha$, $48\varepsilon\alpha^2 < 48\varepsilon^5$ and $16\varepsilon^2\alpha < 16\varepsilon^4$, so that

$$(8) \quad v_k < \left(1 - \frac{1}{2^k}\right)\varepsilon + 3k\alpha + 3\varepsilon^3 + 16\varepsilon^4 + 48\varepsilon^5 < \left(1 - \frac{1}{2^k}\right)\varepsilon + 3k\alpha + 23\varepsilon^3.$$

It remains to choose k and α ; to get $u_k \leq \varepsilon_2$, set $k \stackrel{\text{def}}{=} \lceil \log \frac{\varepsilon}{\varepsilon_2(1-\varepsilon^2)} \rceil \leq \log \frac{4\varepsilon}{3\varepsilon_2} + 1$ and $\alpha \stackrel{\text{def}}{=} \frac{1}{2}\varepsilon_2\varepsilon^2$, so that $\frac{\varepsilon}{2^k} \leq (1 - \varepsilon^2)\varepsilon_2$ and $2\alpha \leq \varepsilon^2\varepsilon_2$. Plugging these values in (8),

$$v_k \underset{(\varepsilon_2 < \varepsilon)}{<} \left(1 - \frac{\varepsilon_2(1 - \varepsilon^2)}{\varepsilon}\right) \varepsilon + \frac{3}{2}k\varepsilon_2\varepsilon^2 + 23\varepsilon^3 = \varepsilon - \varepsilon_2 + \frac{3}{2}k\varepsilon_2\varepsilon^2 + 24\varepsilon^3 < \varepsilon - \varepsilon_2 + 27\varepsilon^3,$$

where the last inequality comes from the fact that $\frac{3}{2}k\frac{\varepsilon_2}{\varepsilon} \leq \frac{3}{2} \log \frac{8\varepsilon}{3\varepsilon_2} \cdot \frac{\varepsilon_2}{\varepsilon} \leq 3$. Therefore, we have $d_{TV}(D_k, \mathcal{U}) \leq \varepsilon_2$, $d_{TV}(D_k, D) \leq \varepsilon - \varepsilon_2 + 27\varepsilon^3$ as claimed. We turn to the number m of queries made along those k steps; from Theorem 8.3, this is at most

$$m \leq \sum_{j=0}^{k-1} \left\lceil \frac{\log \frac{1}{\alpha} - 1}{\log \frac{1}{u_j} - 1} \right\rceil \leq k \cdot \left\lceil \frac{\log \frac{1}{\alpha} - 1}{\log \frac{1}{\varepsilon} - 1} \right\rceil = O\left(\frac{\log^2 \frac{1}{\varepsilon_2}}{\log \frac{1}{\varepsilon}}\right),$$

which concludes the proof. □

Note that in all four cases, as our improvers do not use any randomness of their own, they always output according to the same improved distribution; that is, after fixing the parameters $\varepsilon, \varepsilon_2$ and the unknown distribution D , then \hat{D} is uniquely determined, even across independent calls to the improver.

8.1.1. Correcting uniformity on a subgroup.

Outline. It is easy to observe that all the results above still hold when replacing \mathbb{Z}_n by any finite Abelian group G . Thus, a natural question to turn to is whether one can generalize these results to the case where the unknown distribution is close to the uniform distribution on an arbitrary, unknown, *subgroup* H of the domain G .

To do so, a first observation is that if H were known, and if furthermore a constant (expected) fraction of the samples were to fall within it, then one could directly apply our previous results by conditioning samples on being in H , using rejection sampling. The results of this section show how to achieve this “identification” of the subgroup with only a $\log(1/\varepsilon)$ overhead in the sample complexity. At a highlevel, the idea is to take a few samples and argue that their greatest common divisor will (with high probability) be a generator of the subgroup.

Details. Let G be a finite cyclic Abelian group of order n and $H \subseteq G$ a subgroup of order m . We denote by \mathcal{U}_H the uniform distribution on this subgroup. Moreover, for a distribution D over G , we write D_H for the conditional distribution it induces on H , that is

$$\forall x \in G, \quad D_H(x) = \frac{D(x)}{D(H)} \mathbf{1}_H(x)$$

which is defined as long as D puts nonzero weight on H . The following lemma shows that if D is close to \mathcal{U}_H , then so is D_H .

LEMMA 8.5. *Assume $d_{TV}(D, \mathcal{U}_H) < 1$. Then $d_{TV}(D_H, \mathcal{U}_H) \leq d_{TV}(D, \mathcal{U}_H)$.*

Proof. First, observe that the assumption implies D_H is well defined; indeed, as $d_{TV}(D, \mathcal{U}_H) = \sup_{S \subseteq G} (\mathcal{U}_H(S) - D(S))$, taking $S = H$ yields $1 > d_{TV}(D, \mathcal{U}_H) \geq \mathcal{U}_H(H) - D(H) = 1 - D(H)$, and thus $D(H) > 0$.

Rewriting the definition of $d_{TV}(D_H, \mathcal{U}_H)$, one gets

$$d_{TV}(D, \mathcal{U}_H) = \frac{1}{2} \left(\sum_{x \in H} \left| D(x) - \frac{1}{|H|} \right| + \sum_{x \notin H} D(x) \right);$$

so that

$$\begin{aligned}
 2d_{\text{TV}}(D_H, \mathcal{U}_H) &= \sum_{x \in H} \left| D_H(x) - \frac{1}{|H|} \right| \leq \sum_{x \in H} |D_H(x) - D(x)| + \sum_{x \in H} \left| D(x) - \frac{1}{|H|} \right| \\
 &= \sum_{x \in H} |D_H(x) - D(x)| + \left(2d_{\text{TV}}(D, \mathcal{U}_H) - \sum_{x \notin H} D(x) \right) \\
 &= \sum_{x \in H} D(x) \left| \frac{1}{D(H)} - 1 \right| + 2d_{\text{TV}}(D, \mathcal{U}_H) - (1 - D(H)) \\
 &= D(H) \left| \frac{1}{D(H)} - 1 \right| + 2d_{\text{TV}}(D, \mathcal{U}_H) - (1 - D(H)) \\
 &= |1 - D(H)| + 2d_{\text{TV}}(D, \mathcal{U}_H) - (1 - D(H)) \\
 &= 2d_{\text{TV}}(D, \mathcal{U}_H). \quad \square
 \end{aligned}$$

Let D be a distribution on G promised to be ε -close to the uniform distribution \mathcal{U}_H on some unknown subgroup H , for $\varepsilon < \frac{1}{2} - c$. For the sake of presentation, we hereafter without loss of generality identify G to \mathbb{Z}_n . Let h be the generator of H with smallest absolute values (when seen as an integer), so that $H = \{0, h, 2h, 3h, \dots, (m-1)h\}$.

Observe that $D(H) > 1 - 2\varepsilon$, as $2d_{\text{TV}}(D, \mathcal{U}_H) = \sum_{x \in H} \left| D(x) - \frac{1}{|H|} \right| + D(H^c)$; therefore, if H were known, one could efficiently simulate sample access to D_H via rejection sampling, with only a constant factor overhead (in expectation) per sample. It would then become possible, as hinted in the foregoing discussion, to correct uniformity on D_H (which is ε -close to \mathcal{U}_H by Lemma 8.5) via one of the previous algorithms for Abelian groups. The question remains to show how to find H or, equivalently, h .

Algorithm 5. Algorithm FIND-GENERATOR-SUBGROUP.

Require: $\varepsilon \in (0, \frac{1}{2} - c]$, SAMP_D with D ε -close to uniform on some subgroup $H \subseteq \mathbb{Z}_n$

Ensure: Outputs a generator \hat{h} of H with probability $1 - \tilde{O}(\varepsilon)$

Draw k independent samples s_1, \dots, s_k from D , for $k \stackrel{\text{def}}{=} O(\log \frac{1}{\varepsilon})$

Compute $\hat{h} = \text{gcd}(s_1, \dots, s_k)$

return \hat{h}

LEMMA 8.6. *Let G, H be as before. There exists an algorithm (Algorithm 5) which, given $\varepsilon < 0.49$ as well as sample access to some distribution D over G , makes $O(\log \frac{1}{\varepsilon})$ calls to the oracle and returns an element of G . Further, if $d_{\text{TV}}(D, \mathcal{U}_H) \leq \varepsilon$, then with probability at least $1 - \tilde{O}(\varepsilon)$ its output is a generator of H .*

Proof. In order to argue correctness of the algorithm, we will need the following well-known facts.

Fact 8.7. Fix any $k \geq 1$, and let $p_{n,k}$ be the probability that k independent numbers drawn uniformly at random from $[n]$ be relatively prime. Then $p_{n,k} \xrightarrow{n \rightarrow \infty} \frac{1}{\zeta(k)}$ (where ζ is the Riemann zeta function).

Fact 8.8. One has $\zeta(x) \underset{x \rightarrow \infty}{=} 1 + \frac{1}{2^x} + o(\frac{1}{2^x})$, and in particular $\frac{1}{\zeta(k)} \underset{k \rightarrow \infty}{=} 1 - \frac{1}{2^k} + o(\frac{1}{2^k})$.

With this in hand, let $k \stackrel{\text{def}}{=} O(\log \frac{1}{\varepsilon})$, chosen so that $\varepsilon k = \Theta(\frac{1}{2^k}) = \Theta(\frac{\varepsilon}{\log \frac{1}{\varepsilon}})$. We break the analysis of our subgroup-finding algorithm in two cases.

Case 1: $|H| = \Theta(1)$. This is the easy case: if H only contains constantly many elements (m is a constant of n and ε), then after taking k samples $s_1, \dots, s_k \sim D$, we have

- $s_1, \dots, s_k \in H$ (event E_1) with probability at least $(1 - \varepsilon)^k = 1 - O(k\varepsilon)$;
- the probability that there exists an element of H not hit by any of the s_i 's is at most, by a union bound,

$$\sum_{x \in H} (1 - D(x))^k \leq m \left(1 - \frac{1}{m} + \varepsilon\right)^k = 2^{-\Omega(k)}$$

for ε sufficiently small ($\varepsilon \ll \frac{1}{m}$). Let E_2 be the event each element of H appears amongst the samples.

Overall, with probability $1 - O(k\varepsilon)$ (conditioning on E_1 and E_2), our set of samples is exactly H , and $\gcd(s_1, \dots, s_k) = \gcd(H) = h$.

Case 2: $|H| = \omega(1)$. This amounts to saying that $h = o(n)$. In this case, taking again k samples $s_1, \dots, s_k \sim D$ and denoting by \hat{h} their greatest common divisor,

- $s_1, \dots, s_k \in H$ (event E_1) with probability at least $(1 - \varepsilon)^k = 1 - O(k\varepsilon)$ as before;
- conditioned on E_1 , note that if the s_i 's were *uniformly* distributed in H , then the probability that $\hat{h} = h$ would be exactly $p_{\hat{h},k}^n$, as $\gcd(ha, \dots, hb) = h$ if and only if $\gcd(a, \dots, b) = 1$, i.e., if a, \dots, b are relatively prime. In this ideal scenario, therefore, we would have

$$\Pr[\gcd(s_1, \dots, s_k) = h \mid E_1] = p_{\hat{h},k}^n \xrightarrow{n \rightarrow \infty} \frac{1}{\zeta(k)} = 1 - O\left(\frac{1}{2^k}\right)$$

by Fact 8.7 and our assumption $h = o(n)$.

To adapt this result to our case—where $s_1, \dots, s_k \sim D_H$ (as we conditioned on E_1), it is sufficient to observe that by the data processing inequality for total variation distance,

$$\left| \Pr_{s_1, \dots, s_k \sim D_H} [\gcd(s_1, \dots, s_k) = h] - \Pr_{s_1, \dots, s_k \sim \mathcal{U}_H} [\gcd(s_1, \dots, s_k) = h] \right| \leq d_{\text{TV}}(D_H^{\otimes k}, \mathcal{U}_H^{\otimes k}) \leq k\varepsilon$$

so that in our case

$$\begin{aligned} \Pr[\gcd(s_1, \dots, s_k) = h \mid E_1] &\geq p_{\hat{h},k}^n - k\varepsilon \xrightarrow{n \rightarrow \infty} \frac{1}{\zeta(k)} - k\varepsilon \\ (9) \qquad \qquad \qquad &= 1 - O\left(\frac{1}{2^k}\right) = 1 - O\left(\frac{\varepsilon}{\log \frac{1}{\varepsilon}}\right). \end{aligned}$$

In either case, with probability at least $1 - \tilde{O}(\varepsilon)$, we find a generator h of H , acting as a (succinct) representation of H which allows us to perform rejection sampling. \square

This directly implies the theorem below; any sampling improver for uniformity on a group directly yields an improver for uniformity on an unknown *subgroup* with essentially the same complexity.

THEOREM 8.9. *Suppose we have an $(\varepsilon, \varepsilon_1, \varepsilon_2)$ -sampling improver for uniformity over Abelian finite cyclic groups, with query complexity $q(\varepsilon, \varepsilon_1, \varepsilon_2)$. Then there exists an $(\varepsilon, \varepsilon_1, \varepsilon_2)$ -sampling improver for uniformity on subgroups, with query complexity*

$$O\left(\log \frac{1}{\varepsilon} + q(\varepsilon, \varepsilon_1, \varepsilon_2) \log q(\varepsilon, \varepsilon_1, \varepsilon_2)\right).$$

Proof. Proof is straightforward (rejection sampling over the subgroup, once identified: constant probability of hitting it, so by trying at most $O(\log q)$ draws per samples before outputting FAIL, one can provide a sample from D_H to the original algorithm with probability $1 - 1/10q$, for each of the (at most) q queries). \square

8.2. Comparison with randomness extractors. In the randomness extractor model, one is provided with a source of imperfect random bits (and sometimes an additional source of completely random bits), and the goal is to output as many random bits as possible that are close to uniformly distributed. In the distribution corrector model, one is provided with a distribution that is *close to having* a property \mathcal{P} , and the goal is to have the ability to generate a *similar* distribution that *has* property \mathcal{P} .

One could therefore view extractors as sampling improvers for the property of uniformity of distributions (i.e., $\mathcal{P} = \{\mathcal{U}_n\}$); indeed, both sampling correctors and extractors attempt to minimize the use of extra randomness. However, there are significant differences between the two settings. A first difference is that randomness extractors assume a lower bound on the min-entropy¹⁸ of the input distribution, whereas sampling improvers assume the distribution to be ε -close to uniform in total variation distance. Note that the two assumptions are not comparable.¹⁹

Secondly, in both the extractor and sampling improver models, since the entropy of the output distribution should be larger, one would either need more random bits from the weak random source or additional uniform random bits. Our sampling improvers do not use any extra random bits, which is also the case in deterministic extractors but not in other extractor constructions. However, unlike the extractor model, in the sampling improver model, there is no bound on the number of independent samples one can take from the original distribution. Tight bounds and impossibility results are known for both general and deterministic extractors [Vad12, RTS00], in particular in terms of the amount of additional randomness required. Because of the aforementioned differences in both the assumptions on and access to the input distribution, these lower bounds do not apply to our setting—which explains why our sampling improvers avoid this need for extra random bits.

8.3. Monotone distributions and randomness scarcity. In this section, we describe how to utilize a (close-to-monotone) input distribution to obtain the uniform random samples some of our previous correctors and improvers need. This is at the price of a $\tilde{O}(\log n)$ -sample overhead per draw and follows the same general approach as in Theorem 8.1. We observe that even if this *seems* to defeat the goal (as, with this many samples, one could even *learn* the distribution, as stated in Lemma 6.1), this is not actually the case; indeed, the procedure below is meant as a subroutine

¹⁸The min-entropy of a distribution D is defined as $H_\infty(D) = \log \frac{1}{\max_i D(i)}$.

¹⁹For example, the min-entropy of a distribution which is ε -close to uniform can range from $\log(1/\varepsilon)$ to $\Omega(\log n)$. Conversely, the distance to uniformity of a distribution which has high min-entropy can also vary significantly; there exist distributions with min-entropy $\Omega(\log n)$ but which are respectively $\Omega(1)$ -far from and $O(1/n)$ -close to uniform.

for these very same correctors, emancipating them from the need for truly additional randomness—which they would require otherwise, e.g., to generate samples from the corrected or learned distribution.

LEMMA 8.10 (randomness from (almost) monotone). *There exists a procedure which, given $\varepsilon \in [0, 1/3)$ and $\delta > 0$, as well as sample access to a distribution D guaranteed to be ε -close to monotone, either returns*

- *point mass, if D is ε -close to the point distribution²⁰ on the first element,*
- *or a uniform random sample from $[n]$,*

with probability of failure at most δ . The procedure makes $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples from D in the first case, and $O(\frac{\log n}{\varepsilon} \log \frac{\log n}{\delta})$ in the second.

Proof. By taking $O(\log(1/\delta)/\varepsilon^2)$ samples, the algorithm starts by approximating by \hat{F} the cdf F of the distribution up to an additive $\frac{\varepsilon}{4}$ in ℓ_∞ . Then, it defines

$$m \stackrel{\text{def}}{=} \min \left\{ i \in [n] : \hat{F}(i) \geq 1 - \frac{\varepsilon}{2} \right\}$$

so that $F(m) \geq 1 - \frac{3\varepsilon}{4}$. According to the value of m , we consider two cases:

- If $m = 1$, then D is ε -close to the (monotone) distribution δ_1 which has all weight on the first element; this means we have effectively learned the distribution and can thereafter consider, for all purposes, δ_1 in lieu of D .
- If $m > 1$, then $D(1) < 1 - \frac{\varepsilon}{4}$ and we can partition the domain in two sets $S_0 \stackrel{\text{def}}{=} \{1, \dots, k\}$ and $S_1 \stackrel{\text{def}}{=} \{k, \dots, n\}$, by setting

$$k \stackrel{\text{def}}{=} \min \left\{ i \in [n] : \hat{F}(i) < 1 - \frac{\varepsilon}{2} \right\}.$$

By our previous check we know that this quantity is well defined. Further, this implies that $D(S_0) < 1 - \frac{\varepsilon}{4}$ and $D(S_1) > \frac{\varepsilon}{4}$ (the actual values being known up to $\pm \frac{\varepsilon}{4}$). D being ε -close to monotone, it also must be the case that

$$D(S_0) \geq \frac{k}{k+1} \left(1 - \frac{3\varepsilon}{4} \right) - 2\varepsilon \geq \frac{1}{2} \left(1 - \frac{3\varepsilon}{4} \right) - 2\varepsilon \geq \frac{1}{2} - \frac{19\varepsilon}{8},$$

since $\hat{F}(k+1) \geq 1 - \frac{\varepsilon}{2}$ implies $D(\{1, \dots, k\}) + D(k) = D(\{1, \dots, k+1\}) \geq 1 - \frac{3\varepsilon}{4}$. By setting $p \stackrel{\text{def}}{=} D(S_0)$, this means we have access to a Bernoulli random variable with parameter $p \in [\frac{1}{2} - 3\varepsilon, 1 - \frac{\varepsilon}{4}]$. As in the proof of Theorem 8.1 (the constant c being replaced by $\min(\frac{\varepsilon}{4}, \frac{1}{2} - 3\varepsilon)$), one can then leverage this to output with probability at least $1 - \delta$ a uniform random number $s \in [n]$ using $O(\frac{\log n}{\varepsilon} \log \frac{\log n}{\delta})$ samples—and $O(\frac{\log n}{\varepsilon})$ in expectation. \square

Appendix A. On convolutions of distributions over an Abelian finite cyclic group.

DEFINITION A.1. *For any two probability distributions D_1, D_2 over a finite group G (not necessarily Abelian), the convolution of D_1 and D_2 , denoted $D_1 * D_2$, is the distribution on G defined by*

$$D_1 * D_2(x) = \sum_{g \in G} D_1(xg^{-1})D_2(g).$$

²⁰The Dirac distribution δ_1 defined by $\delta_1(1) = 1$, which can be trivially sampled from without the use for any randomness by always outputting 1.

In particular, if G is Abelian, $D_1 * D_2 = D_2 * D_1$.

Fact A.2. The convolution satisfies the following properties:

(i) It is associative:

$$(10) \quad \forall D_1, D_2, D_3, \quad D_1 * (D_2 * D_3) = (D_1 * D_2) * D_3 = D_1 * D_2 * D_3.$$

(ii) It has a (unique) absorbing element, the uniform distribution $\mathcal{U}(G)$:

$$(11) \quad \forall D, \quad \mathcal{U}(G) * D = \mathcal{U}(G).$$

(iii) It can only decrease the total variation:

$$(12) \quad \forall D_1, D_2, D_3, \quad d_{\text{TV}}(D_1 * D_2, D_1 * D_3) \leq d_{\text{TV}}(D_2, D_3).$$

For more on convolutions of distributions over finite groups, see for instance [Dia88] or [BOCLR08].

Fact A.3 ([Mac13b]). Let G be a finite Abelian group, and D_1, D_2 two probability distributions over G . Then, the convolution of D_1 and D_2 satisfies

$$(13) \quad d_{\text{TV}}(\mathcal{U}(G), D_1 * D_2) \leq 2d_{\text{TV}}(\mathcal{U}(G), D_1)d_{\text{TV}}(\mathcal{U}(G), D_2),$$

where $\mathcal{U}(G)$ denotes the uniform distribution on G . Furthermore, this bound is tight.

Claim A.4. For $\varepsilon \in (0, \frac{1}{5})$, there is distribution D on \mathbb{Z}_n such that $d_{\text{TV}}(D, \mathcal{U}) = \varepsilon$, yet $d_{\text{TV}}(D, D * D) = \varepsilon + \frac{3}{4}\varepsilon^2 + O(\varepsilon^3) > \varepsilon$.

Proof. Inspired by—and following—a question on MathOverflow ([Mac13a]). Setting $\delta = 1 - \varepsilon > 1/2$ and taking D_A to be uniform on a subset A of \mathbb{Z}_n with $|A| = \delta n$, one gets that $d_{\text{TV}}(D_A, \mathcal{U}) = \varepsilon$, and yet

$$d_{\text{TV}}(D_A, D_A * D_A) = \frac{1}{2} \|D_A - D_A * D_A\|_1 = \frac{1}{2} \sum_{g \in G} \left| \frac{\mathbb{1}_A(g)}{|A|} - \frac{r(g)}{|A|^2} \right| = 1 - \frac{1}{|A|^2} \sum_{a \in A} r(a),$$

where $r(g)$ is the number of representations of g as a sum of two elements of A (as one can show that $D_A * D_A(g) = |A|^{-2} r(g)$). Fix A to be the interval of length δn centered around $n/2$; that is, $A = \{\ell, \dots, L\}$ with

$$\ell \stackrel{\text{def}}{=} \frac{1 - \delta}{2} n, \quad L \stackrel{\text{def}}{=} \frac{1 + \delta}{2} n.$$

Computing the quantity $\sum_{a \in A} r(a)$ amounts to counting the number of pairs $(a, b) \in A \times A$ whose sum (modulo n) lies in A . For convenience, define $k = (1 - \delta)n$ and $K = \delta n$:

- for $k \leq a \leq K$, $a + b$ ranges from $\ell + a \leq L$ to $L + a \geq \ell + n$, so that modulo n exactly $|A| - |A^c| = (2\delta - 1)n$ elements of A are reached (each of them exactly once);
- for $\ell \leq a < k$, $a + b$ ranges from $\ell + a < L$ to $L + a < \ell + n$, so that the elements of A not obtained are those in the interval $\{\ell, \ell + a - 1\}$ —there are a of them—and again the others are obtained exactly once;
- for $K < a \leq L$, $a + b$ ranges from $L < \ell + a \leq n$ to $L + a \leq K + n$, so that the elements of A not obtained are those in the interval $\{L + a - n + 1, L\}$ —there are $n - a$ of them—and as before the others are hit exactly once.

It follows that

$$\sum_{a \in A} r(a) = (K - k + 1)(2\delta - 1)n + \sum_{a=\ell}^{k-1} (\delta n - a) + \sum_{a=K+1}^L (\delta n - (n - a))$$

(the two sums are equal)

$$\begin{aligned} &= (2\delta - 1)(2\delta - 1)n^2 + \sum_{a=\ell}^{k-1} (\delta n - a) + \sum_{a=K+1}^L (a - (1 - \delta)n) \\ &= (2\delta - 1)^2 n^2 + 2 \cdot \frac{n^2}{8} (7\delta - 3)(1 - \delta) + O(n) \\ &= \frac{1}{4} (4(4\delta^2 - 4\delta + 1) + (7\delta - 3)(1 - \delta)) n^2 + O(n) \\ &= \frac{1}{4} (9\delta^2 - 6\delta + 1) n^2 + O(n) \\ &= \left(1 - 3\varepsilon + \frac{9}{4}\varepsilon^2\right) n^2 + O(n) \end{aligned}$$

and thus

$$\begin{aligned} d_{\text{TV}}(D_A, D_A * D_A) &= 1 - \frac{1}{|A|^2} \sum_{a \in A} r(a) = 1 - \frac{1 - 3\varepsilon + \frac{9}{4}\varepsilon^2}{(1 - \varepsilon)^2} + O\left(\frac{1}{n}\right) \\ &= \varepsilon + \frac{3}{4}\varepsilon^2 + O(\varepsilon^3) \end{aligned}$$

(as $\varepsilon = \omega(1/\sqrt[3]{n})$). □

Appendix B. Formal definitions: Learning and testing. In this appendix, we define precisely the notions of *testing*, *tolerant testing*, *learning*, and *proper learning* of distributions over a domain $[n]$.

DEFINITION B.1 (testing). *Fix any property \mathcal{P} of distributions, and let ORACLE_D be an oracle providing some type of access to D . A q -sample testing algorithm for \mathcal{P} is a randomized algorithm \mathcal{T} which takes as input n , $\varepsilon \in (0, 1]$, as well as access to ORACLE_D . After making at most $q(\varepsilon, n)$ calls to the oracle, \mathcal{T} outputs either *ACCEPT* or *REJECT*, such that the following holds:*

- if $D \in \mathcal{P}$, \mathcal{T} outputs *ACCEPT* with probability at least $2/3$;
- if $d_{\text{TV}}(D, \mathcal{P}) \geq \varepsilon$, \mathcal{T} outputs *REJECT* with probability at least $2/3$.

We shall also be interested in *tolerant testers*—roughly, algorithms robust to a relaxation of the first item above.

DEFINITION B.2 (tolerant testing). *Fix property \mathcal{P} and ORACLE_D as above. A q -sample tolerant testing algorithm for \mathcal{P} is a randomized algorithm \mathcal{T} which takes as input n , $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$, as well as access to ORACLE_D . After making at most $q(\varepsilon_1, \varepsilon_2, n)$ calls to the oracle, \mathcal{T} outputs either *ACCEPT* or *REJECT*, such that the following holds:*

- if $d_{\text{TV}}(D, \mathcal{P}) \leq \varepsilon_1$, \mathcal{T} outputs *ACCEPT* with probability at least $2/3$;
- if $d_{\text{TV}}(D, \mathcal{P}) \geq \varepsilon_2$, \mathcal{T} outputs *REJECT* with probability at least $2/3$.

Note that the above definition is quite general and can be instantiated for different types of oracle access to the unknown distribution. In this work, we are mainly concerned with two such settings, namely the *sampling oracle access* and *cumulative dual oracle access*.

DEFINITION B.3 (standard access model (sampling)). *Let D be a fixed distribution over $[n]$. A sampling oracle for D is an oracle SAMP_D defined as follows: when queried, SAMP_D returns an element $x \in [n]$, where the probability that x is returned is $D(x)$ independently of all previous calls to the oracle.*

DEFINITION B.4 (cumulative dual access model [BKR04, CR14]). *Let D be a fixed distribution over $[n]$. A cumulative dual oracle for D is a pair of oracles $(\text{SAMP}_D, \text{CEVAL}_D)$ defined as follows: the sampling oracle SAMP_D behaves as before, while the evaluation oracle CEVAL_D takes as input a query element $j \in [n]$ and returns the value of the cdf at j . That is, it returns the probability weight that the distribution puts on $[j]$, $D([j]) = \sum_{i=1}^j D(i)$.*

Another class of algorithms we consider is that of (proper) learners. We give the precise definition below.

DEFINITION B.5 (learning). *Let \mathcal{C} be a class of probability distributions and $D \in \mathcal{C}$ be an unknown distribution. Let also \mathcal{H} be a hypothesis class of distributions. A q -sample learning algorithm for \mathcal{C} is a randomized algorithm \mathcal{L} which takes as input $n, \varepsilon, \delta \in (0, 1)$, as well as access to SAMP_D and outputs the description of a distribution $\hat{D} \in \mathcal{H}$ such that with probability at least $1 - \delta$ one has $d_{\text{TV}}(D, \hat{D}) \leq \varepsilon$.*

If in addition $\mathcal{H} \subseteq \mathcal{C}$, then we say \mathcal{L} is a proper learning algorithm.

The exact formalization of what learning a probability distribution means has been considered in Kearns et al. [KMR+94]. We note that in their language, the variant of learning this paper is most closely related to is *learning to generate*.

Appendix C. Omitted proofs. This section contains the proofs of some of the technical lemmata of the paper, omitted for the sake of conciseness.

Proof of (2). Fix a partition \mathcal{I} of $[n]$ into ℓ intervals I_1, \dots, I_ℓ , and let D_1, D_2 be two arbitrary distributions on $[n]$. Recall that $\Psi_{\mathcal{I}}(D)$ is the flattening of distribution D_j (with relation to the partition \mathcal{I}).

$$\begin{aligned} 2d_{\text{TV}}(\Psi_{\mathcal{I}}(D_1), \Psi_{\mathcal{I}}(D_2)) &= \sum_{i=1}^n |\Psi_{\mathcal{I}}(D_1)(i) - \Psi_{\mathcal{I}}(D_2)(i)| = \sum_{k=1}^{\ell} \sum_{i \in I_k} \left| \frac{D_1(I_k)}{|I_k|} - \frac{D_2(I_k)}{|I_k|} \right| \\ &= \sum_{k=1}^{\ell} |D_1(I_k) - D_2(I_k)| = \sum_{k=1}^{\ell} \left| \sum_{i \in I_k} (D_1(i) - D_2(i)) \right| \\ &\leq \sum_{k=1}^{\ell} \sum_{i \in I_k} |D_1(i) - D_2(i)| = \sum_{i=1}^n |D_1(i) - D_2(i)| \\ &= 2d_{\text{TV}}(D_1, D_2). \end{aligned}$$

(We remark that (2) could also be obtained directly by applying the data processing inequality for total variation distance (Fact 2.1) to D_1, D_2 , for the transformation $\Psi_{\mathcal{I}}(\cdot)$.) □

Proof of Corollary 2.5. Let D be ε -close to monotone, and let D' be a monotone distribution such that $d_{\text{TV}}(D, D') = \eta \leq \varepsilon$. By (2), we have

$$(14) \quad d_{\text{TV}}(\Phi_{\alpha}(D), \Phi_{\alpha}(D')) \leq d_{\text{TV}}(D, D') = \eta,$$

proving the last part of the claim (since $\Phi_{\alpha}(D')$ is easily seen to be monotone).

Now, by the triangle inequality,

$$\begin{aligned} d_{\text{TV}}(D, \Phi_\alpha(D')) &\leq d_{\text{TV}}(D, D') + d_{\text{TV}}(D', \Phi_\alpha(D')) + d_{\text{TV}}(\Phi_\alpha(D'), \Phi_\alpha(D)) \\ &\leq \eta + \alpha + \eta \\ &\leq 2\epsilon + \alpha, \end{aligned}$$

where the last inequality uses the assumption on D' and Theorem 2.4 applied to it. \square

REFERENCES

- [ACCL08] N. AILON, B. CHAZELLE, S. COMANDUR, AND D. LIU, *Property-preserving data reconstruction*, *Algorithmica*, 51 (2008), pp. 160–182.
- [ADLS15] J. ACHARYA, I. DIAKONIKOLAS, J. Z. LI, AND L. SCHMIDT, *Sample-Optimal Density Estimation in Nearly-Linear Time*, arXiv:1506.00671 [cs.DS], 2015.
- [AJOS14] J. ACHARYA, A. JAFARPOUR, A. ORLITSKY, AND A. T. SURESH, *Sorting with adversarial comparators and application to density estimation*, in Proceedings of the IEEE International Symposium on Information Theory, 2014, pp. 1682–1686.
- [Bar78] VIC BARNETT, *The study of outliers: Purpose and model*, *J. R. Stat. Soc. Ser. C. Appl. Stat.*, 27 (1978), pp. 242–250.
- [BDKR05] T. BATU, S. DASGUPTA, R. KUMAR, AND R. RUBINFELD, *The complexity of approximating the entropy*, *SIAM J. Comput.*, 35 (2005), pp. 132–150.
- [BDLS17] S. BALAKRISHNAN, S. S. DU, J. LI, AND A. SINGH, *Computationally efficient robust sparse estimation in high dimensions*, in *Proc. Mach. Learn. Res.*, 65 (2017), pp. 169–212.
- [BFR+10] T. BATU, L. FORTNOW, R. RUBINFELD, W. D. SMITH, AND P. WHITE, *Testing Closeness of Discrete Distributions*, arXiv:1009.5397 [cs.DS], 2010.
- [BGJ+12] A. BHATTACHARYYA, E. GRIGORESCU, M. JHA, K. JUNG, S. RASKHODNIKOVA, AND D. P. WOODRUFF, *Lower bounds for local monotonicity reconstruction from transitive-closure spanners*, *SIAM J. Discrete Math.*, 26 (2012), pp. 618–646.
- [Bir87] L. BIRGÉ, *On the risk of histograms for estimating decreasing densities*, *Ann. Stat.*, 15 (1987), pp. 1013–1022.
- [BKR04] T. BATU, R. KUMAR, AND R. RUBINFELD, *Sublinear algorithms for testing monotone and unimodal distributions*, in Proceedings of the Symposium on Theory of Computing, New York, 2004, pp. 381–390.
- [BLR90] M. BLUM, M. LUBY, AND R. RUBINFELD, *Self-testing/correcting with applications to numerical problems*, in Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing, New York, 1990, pp. 73–83.
- [BOCLR08] M. BEN-OR, D. COPPERSMITH, M. LUBY, AND R. RUBINFELD, *Non-Abelian homomorphism testing, and distributions close to their self-convolutions*, *Random Structures Algorithms*, 32 (2008), pp. 49–70.
- [Bra08] Z. BRAKERSKI, *Local Property Restoring*, manuscript, 2008.
- [Can15] C. L. CANONNE, *A Survey on Distribution Testing: Your Data Is Big. But Is It Blue?*, Technical Report TR15-063, <https://ecc.weizmann.ac.il/report/2015/063/>, 2015.
- [CDGR17] C. L. CANONNE, I. DIAKONIKOLAS, T. GOULEAKIS, AND R. RUBINFELD, *Testing shape restrictions of discrete distributions*, *Theory Comput. Syst.*, 2017, pp. 1–59.
- [CDSS13] S.-O. CHAN, I. DIAKONIKOLAS, R. A. SERVEDIO, AND X. SUN, *Learning mixtures of structured distributions over discrete domains*, in Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, S. Khanna, ed. SIAM, Philadelphia, 2013, pp. 1380–1394.
- [CDSS14] S.-O. CHAN, I. DIAKONIKOLAS, R. A. SERVEDIO, AND X. SUN, *Efficient density estimation via piecewise polynomial approximation*, in Proceedings of the 46th Annual ACM Symposium on Theory of Computing, New York, 2014, pp. 604–613.
- [CDSX14] S.-O. CHAN, I. DIAKONIKOLAS, R. A. SERVEDIO, AND S. XIAORUI, *Near-optimal density estimation in near-linear time using variable-width histograms*, in Proceedings of the Annual Conference on Neural Information Processing Systems, 2014, Montreal, 2014, pp. 1844–1852.
- [CFGM13] S. CHAKRABORTY, E. FISCHER, Y. GOLDBIRSH, AND A. MATSLIAH, *On the power of conditional samples in distribution testing*, in Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ACM, New York, 2013., pp. 561–580.

- [CGM11] S. CHAKRABORTY, D. GARCÍA-SORIANO, AND A. MATSLIAH, *Efficient sample extractors for juntas with applications*, in Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, L. Aceto, M. Henzinger, and J. Sgall, eds., Lecture Notes in Comput. Sci. 6755, Springer, New York, 2011, pp. 545–556.
- [CGR13] A. CAMPAGNA, A. GUO, AND R. RUBINFELD, *Local reconstructors and tolerant testers for connectivity and diameter*, in Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques: 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, P. Raghavendra, S. Raskhodnikova, K. Jansen, J. D. P. Rolim, eds., Lecture Notes in Comput. Sci. 8096, Springer, Berlin, 2013, pp. 411–424.
- [CGR16] C. L. CANONNE, T. GOULEAKIS, AND R. RUBINFELD, *Sampling correctors*, in Proceedings of the Conference on Innovations in Theoretical Computer Science, ACM, 2016, pp. 93–102.
- [CR14] C. L. CANONNE AND R. RUBINFELD, *Testing probability distributions underlying aggregated data*, in Proceedings of the International Colloquium on Automata, Languages and Programming, 2014, pp. 283–295.
- [CRS14] C. L. CANONNE, D. RON, AND R. A. SERVEDIO, *Testing equivalence between distributions using conditional samples*, in Proceedings of the Symposium of Discrete Algorithms, SIAM, Philadelphia, 2014, pp. 1174–1192.
- [CSV17] M. CHARIKAR, J. STEINHARDT, AND G. VALIANT, *Learning from untrusted data*, in Proceedings of the Symposium on Theory of Computing, ACM, 2017, pp. 47–60.
- [DDO+13] C. DASKALAKIS, I. DIAKONIKOLAS, R. O'DONNELL, R. A. SERVEDIO, AND L.-Y. TAN, *Learning sums of independent integer random variables*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE, 2013, pp. 217–226.
- [DDS12] C. DASKALAKIS, I. DIAKONIKOLAS, AND R. A. SERVEDIO, *Learning Poisson binomial distributions*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 2012, pp. 709–728.
- [DDS+13] C. DASKALAKIS, I. DIAKONIKOLAS, R. A. SERVEDIO, G. VALIANT, AND P. VALIANT, *Testing k -modal distributions: Optimal algorithms via reductions*, in Proceedings of the Symposium on Discrete Algorithms, SIAM, Philadelphia, 2013, pp. 1833–1852.
- [DDS14] C. DASKALAKIS, I. DIAKONIKOLAS, AND R. A. SERVEDIO, *Learning k -modal distributions via testing*, Theory Comput., 10 (2014), pp. 535–570.
- [Dia88] P. DIACONIS, *Basics of representations and characters*, in Group Representations in Probability and Statistics, Institute of Mathematical Statistics, Hayward, CA, 1988, pp. 5–16.
- [DK14] C. DASKALAKIS AND G. KAMATH, *Faster and sample near-optimal algorithms for proper learning mixtures of Gaussians*, in Proceedings of the Conference on Learning Theory, Barcelona, 2014, pp. 1183–1213.
- [DKK+16] I. DIAKONIKOLAS, G. KAMATH, D. M. KANE, J. LI, A. MOITRA, AND A. STEWART, *Robust estimators in high dimensions without the computational intractability*, in Proceedings of the Symposium on Foundations of Computer Science, IEEE 2016, pp. 655–664.
- [DKK+17] I. DIAKONIKOLAS, G. KAMATH, D. M. KANE, J. LI, A. MOITRA, AND A. STEWART, *Being robust (in high dimensions) can be practical*, in Proc. Mach. 70, Learn. Res., (2017), pp. 999–1008.
- [DKK+18] I. DIAKONIKOLAS, G. KAMATH, D. M. KANE, J. LI, A. MOITRA, AND A. STEWART, *Robustly learning a Gaussian: Getting optimal error, efficiently*, in Proceedings of the Symposium on Discrete Algorithms, 2018, pp. 2683–2702.
- [DKW56] A. DVORETZKY, J. KIEFER, AND J. WOLFOWITZ, *Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator*, Ann. Math. Statist., 27 (1956), pp. 642–669.
- [DL01] L. DEVROYE AND G. LUGOSI, *Combinatorial Methods in Density Estimation*, Springer Ser. Statist., Springer, New York, 2001.
- [DLR77] A. P. DEMPSTER, N. M. LAIRD, AND D. B. RUBIN, *Maximum likelihood from incomplete data via the em algorithm*, J. R. Stat. Soc., Ser. B Stat. Math., 39 (1977), pp. 1–38.
- [GR00] O. GOLDREICH AND D. RON, *On Testing Expansion in Bounded-Degree Graphs*, Technical Report TR00-020, <https://eccc.weizmann.ac.il/report/2000/020/>, 2000.
- [Gre56] U. GRENANDER, *On the theory of mortality measurement*, Scand. Actuar. J., 1956 (1956), pp. 70–96.

- [Haw80] D. M. HAWKINS, *Identification of Outliers*, Springer, Dordrecht, 1980.
- [HBTB14] T. J. HEFLEY, D. M. BAASCH, A. J. TYRE, AND E. E. BLANKENSHIP, *Correction of location errors for presence-only species distribution models*, *Methods Ecol. Evol.*, 5 (2014), pp. 207–214.
- [HR09] P. J. HUBER AND E. M. RONCHETTI, *Robust Statistics*, 2nd ed., Wiley Series in Probability and Statistics, John Wiley & Sons, Hoboken, NJ, 2009.
- [HRRS11] F. R. HAMPEL, E. M. RONCHETTI, P. J. ROUSSEEUW, AND W. A. STAHEL, *Robust Statistics: The Approach Based on Influence Functions*, John Wiley & Sons, Hoboken, NJ, 2011.
- [IH93] B. IGLEWICZ AND D. C. HOAGLIN, *How to Detect and Handle Outliers*, ASQ Press, Milwaukee, WI, 1993.
- [ILR12] P. INDYK, R. LEVI, AND R. RUBINFELD, *Approximating and testing k -histogram distributions in sub-linear time*, in Proceedings of the 31st Symposium on Principles of Database Systems, ACM, New York, 2012, pp. 15–22.
- [IZ89] R. IMPAGLIAZZO AND D. ZUCKERMAN, *How to recycle random bits*, in Proceedings of the 30th Annual Symposium on Foundations of Computer Science, IEEE, 1989, pp. 248–253.
- [JR11] M. JHA AND S. RASKHODNIKOVA, *Testing and reconstruction of Lipschitz functions with applications to data privacy*, in Proceedings of the Symposium on Foundations of Computer Science, Oct. 2011, pp. 433–442.
- [KMR+94] M. KEARNS, Y. MANSOUR, D. RON, R. RUBINFELD, R. E. SCHAPIRE, AND L. SELLIE, *On the learnability of discrete distributions*, in Proceedings of the Symposium on Theory of Computing, ACM, New York, 1994, pp. 273–282.
- [KR94] E. KUSHLEVITZ AND A. ROSN, *A randomness-rounds tradeoff in private computation*, in Advances in Cryptology-CRYPTO '94, Y. G. Desmedt, ed., Lecture Notes in Computer Science 839, Springer, Berlin, Heidelberg, 1994, pp. 397–410.
- [LR02] R. J. A. LITTLE AND D. B. RUBIN, *Statistical Analysis with Missing Data*, 2nd ed., Wiley Series in Probability and Statistics, John Wiley & Sons, Hoboken, NJ, 2002.
- [LRV16] K. A. LAI, A. B. RAO, AND S. VEMPALA, *Agnostic estimation of mean and covariance*, in Proceedings of the Symposium Foundations of Computer Science, IEEE, 2016, pp. 665–674.
- [Mac13a] S. MACIEJ, *Anticoncentration of the convolution of two characteristic functions*, MathOverflow, <http://mathoverflow.net/q/148973> (version: 2013-11-16), 2013.
- [Mac13b] S. MACIEJ, *Convergence rate of the convolution of almost uniform measures on \mathbb{Z}_p* , MathOverflow, <http://mathoverflow.net/q/148779> (version: 2013-11-13), 2013.
- [Mas90] P. MASSART, *The tight constant in the Dvoretzky–Kiefer–Wolfowitz inequality*, *Ann. Probab.*, 18 (1990), pp. 1269–1283.
- [Pan08] L. PANINSKI, *A coincidence-based test for uniformity given very sparsely sampled discrete data*, *IEEE Trans. Inform. Theory*, 54 (2008), pp. 4750–4755.
- [PMC08] S. PANZERI, C. MAGRI, AND L. CARRARO, *Sampling bias*, *Scholarpedia*, 3 (2008), 4258, revision 91742.
- [RTS00] J. RADHAKRISHNAN AND A. TA-SHMA, *Bounds for dispersers, extractors, and depth-two superconcentrators*, *SIAM J. Discrete Math.*, 13 (2000), pp. 2–24.
- [Rub87] D. B. RUBIN, *Multiple Imputation for Nonresponse in Surveys*, John Wiley & Sons, Hoboken, NJ, 1987.
- [Sch97] JOSEPH L. SCHAFER, *Analysis of Incomplete Multivariate Data*, CRC Press, Boca Raton, FL, 1997.
- [SL82] P. W. SIGNOR AND J. H. LIPPS, *Sampling bias, gradual extinction patterns and catastrophes in the fossil record*, *Geol. Soc. Amer. Spec. Pap.* 190 (1982), pp. 291–296.
- [SS10] M. SAKS AND C. SESHADHRI, *Local monotonicity reconstruction*, *SIAM J. Comput.*, 39 (2010), pp. 2897–2926.
- [STP07] M. SAAR-TSECHANSKY AND F. PROVOST, *Handling missing values when applying classification models*, *J. Mach. Learn. Res.*, 8 (2007), pp. 1623–1657.
- [SV98] A. SAHAI AND S. VADHAN, *Manipulating statistical difference*, in Randomization Methods in Algorithm Design, P. Pardalos, S. Rajasekaran, and J. Rolim, eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, 1998, pp. 251–270.
- [SV03] P. D. SENESE AND J. A. VASQUEZ, *A unified explanation of territorial conflict: Testing the impact of sampling bias, 1919–1992*, *Int. Stud. Q.*, 47 (2003), pp. 275–298.
- [Tuk60] J. W. TUKEY, *A Survey of Sampling from Contaminated Distributions*, in Contributions to Probability and Statistics, I. Olkin, ed., Stanford University Press, Stanford, CA, 1960, pp. 448–485.

- [Vad12] S. P. VADHAN, *Pseudorandomness*, Found. Trends Theor. Comput. Sci., 7 (2012), pp. 1–336.
- [Val15] P. VALIANT, private communication, May 2015.
- [VV11] G. VALIANT AND P. VALIANT, *The power of linear estimators*, in Proceedings of the Symposium on Foundations of Computer Science, 2011, pp. 403–412.
- [Yek10] S. YEKHANIN, *Locally Decodable Codes*, Now Publishers, Hanover, MA, 2010.