Tight Simulation of a Distribution Using Conditional Samples

Tomer Adar
Technion – Israel Institute of Technology, Haifa, Israel

Abstract

We present an algorithm for simulating a distribution using prefix conditional samples (Adar, Fischer and Levi, 2024), as well as "prefix-compatible" conditional models such as the interval model (Cannone, Ron and Servedio, 2015) and the subcube model (CRS15, Bhattacharyya and Chakraborty, 2018). The sample complexity is $O(\log^2 N/\varepsilon^2)$ prefix conditional samples per query, which improves on the previously known $\tilde{O}(\log^3 N/\varepsilon^2)$ (Kumar, Meel and Pote, 2025). Moreover, our simulating distribution is $O(\varepsilon^2)$-close to the input distribution with respect to the Kullback-Leibler divergence, which is stricter than the usual guarantee of being $O(\varepsilon)$-close with respect to the total-variation distance.

We show that our algorithm is tight with respect to the highly-related task of estimation: every algorithm that is able to estimate the mass of individual elements within $(1 \pm \varepsilon)$-multiplicative error must make $\Omega(\log^2 N/\varepsilon^2)$ prefix conditional samples per element.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Distribution learning, Distribution simulation, Probability estimation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.32

Category RANDOM

Related Version *Full Version*: https://arxiv.org/pdf/2506.18444

1 Introduction

Distribution testing, formally initiated by [4, 3], is a branch of property testing where the goal is distinguishing with high probability between distributions that have some property (for example, uniformity) and distributions that are far from any distribution with this property (for example, extremely biased distributions). "Farness" (or closeness) of distributions is usually determined by comparing the total-variation distance to a threshold parameter, usually denoted by ε.

The sampling model of distribution testing allows the algorithm to obtain a sequence of independent samples from its input distribution. One core result of distribution testing in the sampling model is uniformity testing over a domain of size N, at the cost of $\Theta(\sqrt{N}/\varepsilon^2)$ [15] samples. Other elementary properties, such as identity to an explicit distribution [3] and equivalence to an unknown distribution [11], require polynomial number of samples as well (but still sublinear).

The sampling model of distribution testing is ineffective for high-dimensional domains, such as the set of binary strings of length n, since it requires a polynomial number of samples in the domain size even for simple core properties, which is exponential in the dimension n. For obtaining sub-polynomial testing algorithms, we strengthen the model.

The conditional sampling model, independently introduced by [9] and [10], allows the algorithm to choose a set A and draw a sample from the input distribution when conditioned on belonging to A. Various works focus on restricted forms of the conditional sampling model, where only sets of a restricted form are eligible for conditioning. We recall a few of them.

Subcube conditions

The subcube conditional model defined by [5] (see further works: [7, 12, 1, 13]) is defined for distributions over product sets: the domain is Σ^n for some alphabet Σ, and the sets eligible for condition are the cartesian product of their projections on individual characters. More formally, a subcube condition is a set of the form $A = \prod_{i=1}^{n} A_i$ for $A_1, \ldots, A_n \subseteq \Sigma$. Conceptually, the subcube conditional model resembles the "SELECT ... WHERE"-form of SQL queries.

Prefix conditions

In this paper, we focus on prefix conditions [1] (implicitly in [13, 5, 10]). Prefix conditions are a restricted form of subcube conditions in which there exists some k for which $A_1, \ldots, A_{k-1}$ are all singletons and $A_k, \ldots, A_n = \Sigma$. Note that we can identify every condition set $\{a_1\} \times \cdots \times \{a_{i-1}\} \times \Omega^{n-|w|}$ with the prefix $w = a_1 \ldots a_{i-1}$ (of length $i-1$) itself.

Additional works define various condition models such as *pair conditions* [9] (the condition set must have two elements), *interval conditions* [9] ($A = \{a, \ldots, b\} \subseteq \{1, \ldots, N\}$) and *coordinate conditions* [6] (we fix all symbols of a string but one). For binary string domains ($\Omega = \{0,1\}^n$), prefix conditions can be seen as prefix-interval condition sets (through the encoding $x \in \{0,1\}^n \to 1 + \sum_{i=1}^{n} 2^{n-i} x_i$ [1]) and coordinate conditions can be seen as subcube-pair condition sets.

1.1 Estimation and simulation of a distribution

In the string model of property testing (for example [16, 17]), the algorithm can choose an index to query and reveal the bit in that index. In other words, it obtains certain (undoubted) information about its input. In contrast, in all sampling models, the access to the input is indirect: the algorithm only obtains "behavioral" information about its input.

There are two concepts of "bridging" between sampling and querying. The first is *estimation*: given an element $x \in \Omega$, the algorithm estimates $\mu(x)$ (with high probability) within a reasonable error range using samples. The second is *simulation*: the algorithm constructs an explicit distribution $\tilde{\mu}$, which is hopefully close to the input distribution μ, and then queries it instead of the input distribution. Since approximating an entire distribution is too expensive, we allow a lazy construction of $\tilde{\mu}$, starting with "$\tilde{\mu}$ can be any distribution over Ω" and then gradually adding details required to answer queries.

Estimation and simulation are highly related tasks. While estimation demands "bounded worst-case error (for most input elements)", simulation of a close distribution can be seen as a "bounded expected estimation error (over the choice of the input element)". However, they are not fully compatible with each other:

- Assume that we can estimate all probabilities with a small error. Estimations can be biased, and therefore, the estimated probability masses do not necessarily sum to 1. Moreover, even if they do, the specific implementation of the estimator might be unable to *sample* the distribution implied by the individual estimations.
- Assume that we can access a distribution $\tilde{\mu}$ which is close to μ with respect to some divergence measure. For common divergences such as the KL-divergence, it might be the case that most elements are poorly estimated, but the estimation errors cancel each other, which results in a relatively low divergence.

One algorithmic paradigm which is compatible with both estimation and simulation is the distribution tree[1], or "slice and dice". First, we define a tree of sets eligible for conditioning by the model, where the root holds the whole domain Ω and every non-leaf node has two children whose disjoint union is the set it holds. Whenever an estimation $\mu(x)$ is required, we estimate the conditional mass $\mu(\mathsf{CHILD}|\mathsf{PARENT})$ in all nodes in the path from the root to the leaf containing the singleton $\{x\}$. To obtain the estimation, we use the chain rule: if $\{x\} = A_k \subseteq A_{k-1} \subseteq \cdots A_1 \subseteq A_0 = \Omega$, then $\Pr_\mu[x] = \prod_{i=1}^{k} \Pr_\mu[A_i|A_{i-1}]$. The estimation algorithm uses $\tilde{\mu}(x) = \prod_{i=1}^{k} \mathrm{est}(\Pr_\mu[A_i|A_{i-1}])$.

In [13] there is an explicit "slice-and-dice" algorithm for the subcube conditional sampling model. The algorithm can estimate $\mu(x)$ within $(1 \pm \varepsilon)$-factor for every x, unless one or more of its marginals is too close to 0 or to 1. The overall mass of these xs is bounded by $O(\varepsilon)$, which implies that the simulating distribution is $O(\varepsilon)$-close to μ with respect to the total-variation distance. The cost for every query is $\tilde{O}(n^3/\varepsilon^2)$ subcube conditional samples.

In this work we reduce the cost of the "slice and dice" method to $O(n^2/\varepsilon^2)$ conditional samples per query and improve the accuracy of the simulating distribution to be $O(\varepsilon^2)$-close with respect to the Kullback-Leibler divergence. Note that by Pinsker's inequality (see Lemma 59) it is never worse than $O(\varepsilon)$-total-variation-close.

The KL-divergence has two advantages over the total-variation distance for intermediate analysis: its propagation over multiple dimensions is tight (compare the equality in Lemma 62 to the corresponding upper bound for total-variation distance, for example in [5]), and it provides a refined divergence bound of indicator estimations (see Lemma 31 vs. the well-known $O(1/\sqrt{m})$-bound for total-variation distance), allowing for propagating over a higher dimension subject to the constraint $\mathrm{KL} = O(\mathrm{poly}(\varepsilon))$. We provide our algorithm in the prefix conditional model, but it can be implemented in every conditional model that resembles the "slice and dice" behavior (see Section 3 for more details).

Since explicitly learning a whole distribution is inevitably expensive (as we would have to store information about every element in the domain), we prefer to use local learning. Instead of storing a whole distribution, we provide an oracle access to the learned object, and store only the information needed to describe the queried parts. When processing a query, we compute only the (missing) parts needed to respond.

Notations

We refer to a *learning algorithm* whose input is an oracle access to a distribution μ, also called the *simulated distribution*, and its output is an oracle access to an output distribution $\tilde{\mu}$, which is called the *simulating distribution* (of μ).

1.2 Usage example

The distance estimation algorithms of [13, 2] consist of two modules: first, an estimation procedure $x \to \mathrm{est}(\mu(x))$, which we call a *mass query*, and second, an unconditional sampling algorithm that obtains the approximated mass of its samples and uses them to estimate the distance between the input distributions. While the first module requires a different logic for implementation in various models, the second module is fixed and stated in the following proposition.

[1] Note that a specific *implementation* inspired by this paradigm is not necessarily compatible with both.

▶ **Proposition 1** ([2], implicit in [13]). *Consider the following oracle for accessing a distribution μ: the oracle draws $x \sim \mu$, and returns a pair of the form $(x, (1 \pm O(\varepsilon))\mu(x))$. There exists an algorithm whose input is such an oracle access to two distributions μ and τ that makes $O(1/\varepsilon^2)$ oracle calls to estimate $d_{\mathrm{TV}}(\mu, \tau)$ within a $\pm\varepsilon$-additive error with probability at least $2/3$.*

1.3 Our results

The main result of this paper is the following theorem.

▶ **Theorem 2** (Informal statement of Theorem 33). *For every sufficiently small $\varepsilon > 0$ and sufficiently large $n \geq 1$, there exists an algorithm for simulating an input distribution over $\{0,1\}^n$ using $O(n^2/\varepsilon^2)$ prefix conditional samples per mass query and $O(n^2/\varepsilon^2)$ prefix conditional samples for sampling the simulating distribution. Moreover, the expected KL-divergence of the simulating distribution from the input distribution is $O(\varepsilon^2)$.*

Additionally, we show that any estimation algorithm must make $\Omega(n^2/\varepsilon^2)$ prefix conditional samples per query. The exact meaning of an "estimation algorithm" is detailed in the formal statement of Theorem 49.

▶ **Theorem 3** (Informal statement of Theorem 49). *Every estimation algorithm must make at least $\Omega(n^2/\varepsilon^2)$ prefix conditional samples per query.*

In Section 4, we provide a technical overview and the proof roadmap.

2 Preliminaries

In this section we formally define the notation scheme of this paper, as well as a few definitions more specific for this work. See Appendix B for formal definition of widely used distribution notations (such as Bernoulli) and for formal statements of well known facts (such as asymptotic bounds for $\ln x$ and Pinsker's inequality).

2.1 Basic notations

▶ **Definition 4** (Conditional distribution). *Let μ be a distribution over Ω and let $A \subseteq \Omega$ be a set. The* conditional distribution $\mu|^A$ *is defined as the distribution where $\Pr[x] = \frac{\Pr_\mu[x]}{\Pr_\mu[A]}$ for every $x \in A$ and $\Pr[x] = 0$ for every $x \notin A$. If $\Pr_\mu[A] = 0$, then the conditional distribution $\mu|^A$ is undefined.*

▶ **Definition 5** (Projected distribution, restricted distribution). *Let μ be a distribution over $\Omega = \prod_{i=1}^k \Omega_i$. For a non-empty set $I \subseteq \{1, \ldots, k\}$, the* projected distribution μ_I *is defined as the distribution over $\prod_{i \in I} \Omega_i$ for which $\Pr[x] = \Pr_{z \sim \mu}[\forall i \in I : z_i = x_i]$ for every $x \in \prod_{i \in I} \Omega_i$. In other words, it is the distribution obtained by drawing a sample from μ and* restricting *it to the entries in I's indexes.*

The overloaded notation $\mu|_I^A$ is interpreted as $(\mu|^A)_I$. In other words, we first sample according to the condition A, and then restrict the result to I's entries.

▶ **Definition 6** (Product of distributions). *Let $\mu_1, \ldots, \mu_k$ be distributions over $\Omega_1, \ldots, \Omega_k$, respectively. The* product distribution $\mu = \prod_{i=1}^k \mu_i$ *is the distribution over $\Omega = \prod_{i=1}^k \Omega_i$ where, for every $(a_1, \ldots, a_k) \in \Omega$, $\mu((a_1, \ldots, a_k)) = \prod_{i=1}^n \mu_i(a_i)$.*

▶ **Definition 7** (Total-variation distance). *Let μ and τ be two distributions over a domain Ω. Their* total-variation divergence *is $d_{\mathrm{TV}}(\mu,\tau) = \frac{1}{2}\sum_{x\in\Omega}|\mu(x)-\tau(x)|$.*

▶ **Definition 8** (Kullback-Leibler divergence). *Let μ and τ be two distributions over a domain Ω. The* KL-divergence *of μ from τ is $D_{\mathrm{KL}}(\mu \parallel \tau) = \mathrm{E}_{x\sim\mu}[\log_2(\mu(x)/\tau(x))]$.*

For convenience, we abuse the notation to define KL-divergence of probabilities:

▶ **Definition 9** (KL-divergence of probabilities (overloaded notation)). *For $p,q\in[0,1]$, we use the notation $D_{\mathrm{KL}}(p,q) = D_{\mathrm{KL}}(\mathrm{Ber}(p) \parallel \mathrm{Ber}(q)) = p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}$.*

2.2 Distribution testing models

A testing algorithm must access its input using oracles. Below we define two oracles for distribution testing.

▶ **Definition 10** (The sampling oracle). *Given an unknown input distribution μ over a set Ω, a call to the oracle results in a sample $x\in\Omega$ that distributes according to μ, independently of past calls to the oracle.*

▶ **Definition 11** (The conditional sampling oracle). *Given an unknown input distribution μ over a set Ω, the input of the oracle is a non-empty explicit subset $A\subseteq\Omega$ and the output is a sample $x\in\Omega$ that distributes like $\mu|^A$. If the mass of A in μ is zero, then the behavior of the oracle is undefined (it is only guaranteed that $x\in A$ with probability 1).*

▶ **Definition 12** (Prefix condition sets). *Let $\Omega=\Sigma^n$ be a domain. A set A is a* prefix condition *if there exists a string $w\in\bigcup_{i=0}^{n}\Sigma^i$ for which $A=\prod_{i=1}^{|w|}\{w_i\}\times\Sigma^{n-|w|}$.*

2.3 Algorithmic notions

In this subsection we formally define the notions of estimation and simulation of distributions.

Estimation algorithm must be able to correctly estimate the mass of *most* elements within a given *multiplicative factor*.

▶ **Definition 13** ((ε,c)-estimation algorithm). *Consider an algorithm whose input is a pair (μ,x), where μ is a distribution and x is an element in the domain of μ. For given parameters $0<\varepsilon<1$ and $0<c<1$, such an algorithm is an* (ε,c)-estimation algorithm *if for every input distribution μ there exists a set G_μ for which $\mu(G_\mu)\geq 1-c$ such that, for every $x\in G_\mu$, with probability at least $2/3$, the output of the algorithm for the input (μ,x) is a number in the range $(1\pm\varepsilon)\mu(x)$.*

A simulation algorithm must be defined through an interaction interface rather than a one-time call. It consists of a one-time initialization and a sequence of allowed interactions, "sample" and "query". After each interaction, the internal state of the simulation can mutate (in particular, the simulation can track its interaction history).

▶ **Definition 14** (Distribution simulation interface). *For a domain Ω, the interface of a* distribution simulation *consists of three procedures:*

- *Initialization: the input is a distribution μ and any additional parameters.*
- *Query: the input is an element $x\in\Omega$, and the output is a number $p\in[0,1]$.*
- *Sample: the output is an element $x\in\Omega$ and a number $p\in[0,1]$.*

▶ **Definition 15** (Interaction notation). *We formalize the q-round interaction of the caller with the simulation algorithm (Definition 14) as:*

- *A* request *sequence* $(\text{init}(\mu); X_1, \ldots, X_q)$*, where* $X_i \in \Omega \cup \{\perp\}$ *represents a query request (if* $X_i \in \Omega$*) or a sample request (*$X_i = \perp$*).*
- *A* response *sequence* $((\hat{X}_i, \hat{P}_i))_{1 \le i \le q}$*, where* $\hat{X}_i$ *represents the sampled element (for a sample request) or the queried element* X_i *(for a query request), and* $\hat{P}_i$ *represents the p-output.*

The simulation interface lacks of constraints ensuring that the output "looks like a simulation", which we have to formalize separately. Ideally, an implementation of the simulation interface secretly chooses a distribution $\tilde{\mu}$ (which is hopefully a good representation of μ) at the initialization phase, and then responds to sample and query interactions by sampling and querying the chosen $\tilde{\mu}$.

▶ **Definition 16** (The secret-distribution constraint). *An algorithmic implementation $\mathcal{A}$ for the distribution simulation interface follows the* secret-distribution constraint *if for every input distribution μ there exists a distribution $\mathcal{A}_\mu$ over distributions such that the interaction can be seen as drawing a secret distribution $\tilde{\mu} \sim \mathcal{A}_\mu$ in the initialization phase and then responding to sample and query requests through $\tilde{\mu}$. More precisely, given the request sequence $(\text{init}(\mu); X_1, \ldots, X_q)$, the probability to obtain the response $((\hat{X}_1, \hat{P}_1), \ldots, (\hat{X}_q, \hat{P}_q))$ is exactly:*

$$\sum_{\tilde{\mu}:\bigwedge_{i=1}^{q}(\tilde{\mu}(\hat{X}_i)=\hat{P}_i)} \left(\Pr_{\mathcal{A}_\mu}[\tilde{\mu}] \times \prod_{i=1}^{q} \begin{cases} 1 & X_i = \hat{X}_i \text{ (query)} \\ \tilde{\mu}(\hat{X}_i) & X_i = \perp \text{ (sample)} \\ 0 & X_i \neq \hat{X}_i, \perp \text{ (violation)} \end{cases} \right)$$

▶ **Definition 17** ((D, δ)-simulation of a distribution). *For a given divergence measure D and a parameter δ, an algorithmic implementation $\mathcal{A}$ of the distribution simulation interface is a* (D, δ)-simulation *if:*

- *Simulation behavior: $\mathcal{A}$ follows the secret-distribution constraint.*
- *Closeness: for every μ, regarding the distribution $\mathcal{A}_\mu$ (over distributions) required by the secret-distribution constraint,* $\mathrm{E}_{\tilde{\mu} \sim \mathcal{A}_\mu}[D(\tilde{\mu}|\mu)] \le \delta$.

3 Simulating conditional samples using prefix conditions

In this section we provide a conceptual reduction from conditional models compatible with the distribution-tree paradigm, which we formally define below, to the prefix conditional model.

First, we define a *breakdown tree*, which is the underlying structure of a distribution tree.

▶ **Definition 18** (A breakdown tree). *Let Ω be a domain. A* breakdown tree *is the following combinatorial tree structure:*

- *All leaves have the same depth.*
- *Every node holds a subset of the domain.*
- *A leaf can hold either the empty set or a singleton.*
- *The root holds the domain set Ω.*
- *Every non-leaf node has two children, arbitrarily named "left" and "right".*
- *Every non-leaf node holds the disjoint union of the subsets held by its children.*

▶ **Observation 19.** *Let $\mathcal{T}$ be a breakdown tree over Ω. For every $x \in \Omega$, there exists a single leaf that holds the singleton $\{x\}$.*

A conditional sampling model is *compatible with the distribution-tree paradigm* if for every set Ω which is valid as a domain of distributions in this model, there exists a breakdown tree in which every subset associated with a node is a valid condition set in the model. It is immediate to see that the subcube conditional model, the interval conditional model and the prefix conditional model are compatible with the distribution-tree paradigm, whereas the pair conditional model[2] is not.

We extend the structure of a breakdown tree to define distribution trees.

▶ **Definition 20** (A distribution tree). *Let Ω be a domain. A* distribution tree *consists of a breakdown tree and additionally:*

- *Every edge holds a probability.*
- *For every non-leaf node, the sum of its (two) outgoing edges is* 1.
- *The probability of an edge from a node holding a non-empty set to a child holding the empty set is* 0.
- *The probability of an edge from a node holding the empty set to its left child is* 1.

▶ **Definition 21** (Representation). *A distribution tree $\mathcal{T}$ of height ℓ over Ω* represents *a distribution μ over Ω if, for every $x \in \Omega$, considering the tree path $\Omega = A_0 \supseteq A_1 \supseteq \cdots \supseteq A_{\ell-1} \supseteq A_\ell = \{x\}$ and the weights $p_1, \ldots, p_\ell$ associated with the edges $A_0 \to A_1, \ldots, A_{\ell-1} \to A_\ell$, it holds that $\mu(x) = \prod_{i=1}^{\ell} p_i$.*

▶ **Observation 22.** *Let $\mathcal{T}$ be a breakdown tree over a domain Ω. There exists a bijection between distribution trees with the same structure of $\mathcal{T}$ and distributions over Ω such that every tree represents the distribution it is mapped to. Moreover, the mapped tree can be computed directly by associating every edge $A \to B$ ($A \supseteq B$) with the probability $\Pr_{z\sim\mu}[z \in B | z \in A]$.*

Observe that, given a breakdown tree of edge-height ℓ for a conditional model over a domain Ω, every condition associated with a node can be seen as a prefix condition in $\{0,1\}^\ell$, where the encoding of elements $x \in \Omega$ into elements in $\{0,1\}^\ell$ corresponds to the sequence of left and right edges in the path from the root to x.

Therefore, every prefix conditional model algorithm can be directly executed in every conditional model that has a breakdown tree, where the n parameter (for prefix algorithms over $\{0,1\}^n$) is the edge-height of the breakdown tree.

4 Overview

4.1 The prefix conditional model

We re-define the notions of a breakdown tree and a distribution tree to specifically match the behavior of prefix conditions.

▶ **Definition 23** (Prefix domain). *For $n \geq 1$, let $\{0,1\}^{<n} = \bigcup_{i=0}^{n-1} \{0,1\}^i$ be the set of all true prefixes of strings in $\{0,1\}^n$.*

Observe that there is a natural bijection between strings in $\{0,1\}^{<n}$ and non-singleton prefix conditions in $\{0,1\}^n$: a string $w \in \{0,1\}^{<n}$ corresponds to the prefix condition set $\{w\} \times \{0,1\}^{n-|w|}$. Moreover, the prefix conditional model has exactly one breakdown tree: the tree in which the non-leaf nodes are the sets corresponding to binary strings of length strictly less than n, the leaf nodes are the elements of $\{0,1\}^n$, and the children of a non-leaf node w correspond to the prefixes $w0$ and $w1$.

[2] The pair conditional model is defined in [9]. Beside the whole domain, the valid condition sets are all sets with exactly two elements.

▶ **Definition 24** (Marginal tree). *Let $\Omega = \{0,1\}^n$. A marginal tree consists of the (unique) breakdown tree and a function $\{0,1\}^{<n} \to [0,1]$ representing the probabilities associated with the one-edges (the edges of the form $w \to w1$).*

▶ **Observation 25** (Representation). *Given the marginal function $f : \{0,1\}^{<n} \to [0,1]$, the marginal tree defines a distribution in which, for every $x \in \{0,1\}^n$,*

$$\Pr[x] = \prod_{i=1}^{n}(x_i f(x_{1,\ldots,i-1}) + (1-x_i)(1-f(x_{1,\ldots,i-1}))) = \prod_{i=1}^{n}\begin{cases} f(x_{1,\ldots,i-1}) & x_i = 1 \\ 1-f(x_{1,\ldots,i-1}) & x_i = 0 \end{cases}$$

We use the specific structure of the only breakdown tree in the prefix conditional model to introduce a notation for marginal probabilities.

▶ **Definition 26** (Overloaded notation for marginals). *The marginal function $f : \{0,1\}^{<n} \to [0,1]$ represents a single distribution μ over $\{0,1\}^n$. For a prefix w of length $i-1$, we use the notation $\mu(x_i = 1|x_{1,\ldots,i-1} = w)$, its short form $\mu(x_i = 1|w)$ and its shorter form $\mu(1|w)$, to denote $f(w)$. Analogously, $\mu(x_i = 0|x_{1,\ldots,i-1} = w)$, $\mu(x_i = 0|w)$ and $\mu(0|w)$ denote $1 - f(w)$.*

In both the upper bound and the lower bound analysis, we use the chain rule for distributions over strings.

To *query* the mass of $x \in \{0,1\}^n$ with respect to μ, we multiply its marginal probabilities:

$$\mu(x) = \prod_{i=1}^{n}\mu(x_i|x_{1,\ldots,i-1}) = \prod_{i=1}^{n}(x_i f(x_{1,\ldots,i-1}) + (1-x_i)(1-f(x_{1,\ldots,i-1})))$$

To *estimate* $\mu(x)$, we multiply the estimations of its marginals.

$$\mathrm{est}(\mu(x)) = \prod_{i=1}^{n}\mathrm{est}(\mu(x_i|x_{1,\ldots,i-1})) = \prod_{i=1}^{n}(x_i\mathrm{est}(f(x_{1,\ldots,i-1})) + (1-x_i)(1-\mathrm{est}(f(x_{1,\ldots,i-1}))))$$

Note that the estimations of the marginals can be fully independent.

In parts of our analysis we use a weaker form of the prefix conditional sampling oracle.

▶ **Definition 27** (The marginal prefix sampling oracle). *Let μ be a distribution over $\Omega = \{0,1\}^n$. The input of a* marginal prefix sampling oracle *is a prefix condition $\{w\} \times \{0,1\}^{n-|w|}$ (for some $w \in \{0,1\}^{<n}$, and its output is a single bit that distributes like $\mu|_{|w|+1}^{w\times\{0,1\}^{n-|w|}}$.*

In other words, a marginal prefix oracle call draws a prefix conditional sample and then ignores all but the first free bit of the sample.

We can convert any prefix conditional sampling (including an unconditional sampling, in which the condition corresponds to the empty prefix) to a (random) sequence of marginal prefix samples whose length is the same as the number of free bits. Let $X_1, \ldots, X_n$ be the random variables for which the result sample is the concatenation $x = X_1 \cdots X_n$. Let $w \in \{0,1\}^{<n}$ of length $k-1$ ($1 \le k \le n$) be the prefix to condition on. For $1 \le i \le k-1$, X_i is the fixed value w_i. For $k \le i \le n$, once $X_1, \ldots, X_{i-1}$ are determined, we can sample X_i as the first bit of a sample conditioned on the prefix $X_1 \cdots X_{i-1}$.

4.2 The upper bound (Section 5)

We show that we can estimate a single marginal probability using m samples such that the expected KL-divergence of the estimation from the true marginal probability is bounded by $1/m$. By the chain rule, if we estimated all $2^n - 1$ marginal probabilities independently,

then the expected KL-divergence of the result distribution from the input distribution would be bounded by n/m. We choose $m = n/\varepsilon^2$ to have an $O(\varepsilon^2)$-D_{KL}-close distribution in expectation at the cost of $O(n/\varepsilon^2)$ prefix samples per marginal estimation. Since the marginal estimations are fully independent, given an element x we can estimate only its n marginals at the cost of $O(n^2/\varepsilon^2)$ prefix samples.

4.3 The lower bound (Section 6, Appendix A)

Recall that the input for the algorithm is pair of (1) a distribution over $\Omega = \{0,1\}^n$ and (2) an element $x \in \Omega$, and the output should be a number in the range $(1 \pm \varepsilon)\mu(x)$ with probability $2/3$.

We first describe the construction. We define two internal parameters, $\delta = \sqrt{2}\varepsilon/\sqrt{n}$ and $r = 8/\sqrt{n}$. For every $w \in \{0,1\}^{<n}$ we draw $s_w \in \{+1,-1\}$ uniformly and independently. We define the two distributions μ and μ' through its marginals. For every $w \in \{0,1\}^n$, we define:

- $\mu(1|w) = \frac{1}{2}(1 + s_w\delta)$, so that $\Pr_\mu[x] = 2^{-n}\prod_{i=1}^{n}(1 + (-1)^{1-x_i}s_{x_{1,\ldots,i-1}}\delta)$.
- $\mu'(1|w) = \frac{1}{2}(1 - s_w r)$, so that $\Pr_{\mu'}[x] = 2^{-n}\prod_{i=1}^{n}(1 + (-1)^{x_i}s_{x_{1,\ldots,i-1}}r)$.

We define two distributions over inputs, based on the choice of s_w variables:

- D_{yes}: the input provided to the algorithm is (μ, x), where x is drawn uniformly.
- D_{no}: the input provided to the algorithm is (μ, x), where x is drawn according to μ'.

The construction guarantees different concentration ranges for $\mu(x)$, as stated in the following lemma.

▶ **Lemma 28** (See Lemma 38, 43, 44). *For every sufficiently large $n \geq N_0$ and sufficiently small $0 < \varepsilon \leq \varepsilon_0$ there exist p_L and p_H for which:*

- $(1+\varepsilon)p_L < (1-\varepsilon)p_H$.
- *If (μ, x) is drawn from D_{yes}, then $\mu(x) \geq p_H$ with probability at least $1 - e^{-7.95}$.*
- *If (μ, x) is drawn from D_{no}, then $\mu(x) \leq p_L$ with probability at least $e^{-7.38}$.*

To prove a lower bound, we recall the meaning of a good estimation algorithm:

- The set of "good" elements has mass $1 - O(\varepsilon)$ according to the input distribution.
- Given a "good" element x, the output is in the range $(1 \pm \varepsilon)\mu(x)$ with probability at least $\frac{2}{3}$.

Note that the success probability in second requirement can be strengthen to $1 - c$ (instead of $2/3$) at the cost of $O(\log(1/c))$, which is asymptotically ignorable when c is a constant.

Consider an estimation algorithm $\mathcal{A}$ that can estimate $\mu(x)$ within a $(1 \pm \varepsilon)$-factor with probability at least $1 - e^{-20}$, unless x belongs to a small set "bad" elements whose mass is bounded by e^{-20}. In this setting,

$$\begin{aligned} d_{\mathrm{TV}}(\mathcal{A}(D_{\mathrm{yes}}), \mathcal{A}(D_{\mathrm{no}})) &\geq \Pr[\mathcal{A}(D_{\mathrm{yes}}) \geq (1-\varepsilon)p_H] - \Pr[\mathcal{A}(D_{\mathrm{yes}}) \geq (1-\varepsilon)p_H] \\ &\geq (e^{-7.38} - e^{-7.95}) - 4e^{-20} \geq e^{-8.25} \end{aligned}$$

Therefore, if an algorithm makes too few queries to distinguish between D_{yes} and D_{no} with total-variation distance at least $e^{-8.25}$, then it cannot be a good estimation algorithm. It remains to describe the lower bound for distinguishing between D_{yes} and D_{no} using prefix queries.

Let I_x be the set of prefixes (elements in $\{0,1\}^{<n}$) that are not prefixes of x. When x is explicitly given as an input, $\mu(x)$ is fully determined by the random sequence $(s_{x_{1,\ldots,i-1}})_{1 \leq i \leq n}$, which may depend on the choice of x. Also, note that if x is drawn uniformly, then $\mu(x)$ is independent of the assignment of all s_ws for $w \in I_x$. We observe that this independence is preserved even if x is drawn from μ' (Lemma 36).

Based on this analysis, we can define a random sequence $p_i = \mu(x_{1,\ldots,i}|x_{1,\ldots,i-1})$ (for $1 \le i \le n$).

- For D_{yes}, $p_i = (1+\delta)/2$ with probability $1/2$ and otherwise $p_i = (1-\delta)/2$.
- For D_{no}, $p_i = (1+\delta)/2$ with probability $(1-r)/2$ and otherwise $p_i = (1-\delta)/2$.

Clearly, distinguishing between D_{yes} and D_{no} cannot be easier than distinguishing between the sequences $(p_1, \ldots, p_n)$ drawn from D_{yes} and from D_{no}.

To show a lower bound for distinguishing between the p_i-sequence inputs, we use the following model that has been introduced in [14]: the input is a sequence of n coins, and in every step, the algorithm can choose a coin and sample it. Alternatively, the input can be seen as a sequence of inaccessible probabilities $p_1, \ldots, p_n \in [0,1]$, and in every step, the algorithm can choose an index $1 \le i \le n$ and sample a single bit that distributes like $\text{Ber}(p_i)$.

We use the following reduction for simulating the coin model using the prefix sampling model. The input of the distinguishing task corresponds to the sequence of marginals of a given random element x (that is, $p_1, \ldots, p_n$). Every prefix conditional sample translates into $O(1)$ steps in the coin model in expectation (see Lemma 48).

We can use the following result regarding a closely related task to deduce an $\Omega(1/r^2\delta^2) = \Omega(n^2/\varepsilon^2)$ lower bound. However, we explicitly prove hardness in Appendix A for completeness.

▶ **Lemma 29** (Rephrase of Theorem 3 in [14]). *Assume that for every $1 \le i \le n$, $p_i \ge (1+\delta)/2$ or $p_i \le (1-\delta)/2$. In this setting, estimating the fraction of positively-biased ($p_i \ge (1+\delta)/2$) within a $\pm r$-additive error requires $\Omega(1/r^2\delta^2)$ samples.*

4.4 Parameter table

Different sections use different parameter constraints. The upper bound uses n and ε. The lower bound for the sequence-distinguishing task uses n, δ and r, whereas the lower bound for estimation uses n and ε and explicitly defines δ, r.

Section	n	ε	δ	r
5 Upper bound	$n \ge 1$	$\varepsilon > 0$	N/A	N/A
6 Lower bound	$n \ge 4.5 \cdot 10^{16}$	$0 < \varepsilon \le e^{-11}$	$\sqrt{2}\varepsilon/\sqrt{n}$	$8/\sqrt{n}$
A Sequence distinguishing	by context	N/A	$0 < \delta < 1/3$	$0 \le r < 1/12$

5 The simulation algorithm

In this section we describe our simulation algorithm in the prefix conditional sampling model for an input distribution μ over $\Omega = \{0,1\}^n$.

Before we introduce our algorithm, we describe the common method (which is compatible with [13]). For every $1 \le i \le n$ and every $w \in \{0,1\}^{i-1}$, we use $O(n^2 \log(n/\varepsilon)/\varepsilon^2)$ samples to estimate the marginal probability $\mu(x_i = 1|w)$ within a $\left(1 \pm \frac{\varepsilon}{\sqrt{n}}\right)$-multiplicative factor with probability at least $1 - O(\varepsilon/n)$ (unless it is too close to zero or one, and in this case we estimate it as zero or one with probability $1 - O(\varepsilon/n)$).

Along the path from the root to a given leaf there are n edges, but errors of different directions (overshooting and undershooting) mostly cancel each other. Hence, with high probability, the accumulated error (when multiplying all n errors) in the range is $\left(1 \pm \frac{\varepsilon}{\sqrt{n}}\right)^{\pm\Theta(\sqrt{n})} = 1 \pm O(\varepsilon)$. By the union bound, the probability to obtain good estimations in all edges is at least $1 - O(\varepsilon)$. Overall, all elements have their mass estimated within

$(1 \pm O(\varepsilon))$-multiplicative error, except for a set of elements whose expected mass is $O(\varepsilon)$ as well. This implies that the expected total-variation distance of μ and the simulating distribution is $O(\varepsilon)$. The cost of querying the simulating distribution is $O(n^3 \log(n/\varepsilon)/\varepsilon^2)$, since we have to query n edges at the cost of $O(n^2 \log(n/\varepsilon)/\varepsilon^2)$ prefix samples per edge.

Our simulation algorithm uses the chain rule for D_{KL} (Lemma 62). Similarly to [5, 1], repeatedly applying the chain rule results in a bitwise identity.

▶ **Lemma 30** (Repeated chain rule for D_{KL})**.** *Let μ and τ be two distributions over $\{0,1\}^n$. In this setting, $D_{\mathrm{KL}}(\mu \parallel \tau) = \sum_{i=1}^{n} \mathrm{E}_{w \sim \mu|_{1,\ldots,i-1}} \left[D_{\mathrm{KL}}\left(\mu|_i^{\{w\}\times\{0,1\}^{n-i}} \,\middle\|\, \tau|_i^{\{w\}\times\{0,1\}^{n-i}} \right) \right]$.*

While the total-variation distance of Bernoulli distributions is linear, Lemma 61 states that the KL-divergence has a χ^2-like behavior, which is sublinear in many cases.

The following lemma is the core of our algorithm. It states that the expected KL-divergence of an estimated probability mass is linear in the number of samples, rather than its square-root as in total-variation distance.

▶ **Lemma 31.** *Let $p \in [0,1]$ and $m \geq 1$. In this setting, $\mathrm{E}_{t \sim \mathrm{Bin}(m,p)} \left[D_{\mathrm{KL}}(\frac{t}{m}, p) \right] \leq \frac{1}{m}$.*

Proof. For $p = 0$ and $p = 1$, $\Pr[t = mp] = 1$, and hence $\mathrm{E}[D_{\mathrm{KL}}(\frac{t}{m}, p)] = 0 \leq \frac{1}{m}$. For the rest of the proof we assume that $0 < p < 1$. In particular, we can safely divide by p and by $1 - p$.

Let $\hat{p} = \frac{t}{m}$ and $\delta = \frac{\hat{p}}{p} - 1$ (so that $\hat{p} = (1+\delta)p$). By Lemma 61, $D_{\mathrm{KL}}(\hat{p}, p) \leq \frac{(\hat{p}-p)^2}{p(1-p)} = \frac{\delta^2 p}{1-p}$. Therefore,

$$\begin{aligned}
\mathop{\mathrm{E}}_{t}[D_{\mathrm{KL}}(t/m, p)] \leq \frac{p}{1-p} \cdot \mathop{\mathrm{E}}_{t}\left[\left(\frac{t/m}{p} - 1 \right)^2 \right] &= \frac{p}{(mp)^2(1-p)} \cdot \mathop{\mathrm{E}}_{t}\left[(t - \mathrm{E}[t])^2 \right] \\
&= \frac{1}{m^2 p(1-p)} \mathrm{Var}[t] \\
&= \frac{1}{m^2 p(1-p)} p(1-p)m = \frac{1}{m}
\end{aligned}$$

◀

We show two implementations of $(D_{\mathrm{KL}}, \delta)$-simulation. The first, shown in Algorithms 2, 3 and 4 as a conceptual introduction, is extremely inefficient, but it is quite easy to show its correctness as a learning algorithm. The second, in Algorithms 5, 7 and 8, is efficient, and we prove its correctness by coupling its behavior with the behavior of the first implementation.

Both implementations use a common procedure Estimate-edge (Algorithm 1) that estimates the probability mass of a single edge by drawing $m = \lceil n/\delta \rceil$ prefix conditional samples and averaging the first free bit.

The first implementation has the learn-then-read concept. We exhaustively build the distribution tree of a distribution $\tilde{\mu}$ simulating the input distribution μ, and then query $\tilde{\mu}$ at individual elements whenever needed.

The initialization procedure, Learn-distribution, estimates all right edges (of the form $w \to w1$) independently.

The query procedure Query-learned-distribution and the sample procedure Sample-learned-distribution just access the explicitly-written learned distribution $\tilde{\mu}$.

▶ **Lemma 32.** *The triplet (Learn-distribution, Query-learned-distribution, Sample-learned-distribution) is a $(D_{\mathrm{KL}}, \delta)$-simulation of its input distribution.*

The second implementation has the lazy-construction concept. Since the estimations of Algorithm 2 are independent, it suffices to evaluate an edge when it (or its sibling edge) lies on a path of a query for the first time.

The initialization procedure Initialize-simulation initializes an empty memorization object.

Algorithm 1: Procedure Estimate-edge$(n, \mu, \delta; w, b)$.

Input: μ is the distribution over $\Omega = \{0,1\}^n$.
Input: δ is the accuracy parameter.
Input: $w \to wb$ is the edge to estimate.
Output: An estimation of $\mu(b|w)$.

1. Let $m \leftarrow \lceil n/\delta \rceil$.
2. Draw $x_1, \ldots, x_m$ independently from μ, conditioned on $x_{1,\ldots,|w|} = w$.
3. Let k be the number of samples $x_1, \ldots, x_m$ in which the $|w|+1$st bit is b.
4. Return k/m.

Algorithm 2: Procedure Learn-distribution(n, μ, δ).

Input: μ is the distribution over $\Omega = \{0,1\}^n$ to simulate.
Input: δ is the accuracy parameter.
Output: The simulating distribution $\tilde{\mu}$.
Accuracy: $\mathrm{E}\left[D_{\mathrm{KL}}(\tilde{\mu} \parallel \mu)\right] \leq \delta$.

1. Initialize the marginal function $f : \{0,1\}^{<n} \to [0,1]$, with all entries initially undefined.
2. For $w \in \{0,1\}^{<n}$ in an arbitrary order:
   a. Run and write $f(w) \leftarrow$ Estimate-edge$(n, \mu, \delta; w, 1)$.
3. Let $\tilde{\mu}$ be the distribution represented by the marginal function f. (Def. 24, Obs. 25)
4. Return $\tilde{\mu}$.

Algorithm 3: Procedure Query-learned-distribution$(\tilde{\mu}, x)$.

Input: $\tilde{\mu}$ is the learned distribution returned by Learn-distribution.
Output: The probability mass of x according to $\tilde{\mu}$.

1. Return $\tilde{\mu}(x)$.

Algorithm 4: Procedure Sample-learned-distribution$(\tilde{\mu})$.

Input: $\tilde{\mu}$ is the learned distribution returned by Learn-distribution.
Output: A sample x drawn from the learned distribution, and its probability mass $\tilde{\mu}(x)$.

1. Draw x from $\tilde{\mu}$.
2. Return $(x, \tilde{\mu}(x))$.

Algorithm 5: Procedure Initialize-simulation(n, μ, δ).

Input: μ is the distribution over $\Omega = \{0,1\}^n$ to simulate.
Input: δ is the accuracy parameter.

1. Return $(n, \mu, \delta, hist)$, where $hist$ is an empty list.

The sample and query procedures internally call the Access-simulation-edge procedure for an edge-by-edge execution.

The Access-simulation-edge procedure allows a memorized estimation of an edge. If the requested edge has never been estimated, then the procedure estimates it using Estimate-edge and then memorizes the result for both that edge and its sibling edge.

Algorithm 6: Procedure Access-simulation-edge(sim, w, b).

Input: sim: the simulation object obtained by Initialize-simulation.
Input: $w \to wb$ is edge to access.
Side effects: The $hist$ component of sim may change.

1. Let $n, \mu, \delta, hist$ be the components of sim as a 4-tuple.
2. If $hist$ contains $((w, b), ans)$:
   a. Let $f \leftarrow ans$.
3. Else:
   a. Let $f \leftarrow$ Estimate-edge$(n, \mu, \delta; w, b)$.
   b. Add $((w, b), f)$, $((w, 1-b), 1-f)$ to $hist$.
4. Return f.

The query and sample procedures execute an edge-by-edge logic. The edges are estimated through Access-simulation-edge.

Algorithm 7: Procedure Query-simulation(sim, x).

Input: sim: the simulation object obtained by Initialize-simulation.
Output: The probability mass of x according to the secret distribution.
Side effects: The $hist$ component of sim may change.

1. Let $n, \mu, \delta, hist$ be the components of sim as a 4-tuple.
2. Set $p_x \leftarrow 1$.
3. For i from 1 to n:
   a. Let $f_{x,i} \leftarrow$ Access-simulation-edge$(sim, x_{1,\ldots,i-1}, x_i)$.
   b. Set $p_x \leftarrow p_x \cdot f_{x,i}$.
4. Return p_x.

Algorithm 8: Procedure Sample-simulation(sim).

Input: sim: the simulation object obtained by Initialize-simulation.
Output: A sample x drawn from the secret distribution, and its probability mass according to that distribution.
Side effects: The *hist* component of sim may change.

1. Let $n, \mu, \delta, hist$ be the components of sim as a 4-tuple.
2. Set $p \leftarrow 1$.
3. Set $x \leftarrow$ the empty word.
4. For i from 1 to n:
   a. Let $f_i \leftarrow$ Access-simulation-edge$(sim, x, 1)$.
   b. Draw $b_i \in \{0, 1\}$ according to $\mathrm{Ber}(f_i)$.
   c. Set $x \leftarrow xb_i$. Append b to the end of x.
   d. If $b_i = 1$:
      i. Set $p \leftarrow p \cdot f_i$.
   e. Else:
      i. Set $p \leftarrow p \cdot (1 - f_i)$.
5. Return (x, p).

▶ **Theorem 33.** *The triplet (Initialize-simulation, Query-simulation, Sample-simulation) has the exact behavior (with respect to the user) as the triplet (Learn-distribution, Query-learned-distribution, Sample-learned-distribution). In particular, it is a $(D_{\mathrm{KL}}, \delta)$-simulation of its input distribution.*

Proof. The interaction of Query-simulation and Sample-simulation with the simulation object is only done through Access-simulation-edge calls. Since in Algorithm 2 the edges are independent, the specific order of assigning their weight is irrelevant. Each call to Access-simulation-edge assigns a weight to two sibling edges, independently of other edges. Therefore, the sequential access to the constructed distribution tree through Access-simulation-edge is identically distributed as evaluating all edges in advance as in Learn-distribution followed by reading the preprocessed data. ◀

We improve on [13] by applying Theorem 33 to Proposition 1.

▶ **Corollary 34.** *Let μ and τ be two distributions over $\Omega = \{0, 1\}^n$. Estimating the total-variation distance of μ and τ up to a $\pm\varepsilon$-additive error with probability at least $2/3$ can be done using $O(n^2/\varepsilon^4)$ prefix conditional samples.*

Proof. We use Theorem 33 to simulate μ and τ using random distributions $\tilde{\mu}$ and $\tilde{\tau}$ such that $\mathrm{E}[D_{\mathrm{KL}}(\tilde{\mu} \parallel \mu)] < \frac{1}{36}\varepsilon^2$ and $\mathrm{E}[D_{\mathrm{KL}}(\tilde{\tau} \parallel \tau)] < \frac{1}{36}\varepsilon^2$. After we initialize the simulation at no sample cost, we run a $O(1/\varepsilon^2)$-test of Proposition 1 to estimate the total-variation distance between $\tilde{\mu}$ and $\tilde{\tau}$ within $\pm\frac{1}{3}\varepsilon$-additive error. Such a test appears in [13] (inlined in a bigger subroutine) and in [2] (explicitly).

The complexity of the test is $O(1/\varepsilon^2)$ samples of the simulating distributions, each costing $O(n^2/\varepsilon^2)$ prefix conditional samples of the input distributions.

By Markov's inequality, with probability at least $(7/8)^2$, both $D_{\mathrm{KL}}(\tilde{\mu} \parallel \mu)$ and $D_{\mathrm{KL}}(\tilde{\tau} \parallel \tau)$ are at most $\frac{2}{9}\varepsilon^2$, and in this case, $|d_{\mathrm{TV}}(\mu, \tau) - d_{\mathrm{TV}}(\tilde{\mu}, \tilde{\tau})| \le d_{\mathrm{TV}}(\tilde{\mu}, \mu) + d_{\mathrm{TV}}(\tilde{\tau}, \tau) \le 2 \cdot \frac{1}{3}\varepsilon$.

Hence, estimating $d_{\mathrm{TV}}(\tilde{\mu},\tilde{\tau})$ within $\pm\frac{1}{3}\varepsilon$-additive error suffices to deduce a $\pm\varepsilon$-additive error estimation of $d_{\mathrm{TV}}(\mu,\tau)$. If we use $O(1)$ trials to amplify the success probability of estimating $d_{\mathrm{TV}}(\tilde{\mu},\tilde{\tau})$ to 8/9, then the success probability is at least $(7/8)^2 \cdot (8/9) > 2/3$. ◀

In particular, estimating the total-variation distance between two distributions over binary strings using subcube conditional sampling requires $O(n^2/\varepsilon^4)$ samples, a polynomial gain over the previously best known distance estimation algorithm for subcube conditions (by [13]), whose complexity is $\tilde{O}(n^3/\varepsilon^5)$ subcube conditional samples.

6 Lower bound

We show that estimation of the probability mass of individual elements using prefix conditional samples cannot be more efficient than simulating the input distribution as a whole (through Algorithms 5, 7, 8 and Theorem 33). Recall that we can fully define a distribution by assigning weights to the edges of the breakdown tree implied by the prefix condition sets.

▶ **Definition 35** (The counter-estimation distributions over inputs). *Let $\Omega = \{0,1\}^n$ be the domain and $\delta = \sqrt{2}\varepsilon/\sqrt{n}$, $r = 8/\sqrt{n}$ be internal parameters. For every $w \in \{0,1\}^{<n}$, we draw $s_w \in \{+1,-1\}$ uniformly and independently. We define the input distribution μ, and an additional distribution μ', through their marginals. For every $w \in \{0,1\}^{<n}$, we define $\mu(x_{|w|+1} = 1|w) = \frac{1+s_w\delta}{2}$ and $\mu'(x_{|w|+1} = 1|w) = \frac{1-s_w r}{2}$. In D_{yes}, we draw x uniformly and then return the pair (μ,x). In D_{no}, we draw $x \sim \mu'$ and return the pair (μ,x).*

We insist that μ is the input distribution in both cases. We only use μ' for drawing x in D_{no}.

▶ **Lemma 36.** *For every $x \in \{0,1\}^n$, let I_x be the set of prefixes $w \in \{0,1\}^{<n}$ that are not prefixes of x. In both D_{yes} and in D_{no}, when conditioned on an explicitly given x, the random variables $\{s_w\}_{w\in I_x}$ are drawn uniformly and independently from $\{+1,-1\}$.*

6.1 Concentration gap

For $n \geq 2$, $\varepsilon > 0$, $\delta = \sqrt{2}\varepsilon/\sqrt{n}$ and $r = 8/\sqrt{n}$, we define the following notations.

▶ **Definition 37** (A few technical definitions).

$$k_H = \frac{1}{2}n - \sqrt{3}\sqrt{n} \qquad k_L = \frac{1}{2}n - \sqrt{6}\sqrt{n}$$
$$p_H = 2^{-n}(1+\delta)^{k_H}(1-\delta)^{n-k_H} \qquad p_L = 2^{-n}(1+\delta)^{k_L}(1-\delta)^{n-k_L}$$
$$H_\mu = \{x : \mu(x) \geq p_H\} \qquad L_\mu = \{x : \mu(x) \leq p_L\}$$

The following lemmas (38-44) are technical.

▶ **Lemma 38.** *For $n \geq 1$ and $\varepsilon < 1/150$, $(1-\varepsilon)p_H > (1+\varepsilon)p_L$.*

▶ **Lemma 39.** *With probability 1 over the choice of μ, μ is ε-close to the uniform distribution over $\{0,1\}^n$.*

▶ **Lemma 40.** *For $n \geq 2.2 \cdot 10^8$, with probability 1 over the choice of μ, $\mu(H_\mu) \geq 1 - e^{-8}$.*

▶ **Lemma 41.** *For $n \geq 4.5 \cdot 10^{16}$ and $\varepsilon < \frac{1}{2000}$, with probability 1 over the choice of μ, $\mu(L_\mu) > e^{-14.57}$.*

▶ **Lemma 42.** *For $n \geq 3 \cdot 10^{10}$ and $\varepsilon > 0$, with probability 1 over the choice of μ and μ', for every $x \in L_\mu$, $\mu'(x) \geq e^{7.19}\mu(x)$.*

▶ **Lemma 43.** *For $n \geq 4.5 \cdot 10^{16}$ and $\varepsilon \leq e^{-11}$, $\Pr_{(\mu,x)\sim D_{\mathrm{yes}}}[\mu(x) \geq p_H] \geq 1 - e^{-7.95}$.*

▶ **Lemma 44.** *For $n \geq 4.5 \cdot 10^{16}$ and $\varepsilon \leq e^{-11}$, $\Pr_{(\mu,x)\sim D_{\mathrm{no}}}[\mu(x) \leq p_L] \geq e^{-7.38}$.*

6.2 The lower bound theorem

▶ **Lemma 45.** *For $n \geq 4.5 \cdot 10^{16}$ and $\varepsilon \leq e^{-11}$, let $\mathcal{A}$ be an algorithm that uses prefix conditional samples and results in a real number such that for every distribution μ over $\Omega = \{0,1\}^n$ there exists a set $G_\mu \subseteq \Omega$ of mass $\Pr_\mu[G_\mu] > 1 - e^{-20}$ for which $\Pr[\mathcal{A}(\mu, x) \in (1 \pm \varepsilon)\mu(x)] > 1 - e^{-20}$ for every $x \in G_\mu$. In this setting, the total variation distance of the behaviors of the algorithm when given an input drawn from D_{yes} and from D_{no} is greater than $e^{-8.25}$.*

Note that, for estimation algorithms, having $\mu(G_\mu) = 1$ is infeasible, and general upper bound algorithms usually guarantee that $\mu(G_\mu) = 1 - O(\varepsilon)$. Here we show a lower bound for a much weaker demand, that allows for excluding a fixed weight of the elements.

Proof. Consider the distributions D_{yes} and D_{no} and the following algorithm: we use $\mathcal{A}$ to estimate $(1 \pm \varepsilon)\mu(x)$. If the result estimation is greater than $(1 - \varepsilon)p_H$ then we accept, and otherwise we reject. By Lemma 38, the ranges $(1 \pm \varepsilon)p_H$ and $(1 \pm \varepsilon)p_L$ are disjoint, and therefore, a $1 \pm \varepsilon$-multiplicative error suffices to distinguish between p_H and p_L.

Let B_{wrong} be the bad event "the estimation of $\mu(x)$ is outside the range $(1 \pm \varepsilon)\mu(x)$". The probability to reject an input (μ, x) drawn from D_{yes} is bounded by:

$$\begin{aligned}
\Pr[\text{REJECT}|D_{\text{yes}}] &\leq \Pr_{D_{\text{yes}}}[x \notin H_\mu \cap G_\mu] + \Pr[B_{\text{wrong}}|x \in G_\mu] \\
&\leq d_{\text{TV}}(\mathcal{U}, \mu) + (1 - \mu(H_\mu)) + (1 - \mu(G_\mu)) + e^{-20} \\
\text{[L 39, L 40]} &\leq \varepsilon + e^{-8} + e^{-20} + e^{-20}
\end{aligned}$$

By Lemma 41 and Lemma 42,

$$\mu'(L_\mu \cap G_\mu) \geq \min_{x \in L_\mu} \frac{\mu'(x)}{\mu(x)} \cdot (\mu(L_\mu) - (1 - \mu(G_\mu))) \geq e^{7.19} \cdot (e^{-14.57} - e^{-20}) \geq e^{-7.385}$$

The probability to reject an input (μ, x) drawn from D_{no} is at least:

$$\Pr[\text{REJECT}|D_{\text{no}}] \geq \Pr_{D_{\text{no}}}[x \in G_\mu \cap L_\mu] \cdot \Pr[\neg B_{\text{wrong}}|x \in G_\mu] \geq e^{-7.385} \cdot (1 - e^{-20}) \geq e^{-7.39}$$

Therefore, the distance in behaviors of the algorithms when running on D_{yes} and D_{no} is at least $e^{-7.39} - (\varepsilon + e^{-8} + e^{-20} + e^{-20}) > e^{-8.25}$ (for $\varepsilon \leq e^{-11}$). ◀

▶ **Lemma 46.** *For $n \geq 4.5 \cdot 10^{16}$ and $\varepsilon < e^{-11}$, let $\mathcal{A}$ be an algorithm that uses marginal prefix conditional samples and results in a real number such that for every distribution μ over $\Omega = \{0,1\}^n$ there exists a set $G_\mu \subseteq \Omega$ of mass $\Pr_\mu[G_\mu] > 1 - e^{-20}$ for which $\Pr[\mathcal{A}(\mu, x) \in (1 \pm \varepsilon)\mu(x)] > 1 - e^{-20}$ for every $x \in G_\mu$. In this setting, $\mathcal{A}$ must make at least $\frac{n^2}{2.8 \cdot 10^{25} \varepsilon^2}$ marginal prefix conditional samples that contain x in expectation.*

Proof. Observe that, for every $w \in \{0,1\}^{<n}$, the marginal prefix conditional sample $\mu|_{|w|+1}^{w \times \{0,1\}^{n-|w|}}$ is identical to drawing a bit from $\text{Ber}\left(\frac{1+s_w\delta}{2}\right)$. By Lemma 36, the random variables $\{s_w\}_{w \in I_x}$, for $I_x = \{0,1\}^{<n} \setminus \{1 \leq i \leq n : x_{1,\ldots,i-1}\}$, are independent, and distribute the same in D_{yes} and in D_{no}. Hence, sampling them is redundant.

In other words, in every non-redundant step, the algorithm chooses some $1 \leq i \leq n$ and samples a bit whose probability to be 1 is $\frac{1+s_{x_{1,\ldots,i-1}}\delta}{2}$. For every $1 \leq i \leq n$, let $p_i = \frac{1+s_{x_{1,\ldots,i-1}}\delta}{2}$.

In D_{yes}, every s_w is uniformly drawn from $\{+1, -1\}$, and hence, the p_is are uniformly drawn from $\frac{1 \pm \delta}{2}$. This matches the $D_n(\delta, 0)$ distribution of the sequence-distinguishing task. In D_{no}, when conditioned on drawing x from μ', every s_w (for w on the path of x) is $+1$ with probability $\frac{1-r}{2}$ and otherwise -1. This matches the $D_n(\delta, r)$ distribution of the sequence-distinguishing task.

Note that this analysis is oblivious of the dependence between the s_ws and x. Observe that if we denote $s_{x_{1,\dots,i-1}}$ by $\hat{s}_{i-1}$, then the sequence $(\hat{s}_0,\dots,\hat{s}_{n-1})$ is independent of x and iid.

Let q be the expected number of non-redundant steps. By Markov's inequality, with probability at least $1-e^{-9}$ it is bounded by $e^9 q$.

If $e^9 q \le \frac{(e^{-9})^4}{4\cdot 10^5\delta^2}n = \frac{n^2}{e^{36}\cdot 4\cdot 10^5\cdot 2\varepsilon^2}$, then by Lemma 56, the total variation distance between the runs is less than e^{-9}.

Overall, if $q \le \frac{n^2}{2.8\cdot 10^{25}\varepsilon^2} \le \frac{n^2}{e^9\cdot e^{36}\cdot 4\cdot 10^5\cdot 2\varepsilon^2}$, then the total-variation distance between the run is bounded by $e^{-9}+e^{-9} < e^{-8.25}$. By Lemma 45, a "good" estimation algorithm behaves more differently ($d_{\mathrm{TV}} > e^{-8.25}$) on D_{yes} on D_{no}, and hence, every such an algorithm must make at least $\frac{n^2}{2.8\cdot 10^{25}\varepsilon^2}$ prefix conditional queries. ◀

We describe a setting for the following lemma.

▶ **Definition 47** (Setting for Lemma 48). *Let $x \in \{0,1\}^n$, $1 \le i \le n$ and $w \in \{0,1\}^{i-1}$ be a prefix. Let $X_i,\dots,X_n$ be the bits of a prefix conditional sample $\mu|^{w\times\{0,1\}^{n-i}}$. Let $W_{i-1},\dots,W_{n-1}$ be the intermediate prefixes ($W_{i-1}=w$, $W_{j-1}=W_{j-2}X_{j-1}$ for $i+1\le j\le n$). Also, let $N_{\mathrm{effective}} = |\{i\le j\le n : W_{j-1} \text{ is a prefix of } x\}|$.*

As observed before, the sampling from $\mu|^{w\times\{0,1\}^{n-i}}$ can be decomposed into $n-i+1$ marginal samplings such that for $i\le j\le n$, after W_{i-1} is determined, X_j distributes as $\mu|_j^{W_{j-1}\times\{0,1\}^{n-j}}$. Therefore, $N_{\mathrm{effective}}$ describes the number of effective marginal samples.

▶ **Lemma 48.** *In the setting of Definition 47, if $n \ge 20$, then $\mathrm{E}[N_{\mathrm{effective}}] \le 3$.*

▶ **Theorem 49.** *For $n \ge 4.5\cdot 10^{16}$ and $\varepsilon \le e^{-11}$, let $\mathcal{A}$ be an algorithm that uses prefix conditional samples and results in a real number such that for every distribution μ over $\Omega = \{0,1\}^n$ there exists a set $G_\mu \subseteq \Omega$ of mass $\Pr_\mu[G_\mu] > 1-e^{-20}$ for which $\Pr[\mathcal{A}(\mu,x)\in(1\pm\varepsilon)\mu(x)] > 1-e^{-20}$ for every $x\in G_\mu$. In this setting, $\mathcal{A}$ must use $\Omega(n^2/\varepsilon^2)$ prefix conditional samples.*

Proof. Recall that we can see every prefix condition sampling with k free bits as a (random) sequence of k marginal prefix condition sampling. In the ith step, we condition on the first already-fixed $n-k$ bits and on the additional $i-1$ already-sampled bits to obtain the $n-k+i$th bit. By Lemma 48, for every individual prefix sample, the expected number of marginal samples whose condition contains x is bounded by 3. By linearity of expectation, the expected number of marginal samples whose prefix condition contains x is bounded by $3q$.

By Lemma 46, since the expected number of "non-redundant" samples is $O(q)$, the algorithm must make $q=\Omega(n^2/\varepsilon^2)$ prefix conditional samples. ◀

References

1. Tomer Adar, Eldar Fischer, and Amit Levi. Improved bounds for high-dimensional equivalence and product testing using subcube queries. In *Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX/RANDOM)*, 2024.
2. Tomer Adar, Eldar Fischer, and Amit Levi. Optimal mass estimation in the conditional sampling model. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2026.
3. Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 442–451. IEEE, 2001. doi:10.1109/SFCS.2001.959920.

4 Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE, 2000. doi:10.1109/SFCS.2000.892113.

5 Rishiraj Bhattacharyya and Sourav Chakraborty. Property testing of joint distributions using conditional samples. *ACM Transactions on Computation Theory (TOCT)*, 10(4):1–20, 2018. doi:10.1145/3241377.

6 Antonio Blanca, Zongchen Chen, Daniel Štefankovič, and Eric Vigoda. Complexity of high-dimensional identity testing with coordinate conditional sampling. *ACM Transactions on Algorithms*, 21(1):1–58, 2024. doi:10.1145/3686799.

7 Clément L Canonne, Xi Chen, Gautam Kamath, Amit Levi, and Erik Waingarten. Random restrictions of high dimensional distributions and uniformity testing with subcube conditioning. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 321–336. SIAM, 2021. doi:10.1137/1.9781611976465.21.

8 Clément L Canonne, Ilias Diakonikolas, Daniel M Kane, and Alistair Stewart. Testing bayesian networks. In *Conference on Learning Theory*, pages 370–448. PMLR, 2017. URL: http://proceedings.mlr.press/v65/canonne17a.html.

9 Clément L Canonne, Dana Ron, and Rocco A Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015. doi:10.1137/130945508.

10 Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM Journal on Computing*, 45(4):1261–1296, 2016. doi:10.1137/140964199.

11 Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1193–1203. SIAM, 2014. doi:10.1137/1.9781611973402.88.

12 Xi Chen and Cassandra Marcussen. Uniformity testing over hypergrids with subcube conditioning. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4338–4370. SIAM, 2024. doi:10.1137/1.9781611977912.152.

13 Gunjan Kumar, Kuldeep S Meel, and Yash Pote. Distance estimation for high-dimensional discrete distributions. In *The 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

14 Jasper CH Lee and Paul Valiant. Uncertainty about uncertainty: Optimal adaptive algorithms for estimating mixtures of unknown coins. *arXiv e-prints*, pages arXiv–1904, 2019.

15 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008. doi:10.1109/TIT.2008.928987.

16 Ronitt Rubinfeld and Madhu Sudan. Self-testing polynomial functions efficiently and over rational domains. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 23–32, 1992. URL: http://dl.acm.org/citation.cfm?id=139404.139410.

17 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

18 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

A Lower bound for the sequence-distinguishing task

In this appendix we show a lower bound for the sequence-distinguishing task. The goal is distinguishing between the following random sequences, regarding the parameters $\delta > 0$ and $r \geq 0$:

- $D_n(\delta, 0)$: $p_i = (1+\delta)/2$ with probability $1/2$, and otherwise $p_i = (1-\delta)/2$, independently for every $1 \le i \le n$.
- $D_n(\delta, r)$: $p_i = (1+\delta)/2$ with probability $(1-r)/2$, and otherwise $p_i = (1-\delta)/2$, independently for every $1 \le i \le n$.

In each step, the algorithm can choose an index $1 \le i \le n$ and obtain a bit drawn from $\mathrm{Ber}(p_i)$.

We show that, if $n \le 64/r^2$, then we must make $\Omega(1/r^2\delta^2)$ queries, even if we allow adaptive behavior.

A.1 Technical lemmas

We first state a few technical lemmas.

▶ **Lemma 50.** *Let $0 \le t \le 1/4$. For two distributions μ and ν over Ω for which $\mu(x) \in (1 \pm t)\nu(x)$ for every $x \in \Omega$, $D_{\mathrm{KL}}(\mu \parallel \nu) \le t^2/\ln 2$.*

▶ **Lemma 51.** *Let μ and ν be two distributions over a domain Ω. It holds that:*

$$\sum_{x \in \Omega} \frac{(\mu(x) - \nu(x))^2}{\mu(x) + \nu(x)} \le (D_{\mathrm{KL}}(\mu \parallel \nu) + D_{\mathrm{KL}}(\nu \parallel \mu)) \cdot \ln 2$$

▶ **Lemma 52.** *Let μ and ν be two distributions over a domain Ω. If $|r| < \frac{1}{2}$, then:*

$$D_{\mathrm{KL}}\left(\frac{1}{2}\mu + \frac{1}{2}\nu \,\middle\|\, \frac{1+r}{2}\mu + \frac{1-r}{2}\nu\right) \le \frac{1}{2}r^2(D_{\mathrm{KL}}(\mu \parallel \nu) + D_{\mathrm{KL}}(\nu \parallel \mu))$$

A.2 Lower bound for non-adaptive algorithms

Before we introduce the lower bound against adaptive algorithms, we state a lower bound for non-adaptive algorithms. A non-adaptive algorithm draws m_i samples from the ith index for every $1 \le i \le n$. The sample complexity is $q = \sum_{i=1}^{n} m_i$.

▶ **Lemma 53.** *For $\delta < 1/3$, consider an algorithm that for every $i \ge 1$ draws m_i samples from the ith index, and let $q = \sum_{i=1}^{n} m_i$. The KL-divergence of runs over inputs drawn from $D_n(\delta, 0)$ and from runs over inputs drawn from $D_n(\delta, r)$ is bounded by $5r^2\delta^2 q$.*

A.3 Lower bound for adaptive algorithms

Adaptive algorithms are allowed to choose their next step based on past samples. Therefore, their analysis is more difficult. By Yao's principle [18], a probabilistic algorithm is equivalent to a distribution over deterministic algorithms. Moreover, the success probability of a probabilistic algorithm is the expected success probability over the choice of a deterministic algorithm.

Every deterministic adaptive algorithm can be described as a decision tree, where edges correspond to samples (0 and 1) and every non-leaf node is associated with the index $1 \le i \le n$ to sample in the following step. A run of a deterministic algorithm is its execution path, which corresponds to the leaf the algorithm reaches in its last step.

To show a lower bound of q samples, we have to show that for every decision tree of height q, the total-variation distance between the distribution of leaves reached when the input is drawn from $D_n(\delta, 0)$ and their distribution when the input is drawn from $D_n(\delta, r)$ is bounded by some $d < 1/3$. This implies that the success probability, which cannot exceed the maximal success probability of a deterministic algorithm, is at most $1 - d$. By Pinsker's inequality, it suffices to bound the KL-divergence of the leaf distributions by $2d^2$.

We repeatedly apply the chain-rule for KL-divergence (Lemma 62) to obtain that the KL-divergence of the leaf distributions is bounded by the expected KL-divergence of every step individually.

To analyze an adaptive algorithm, whose structure can be arbitrarily detailed, we simulate it using a stronger model whose runs have shorter descriptions. Let $m = \left\lfloor \frac{d^2}{400\delta^2} \right\rfloor$ and $q \leq \frac{d^2}{1000}mn$. At the initialization phase, we run an m-height rectangle algorithm, which is a non-adaptive algorithm that makes m queries in each index $1 \leq i \leq n$, obtaining a pre-processing sampling matrix $M \in \{0,1\}^{m \times n}$ at the cost of mn samples. For every individual index $1 \leq i \leq n$, the first m queries are answered based on the pre-processing sampling matrix. Once the algorithm makes the $m+1$st sample on the ith index, the simulator reveals the exact value of p_i, and exactly simulates further queries to the ith index using the full-knowledge about p_i.

▶ **Observation 54.** *Let $d > 0$. An adaptive algorithm that draws $q \leq \frac{d^2}{1000}mn$ samples has at most $\frac{d^2}{1000}n$ indexes sampled more than m times.*

▶ **Observation 55.** *Every run of an adaptive algorithm with $q \leq \frac{d^2}{1000}mn$ samples can be described as a tuple of the pre-processing sampling matrix M, the set I of at most $\frac{d^2}{1000}n$ fully-revealed indexes and a map $f : I \to \{+,-\}$ corresponding to the exact value of $p_i \in \frac{1\pm\delta}{2}$.*

▶ **Lemma 56.** *Let $\delta < 1/3$, $r < 1/12$, $n \leq 64/r^2$ and $d > 0$. For every adaptive algorithm with $q \leq \frac{d^4}{4\cdot 10^5 \delta^2}n$ samples, the distance between runs over inputs drawn from $D_n(\delta, 0)$ and runs over inputs drawn from $D_n(\delta, r)$ is bounded by d.*

B Common distributions and fact sheet

B.1 Common distributions

▶ **Definition 57** (Bernoulli distribution). *Bernoulli distribution* $\mathrm{Ber}(p)$ *is the distribution over* $\{0,1\}$ *for which* $\Pr[1] = p$ *and* $\Pr[0] = 1-p$.

▶ **Definition 58** (Binomial distribution). *The binomial distribution* $\mathrm{Bin}(n,p)$ *is the sum of* n *independent samples from the Bernoulli distribution* $\mathrm{Ber}(p)$.

B.2 Fact sheet

In the following we provide some well-known bounds and identities.

▶ **Lemma 59** (Pinsker's inequality). $D_{\mathrm{KL}}(\mu \parallel \tau) \geq 2(d_{\mathrm{TV}}(\mu,\tau))^2$.

▶ **Lemma 60.** *For $x \geq -2/3$, $-\ln(1+x) \leq -x + x^2$.*

▶ **Lemma 61** (see [8]). *For $p, q \in [0,1]$, $D_{\mathrm{KL}}(p,q) \leq \frac{(p-q)^2}{q(1-q)}$.*

▶ **Lemma 62** (Chain rule for D_{KL}). *Let μ and τ be two distributions over $A \times B$. In this setting, $D_{\mathrm{KL}}(\mu \parallel \tau) = D_{\mathrm{KL}}(\mu|_A \parallel \tau|_A) + \mathrm{E}_{a\sim\mu|_A}\left[D_{\mathrm{KL}}\left(\mu|^{\{a\}\times B} \,\middle\|\, \tau|^{\{a\}\times B}\right)\right]$.*