

Interactive Proofs For Distribution Testing With Conditional Oracles

Ari Biswas   



University Of Warwick, United Kingdom

Mark Bun¹  

Boston University

Clément Canonne²  

University of Sydney

Satchit Sivakumar³  

Boston University

Abstract

We revisit the framework of interactive proofs for distribution testing, first introduced by Chiesa and Gur (ITCS 2018), which has recently experienced a surge in interest, accompanied by notable progress (e.g., Herman and Rothblum, STOC 2022, FOCS 2023; Herman, RANDOM 2024). In this model, a data-poor verifier determines whether a probability distribution has a property of interest by interacting with an all-powerful, data-rich but untrusted prover bent on convincing them that it has the property. While prior work gave sample-, time-, and communication-efficient protocols for testing and estimating a range of distribution properties, they all suffer from an inherent issue: for most interesting properties of distributions over a domain of size N , the verifier must draw at least $\Omega(\sqrt{N})$ samples of its own. While sublinear in N , this is still prohibitive for large domains encountered in practice.

In this work, we circumvent this limitation by augmenting the verifier with the ability to perform an exponentially smaller number of more powerful (but reasonable) *pairwise conditional* queries, effectively enabling them to perform “local comparison checks” of the prover’s claims. We systematically investigate the landscape of interactive proofs in this new setting, giving polylogarithmic query and sample protocols for (tolerantly) testing all *label-invariant* properties, thus demonstrating exponential savings without compromising on communication, for this large and fundamental class of testing tasks.

2012 ACM Subject Classification Theory of computation \rightarrow Interactive computation

Keywords and phrases Distribution Testing, Interactive Proofs

Digital Object Identifier [10.4230/LIPIcs.ITCS.2026.8](https://doi.org/10.4230/LIPIcs.ITCS.2026.8)

Funding Mark Bun: [funding]

Clément Canonne: [funding]

Satchit Sivakumar: [funding]

Acknowledgements I want to thank ...

1 Introduction

Distribution testing, as introduced by [?], is a mature subfield of property testing [? ?] aimed at investigating statistical properties of an unknown distribution given sample access to it. Given a property (a set of distributions) and a proximity parameter $\tau \in (0, 0.1]$,

¹ Optional footnote, e.g. to mark corresponding author

² Optional footnote, e.g. to mark corresponding author

³ Optional footnote, e.g. to mark corresponding author



distribution testing algorithms output **Accept** if the distribution is in the property (or close to it), or **Reject** if the distribution is τ -far from the property, both with high probability. Closeness and farness are quantified with respect to a prespecified notion of distance, typically total variation distance. The primary motivation behind distribution testing is to design testing algorithms for deciding properties with sample complexity sub-linear in the domain size N (which is demonstrably more efficient than learning the distribution, which requires drawing $\Theta(N)$ samples). Accordingly, over the last two decades, researchers have extensively studied the sample complexity of numerous distribution properties, such as simple uniformity testing [?] (testing whether a distribution is uniform over its entire domain), support size decision problem [?] (testing whether a distribution's support is within some pre-specified range), and many more: see, e.g., [6, Chapter 11] and [11, 1, 2] for a more thorough introduction to distribution testing. Unfortunately, although distribution testing is often more efficient than learning the distribution, it is still prohibitively expensive for practical use. For example, it is known that generalized uniformity testing (testing whether a distribution is uniform over its support) over a domain of size N requires $\Omega(N^{2/3})$ samples [?], which can be impractical for large domain sizes. Even simple uniformity testing requires $\Omega(\sqrt{N})$ samples [?], and its *tolerant* testing version (which asks to distinguish distributions *close* to uniform from those which are far) needs $\Omega(N/\log N)$ samples [12].

In the face of these limitations, a nascent line of work [5?, ?, 7] has asked a related question: *with testing being hard by itself, what is the complexity of verifying the properties of a distribution given sample access to it?* Here, in addition to drawing samples from the distribution, the tester is allowed to interactively communicate with an omniscient but *untrusted* prover that knows the distribution in its entirety. The idea here is to leverage the provers extra knowledge about the distribution, with the hope that checking the provers' claims is easier than naively testing the property. While this model of verifiable computation has only recently been explored in the context of distribution testing, it has been an active area of research in other areas of theoretical computer science for over 40 years (see for e.g. [?????]). It models settings where a centralized organization (for example, a company turning billions of dollars of profit) has the ability to collect large amounts of data and learn distributions to high precision, while end-users may not have the same ability. At the same time, the company might have incentives to lie, and so verifying whether the company is being truthful is important in this setting. The work of (author?) [5] shows that the verification of *any* distribution property over domain $[N]$ can be reduced to identity testing⁴, with communication *superlinear* in the domain size. Follow up work [??] recovers this result for the broad class of *label-invariant properties*, while only requiring communication *sub-linear* in the domain size. More specifically, the work of ?? show that for label-invariant properties, verification requires only $O(\sqrt{N})$ samples and $\tilde{O}(\sqrt{N})$ communication, even though, as mentioned earlier, testing some properties in this class could require $\Theta(N/\log N)$ samples. Here, a property is *label-invariant* (also known as *symmetric*) if the names of the elements themselves are not significant to the decision outcome (see Definition ??). Testing if a distribution is uniform over its support (also known as generalized uniformity testing) is an example of a label-invariant property.

Unfortunately, while a significant improvement over unaided testing, requiring $O(\sqrt{N})$ samples from the verifier can still be prohibitive when considering massive domains. Further,

⁴ Identity testing refers to the task of testing if a distribution is exactly equal to a pre-specified reference distribution or is τ -far from it.

there is a matching sample complexity lower bound – verification of even basic label-invariant properties such as checking if a distribution is uniform over its entire domain requires $\Omega(\sqrt{N})$ samples. To summarise: For most properties, with access to *only* samples from a distribution, it is impossible for any tester to do better than drawing $\Omega(\sqrt{N})$ samples, with or without the help of a prover. To bypass these limitations and develop more practical algorithms, in this work we study verifiers that can make a very small number of calls to a more powerful *conditional sampling* oracle. These oracles were introduced in the context of distribution testing [4?]; allowing the tester to condition that samples from the oracle come from a subset S of the domain, of their choosing. The oracle responds with a sample with probability re-normalised over S . If no element in S is supported, the oracle responds with FAIL. Since specifying an arbitrary set may be considered unrealistic for practical purposes, a commonly studied restriction is the pairwise conditional sampling model (PCond), where the specified sets are restricted to be of size exactly 2 or the entire domain (thus, just a regular sample from the distribution). These oracles can be thought of as allowing for local comparisons between the probabilities of two points. While access to a PCond oracle can be significantly helpful for problems like simple uniformity testing, it is unclear from prior work whether it results in more efficient testing for the general class of label-invariant properties. Verification with access to a PCond oracle (or any type of conditional sampling oracle) has also, to the best of our knowledge, not been explored. In our quest to find practical algorithms that work for large domains, we thus ask the following question.

Can label-invariant properties be verified in a (query, sample and communication)-efficient way when the tester has access to a PCond oracle?

1.1 Our Results

Our main result is an *exponential* query complexity separation between testing and verification for testing label-invariant properties with access to a PCond oracle. A detailed accounting of our results and comparison to existing work can be found in Table 1. A description follows.

	Query Complexity Without Prover	Query Complexity With Prover	Communication	Rounds
Samp	$\tilde{\Omega}\left(\frac{N}{\log N}\right)$ [12]	$\tilde{O}(\sqrt{N})$ [? ?]	$\tilde{O}(\sqrt{N})$ [? ?]	2 [? ?]
PCond	$\Omega(N^{1/3})$ (Theorem ??)	$\text{poly}(\log N, \frac{1}{\epsilon})$	$\tilde{O}(\sqrt{N})$ (Theorem ??)	$\text{poly}(\log N, \frac{1}{\epsilon})$

Table 1 Results on testing and verifying label-invariant properties under different types of access to the distribution. We state the best known lower bounds for label invariant properties. For **Samp** the lower bound is for entropy estimation, whereas for **PCond** it is the support size decision problem described in Section ???. The upper bounds apply for all label-invariant properties. Our results are highlighted in bold.

One might have initially hoped that the power of a PCond oracle allows us to test label-invariant properties efficiently, even without the help of a prover. Indeed, with access to the full power of the Cond model (where arbitrary subsets S can be queried and a sample

conditional on S is obtained), [4] show that this class of properties over a domain of size N can be tested with $O(\text{poly log } N)$ queries to the **Cond** oracle. Our first result dashes this hope – we show a lower bound on the number of **PCond** queries required to test label-invariant properties with constant proximity parameter τ , demonstrating that the **PCond** oracle is not much better in the worst case than the sampling oracle for this class of properties. Specifically, we show that a simple variant of the support size distinguishing problem for distributions over a domain of size N requires $\Omega(N^{1/3})$ queries to a **PCond** oracle (the exact same as with access to only a sampling oracle). Thus, unaided, there exist (label-invariant) properties for which the **PCond** oracle is not much better than just sample access.

► **Theorem 1** (Informal Version of ??). *There exists a label-invariant property Π such that every tester with access to a **PCond** oracle for Π with proximity parameter $\tau \leq 1/2$ and failure probability 0.01 must make $\Omega(N^{1/3})$ queries.*

The above lower bound motivates the investigation of verification with access to a **PCond** oracle. As mentioned earlier, [5, Proposition 3.4] showed that with super-linear communication complexity, there exists a reduction from verification to identity testing. Instantiating this reduction with an identity tester using **PCond** oracles [10, Theorem 1.5], we get that there exists an interactive proof system for every property with super-linear communication complexity that makes only $O(\sqrt{\log N}/\tau^2)$ queries to the **PCond** oracle. However, super-linear communication is also prohibitive for practical algorithms; the proof systems by [?] require the prover to only communicate $\tilde{O}(\sqrt{N})$ domain elements, but still achieve the sample complexity of identity testing (for the class of label-invariant properties). Could we also hope to achieve such communication while maintaining similar query complexity as that of identity testing? The main result of this paper is an affirmative answer to this question. Specifically, we give an interactive proof system for tolerantly verifying *any* label-invariant property that has communication complexity $\tilde{O}(\sqrt{N})$ and query complexity $\text{poly}(\log N)$ (suppressing the dependence on the proximity parameter).

► **Theorem 2** (Informal Label-Invariant Tolerant Verification Theorem (Theorem ??)). *Fix a label-invariant property Π over a domain $[N]$ and proximity parameters $\tau_c, \tau_f \in (0, 1/2]$. There exists a polylogarithmic (in N) round interactive protocol Π between an honest verifier T , and an omniscient untrusted prover $\mathsf{P}^{\mathcal{D}}$, where the verifier has **PCond** access to \mathcal{D} , such that at the end of the interaction the verifier satisfies the following conditions:*

1. **Completeness:** *If the prover follows the protocol as prescribed, and $d_{TV}(D, \Pi) \leq \tau_c$, then*

$$\Pr \left[\text{out} \left[\Pi \left(\mathsf{P}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; \tau_c, N \right) \right] = \text{Accept} \right] \geq 2/3$$

2. **Soundness:** *If $d_{TV}(D, \Pi) \geq \tau_f$, then for any prover $\widetilde{\mathsf{P}}^{\mathcal{D}}$*

$$\Pr \left[\text{out} \left[\Pi \left(\widetilde{\mathsf{P}}^{\mathcal{D}}, \mathsf{T}^{(\mathcal{D})}; \tau_c, N \right) \right] = \text{Reject} \right] \geq 2/3$$

The complexity of the verifier is as follows:

1. **Query Complexity + Sample Complexity:** $O(\text{poly}(\log N, 1/(\tau_f - \tau_c)))$

2. **Communication Complexity:** $\tilde{O}(\sqrt{N} \text{poly}(1/(\tau_f - \tau_c)))$

In the process of proving this result, we give protocols for more basic primitives that may be of independent interest. Most significantly, we give an interactive proof system that is able to calculate the approximate probability mass of any point⁵ in the domain using communication complexity $\tilde{O}(\sqrt{N})$ and query complexity $O(\text{poly}(\log N, 1/\tau))$. As we will explain in the techniques section to follow, this is a key technical workhorse in our protocol for verifying label-invariant properties.

► **Theorem 3** (Informal Version of ??). *Fix a domain $[N]$. For every $\tau > 0$ and $\delta \in (0, 1/2)$, there exists a $O(\text{poly}(\log N, \log 1/\delta, \frac{1}{\tau}))$ -round interactive proof system such that the verifier T with access to a PCond oracle satisfies the following.*

1. **Completeness:** For every distribution \mathcal{D} , if the prover $\mathsf{P}^{\mathcal{D}}$ is honest, then

$$\Pr \left[\mathsf{T} \text{ outputs } (y^*, \tilde{\mathcal{D}}[y^*]) \text{ s.t. } \frac{\tilde{\mathcal{D}}[y^*]}{\mathcal{D}[y^*]} \in \left[\frac{1}{(1+\tau)}, 1+\tau \right] \right] \geq 1 - \delta$$

2. **Soundness:** For any cheating prover $\tilde{\mathsf{P}}^{\mathcal{D}}$, then

$$\Pr \left[\mathsf{T} \text{ outputs Reject} \vee \mathsf{T} \text{ outputs } (y^*, \tilde{\mathcal{D}}[y^*]) \text{ s.t. } \frac{\tilde{\mathcal{D}}[y^*]}{\mathcal{D}[y^*]} \in [1/(1+\tau)^2, (1+\tau)^2] \right] \geq 1 - \delta$$

The complexity of the verifier is as follows:

1. **Query Complexity + Sample Complexity:** $O(\text{poly}(\log N, 1/(\tau)))$
2. **Communication Complexity:** $\tilde{O}(\sqrt{N} \text{poly}(1/(\tau)))$

1.2 Technical Overview

In this section, we give an overview of our protocol for verifying label-invariant properties.

1.2.0.1 Unlabelled Bucket Histogram:

There is now a long line of work on the testing and verification of label-invariant properties [? ? 4? ?], and a key object used in this work is the unlabelled *approximate* τ -bucket histogram of a distribution. Bucketing corresponds to partitioning the interval $[0, 1]$ into smaller multiplicative probability intervals (see Definition ??). The τ -bucket histogram divides the interval $[0, 1]$ into $\tilde{O}(\log N/\tau)$ buckets where the ℓ^{th} bucket is a set of domain elements with individual probability mass in the range $(\tau(1+\tau)^\ell/N, \tau(1+\tau)^{\ell+1}/N]$. The *approximate* unlabelled bucket histogram of a distribution then corresponds to a list of $\tilde{O}(\log N/\tau)$ fractions, where the ℓ^{th} element of the list is the fraction of domain points whose probability lies in the range specified by the ℓ^{th} bucket (see Definition ??). It is well known (see [?]) that the *approximate* unlabelled τ -bucket histogram of a distribution is a sufficient statistic to (tolerantly) test any label-invariant property with proximity parameter(s) $O(\tau)$. Thus, similar to prior work [? ? 7], our protocol also focuses on efficiently verifying an unlabelled τ -bucket histogram given to us by the prover.

⁵ Provided it does not have prohibitively small probability mass in its neighborhood.

182 1.2.0.2 Using Pairwise Comparisons to Learn Bucket Histogram:

183 Note that the unlabelled bucket histogram is a distribution over the buckets of the τ -bucket
 184 histogram and is hence a distribution over a domain of size $\tilde{O}(\log \frac{N}{\tau})$. Hence, by standard
 185 results in distribution learning, $\tilde{O}(\log N/\tau^2)$ samples from this bucket distribution would be
 186 sufficient to learn it. However, sampling from this bucket distribution is non-trivial since a
 187 sample from the original distribution \mathcal{D} does not come with information about histogram
 188 bucket index.

189 While sampling from the bucket distribution might be hard with **Samp** access to the
 190 distribution, one might be more optimistic about the possibility of sampling from the τ -
 191 approximate histogram with **PCond** queries. In particular, one approach that we might take is
 192 the following: the verifier draws a dataset of size $\tilde{O}(\log N/\tau^2)$ (enough to learn the histogram),
 193 and sends these samples to the prover, who responds with the bucket index of each sample.
 194 From how a τ -histogram is defined, if x and y belong to buckets i and j respectively, then
 195 this implies that the ratio of the probability masses of x and y under \mathcal{D} is guaranteed to
 196 be in the interval $\left[\frac{1}{(1+\tau)^{|j-i|}}, (1+\tau)^{|j-i|}\right]$. As **PCond** access allows us to conditionally sample
 197 from a set restricted to two domain elements, it allows us to approximately learn the ratio of
 198 their probability masses up to a multiplicative constant (see Lemma ??). Equipped with this
 199 power, for each pair $x \neq y$ from the set of drawn samples, the verifier uses the **PCond** oracle to
 200 check if the learned ratios align with the provers claims. If the prover were to lie significantly,
 201 then for at least one pair of samples, the claimed ratio would significantly different from
 202 the learned ratio. Unfortunately, this simplistic strategy comes with two pitfalls. Firstly,
 203 assuming the above strategy was sound, naively comparing elements with arbitrary bucket
 204 indices could require $\Omega(N)$ **PCond** queries if the elements being compared had significantly
 205 different probability masses (see the ??, wherein K could be as large as N). This would be as
 206 bad as learning the distribution itself. Secondly, and more importantly, the above strategy is
 207 *not* sound. It does not catch a prover that “slides” all samples into different buckets *in the*
 208 *same way*, i.e., it lies about *every* bucket index by the same offset. As an example, consider
 209 two distributions: \mathcal{D}_1 is the uniform distribution over $N^{1/4}$ domain elements and \mathcal{D}_2 is the
 210 uniform distribution over \sqrt{N} domain elements. The bucket histograms of both involve a
 211 unit mass on a single (but different) bucket. However, pairwise comparisons between samples
 212 taken from either distribution will always reveal a ratio that is approximately 1, since the
 213 probabilities of all elements in the support of both distributions are identical. Hence, given
 214 distribution \mathcal{D}_1 , the prover can output the bucket histogram of distribution \mathcal{D}_2 , and it is
 215 impossible for a verifier to catch it purely by using the test described above.

216 A remedy to these obstacles lies in the following observation. If we had a good estimate
 217 of the probability mass of a *single point* y in the domain, then we could resolve the soundness
 218 issue discussed above. We simply use the **PCond** oracle to learn the ratio of probabilities
 219 between each of our samples and y . Using this ratio, and knowledge of the approximate mass
 220 of y , we can compute estimates of the probabilities of all samples. This would squash the
 221 sliding attack described above. To deal with the first issue (that of the probability mass of y
 222 being very far from that of a sample), we would need to do more than learn just one value of
 223 y . Instead, we could learn the mass of a point y_j , for every bucket j that has large enough
 224 mass. This way, for any sample x in the tester’s set of samples (which are likely to come
 225 from buckets with sufficiently large mass), we can find some y_j that is in a near-by bucket
 226 with high probability.

1.2.0.3 Verifying the probability of points:

Given that we simply need to identify the probability of a few points in the domain, one might expect that this could be done even without access to a prover — this would give a query-efficient tester for label-invariant properties. However, this ends up being a surprisingly challenging task. Indeed, our lower bound in Theorem 1 shows that this is impossible (if we wanted to bypass the $\text{poly}(N)$ lower bound). This indicates a power of an untrusted prover; it is able to certify the probability of a few points in the distribution support. Indeed, the proof system with super-linear communication [5] achieves something stronger- it certifies the entire distribution. Since we only need to certify the mass of at most $O(\log N)$ points, we ask if this be done in a more communication efficient way?

1.2.0.4 Support Size Verification:

Inspired by the “sliding” cheating prover from earlier, we consider the orthogonal but related problem of verifying the support size of a flat distribution.⁶ We will subsequently show that a protocol for this problem can be combined with the ability of a PCond oracle to learn neighbourhoods around points, enabling us to solve the probability approximation problem for “relevant” domain elements.

Given a support size claim represented by four numbers A', A, B, B' , corresponding to the claim that $A' < A \leq \text{Supp}(\mathcal{D}) \leq B < B'$, our hope is to accept if the claimed support size range is accurate, and reject if the true support is larger than B' or smaller than A' . Given different values of A and B , we develop a number of tests to verify with *sub-linear* communication, the support size assuming the distribution is uniform over its support⁷. We summarize the ideas below.

If the claimed support size upper bound B is small (that is, $O(\sqrt{N})$), we could ask the prover to send us the claimed support of the distribution. If the prover lies, and the true support is actually much larger, then taking a few samples from the distribution would give us a domain element outside the claimed support, thus catching the lie of the prover. If the true support is much smaller than A , then taking a number of uniform samples from the claimed support sent by the prover would result in a sample outside the support of the distribution, which could be easily detected with a few PCond queries (see Lemma ??). This gives us a protocol with a constant number of queries, and communication complexity roughly $O(B)$.

On the other hand, if the claimed support size lower bound A is large (that is, $\omega(\sqrt{N})$), then asking the prover to send the support is not communication-efficient. Our approach instead involves using uniform samples from the domain. The first test is to catch provers that lie that the support is much larger than it really is. It involves drawing $O(N/A)$ uniform samples S_1 from the domain, and sending them to the prover. We ask the prover to send back a sample in this subset that is in the support of the distribution. If the true support is much smaller than A , there are (with high probability) no samples in the support of the distribution in S_1 , and we can ensure the prover does not cheat by checking whether any element it sends back is in the support using a constant number of PCond queries. The second test catches provers that lie that the support is much smaller than it really is. It involves drawing $O(N/B')$ samples z_1, \dots, z_m uniformly from the domain and permuting

⁶ A distribution is *flat* if it is uniform over its support.

⁷ We relax this condition in the main protocol, but assuming uniformity makes the description more intuitive.

269 them with one sample x taken from the distribution. We ask the prover to identify the
 270 index of x in the permuted set. If the true support is smaller than B , then w.h.p, we expect
 271 that none of z_1, \dots, z_s are in the support of the distribution and hence, the honest prover
 272 can identify x exactly. On the other hand, if the support is larger than B' , we expect that
 273 at least one of z_1, \dots, z_s is in the support of the distribution, and the prover is unable to
 274 tell what the sample inserted by the prover was (since it could be x or z_i). This gives us a
 275 protocol with a constant number of queries, and communication complexity roughly $O(N/A)$.
 276 Balancing parameters in these tests to optimize the communication complexity, we get an
 277 overall protocol for support size verification with communication complexity roughly \sqrt{N} .

278 We emphasize that the above outline is a simplification of the truth, and sweeps important
 279 details under the rug. Recall that our main goal is to certify the histogram of a distribution.
 280 In an attempt to do so, we will use the support size protocol above repeatedly as a sub-routine.
 281 This requires bounding the soundness and completeness errors in a meaningful way. As
 282 described above, these protocols do not have low enough soundness and completeness error
 283 to be used as sub-routines. Additionally, the above description assumes that distributions
 284 are exactly flat. In practice this will not be the case, and we need to be able to handle
 285 distributions where the probability ratio between any two elements in the support is upper
 286 bounded by some constant α (nearly flat). In Section ??, we show how to analyse the
 287 protocols above to facilitate amplification of soundness and completeness errors, and make
 288 the protocol work for the more general class of α -flat distributions.

289 1.2.0.5 Estimating Probability of a Point using Support Size Verification:

290 Finally, we explain how we can use the support size protocols to estimate the probability of
 291 a point. Prior work [3] using the PCond oracle has shown that it can be used to estimate the
 292 mass within a multiplicative $(1 + \tau)$ -neighbourhood of any point⁸ (see ??). We ask the prover
 293 to send us a point y^* ⁹ with sufficiently large mass in its neighbourhood (by an averaging
 294 argument, at least one histogram bucket needs to have $\tau/\log N$ mass, ensuring that such a
 295 point exists, see Claim ??), and tell us the bucket the y^* belongs to. We then use the PCond
 296 oracle to estimate the mass within the multiplicative neighbourhood of y^* . The learned
 297 mass of the neighborhood divided by the prover's claimed probability mass¹⁰ of y^* gives us
 298 bounds on the number of elements in y^* 's neighbourhood. Additionally, by definition the
 299 neighbourhood of y^* will be nearly flat. Hence, we have reduced the problem to a claim
 300 about the support size of a nearly-uniform distribution over a subset of the domain. Observe
 301 that we can sample from the distribution restricted to this bucket using PCond queries—since
 302 there is sufficient mass in the neighbourhood of the point, $O(\text{poly log } N)$ samples from the
 303 distribution will contain at least one sample from the bucket, and we can use PCond queries
 304 between the samples and y^* to find out which sample it is (see ??). If the prover was telling
 305 the truth (or rather did not lie egregiously), then the support size claim holds and the verifier
 306 will accept, thereby giving us a point and its approximate probability mass¹¹. If the prover
 307 significantly lied, then the support size claim will be false and the prover will be caught. We
 308 note that the final analysis needs to handle some additional subtleties, since the PCond oracle

⁸ As long as the point does not have prohibitively small probability mass under the distribution.

⁹ Recall that in the final protocol, we will ask the prover to send us points from every bucket with sufficiently large mass. For simplicity, we consider a single point in this description.

¹⁰ The bucket index gives a lower and upper bound on the true probability mass of y^* . This is sufficient to catch a cheating prover.

¹¹ The claimed mass of y^* is derived from the bucket sent by the prover.

is itself only approximate and does not provide perfect comparisons (which can affect the sampling), and additionally the bucket boundaries and the neighborhood of the provided point do not overlap precisely. We refer to Sections ?? and ?? for the details. Once we have estimated the probability of important points, we can use the ratio learning techniques discussed earlier to certify the prover's claim about the bucket histogram of the distribution.

1.3 Other Related Work:

1.3.0.1 Interactive Proofs for Distribution Testing:

As mentioned earlier, interactive proofs for verifying distribution properties were first introduced in the work of [5]. Follow-up work [?] studied interactive proofs for verifying *label-invariant* properties, focusing on sublinear communication, and doubly efficient protocols (i.e. computationally-efficient and sample-efficient generation of a proof). [7] studied *public-coin* interactive proofs for testing label-invariant properties, where the verifier has no private randomness. [8] gives communication-efficient and sample-efficient interactive proofs for more general distribution properties that can be decided by uniform polynomial-size circuits with bounded depth. [9] introduces *computationally sound* interactive proofs and shows communication-, time-, and sample-efficient protocols for any distribution property that can be decided in polynomial time given explicit access to the full list of distribution probabilities. All of the above works assume that the verifier only has sample access to the distribution, and the verifier sample complexity in all of them is $\Omega(\sqrt{N})$ (where N is the size of the domain).

1.3.0.2 Interactive Proofs for Learning:

A related but orthogonal line of work [?] focuses on interactive proofs for verifying learning problems- for a specific hypothesis class H , given access to an untrusted prover, the goal is for a verifier to output an accurate hypothesis from H for an underlying unknown distribution D if the resource-rich prover is honest, and to reject if the prover lies egregiously. Different types of resource asymmetry between the prover and the verifier are explored in these papers – including differing number of samples, different computational complexities, different types of access to the underlying function (sample vs query), and differing access to computational resources (classical vs quantum computation and communication).

1.3.0.3 Distribution testing under conditional oracles:

Our work focuses on interactive proofs for *verifying* distribution properties under conditional sampling models. There is a long line of work on *testing* with access to conditional samples. (author?) [4?] introduced the conditional sampling (Cond) model and its more restricted variants (PCond, ICond). They gave algorithms for uniformity testing, tolerant uniformity testing, identity testing etc. in these conditional sampling models with query and sample complexity significantly better than the Samp model. Follow-up work shows improved bounds for identity testing (and its tolerant version), tolerant uniformity testing, and new algorithms for other tasks such as equivalence testing and support size problem in the Cond model [10?]. There is also a line of work studying the power of non-adaptive queries in the conditional sampling model [?]. The PCond model was studied in detail by [10] who gave optimal bounds for identity testing and tolerant uniformity testing in this model, improving on results from [3]. Testing under other types of conditional sampling has also been studied in the literature including subcube conditioning, where the distribution is supported on the

hypercube, and the tester is allowed to ask for conditional samples from subcubes [? ? ? ? ?], and coordinate conditional sampling [?] (a version of subcube conditioning where all but one coordinate is fixed to a specific configuration and a sample is obtained from the remaining coordinate). Testing under other types of access to the distribution such as Probability Mass Function queries or Cumulative Distribution Function queries has also been studied in the literature [? ? ? ? ?].

References

- 1 Clément L. Canonne. *A Survey on Distribution Testing: Your Data is Big. But is it Blue?* Number 9 in Graduate Surveys. Theory of Computing Library, 2020. URL: <http://www.theoryofcomputing.org/library.html>, doi:10.4086/toc.gs.2020.009.
- 2 Clément L. Canonne. Topics and techniques in distribution testing: A biased but representative sample. *Foundations and Trends® in Communications and Information Theory*, 19(6):1032–1198, 2022. URL: <http://dx.doi.org/10.1561/0100000114>, doi:10.1561/0100000114.
- 3 Clément L. Canonne, Dana Ron, and Rocco A. Servedio. Testing probability distributions using conditional samples. *SIAM Journal on Computing*, 44(3):540–616, 2015. URL: <https://epubs.siam.org/doi/10.1137/130945508>.
- 4 Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 561–580, 2013. URL: <https://arxiv.org/abs/1210.8338>.
- 5 Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. URL: <https://drops.dagstuhl.de/storage/00lipics/lipics-vol094-itcs2018/LIPIcs.ITCS.2018.53/LIPIcs.ITCS.2018.53.pdf>.
- 6 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
- 7 Tal Herman. Public coin interactive proofs for label-invariant distribution properties. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2024. URL: <https://drops.dagstuhl.de/storage/00lipics/lipics-vol317-approx-random2024/LIPIcs.APPROX-RANDOM.2024.72/LIPIcs.APPROX-RANDOM.2024.72.pdf>.
- 8 Tal Herman and Guy N. Rothblum. Interactive proofs for general distribution properties. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 528–538. IEEE, 2024. doi:10.1109/FOCS61266.2024.00041.
- 9 Tal Herman and Guy N. Rothblum. How to verify any (reasonable) distribution property: Computationally sound argument systems for distributions. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL: <https://openreview.net/forum?id=GfXMTAJaxZ>.
- 10 Shyam Narayanan. On tolerant distribution testing in the conditional sampling model. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '21*, pages 357–373, USA, 2021. Society for Industrial and Applied Mathematics. URL: <https://dl.acm.org/doi/10.5555/3458064.3458087>.
- 11 Ronitt Rubinfeld. Taming big probability distributions. *XRDS: Crossroads, The ACM*

- 397 *Magazine for Students*, 19(1):24, sep 2012. URL: [http://dx.doi.org/10.1145/2331042.](http://dx.doi.org/10.1145/2331042.2331052)
398 [2331052](http://dx.doi.org/10.1145/2331042.2331052), doi:10.1145/2331042.2331052.
- 399 12 Gregory Valiant and Paul Valiant. Estimating the unseen: improved estimators for
400 entropy and other properties. *Journal of the ACM (JACM)*, 64(6):1–41, 2017. URL:
401 <https://dl.acm.org/doi/10.1145/3125643>.