Beyond the Worst-Case Analysis of Algorithms

Edited by
Tim Roughgarden

Contents

1

Instance Optimal Distribution Testing and Learning

Gregory Valiant and Paul Valiant

Abstract

This chapter considers the challenge of saying as much as possible about a probability distribution given a limited number of samples. Traditionally, work has focused on either developing algorithms that are optimal in an asymptotic sense as the amount of data goes to infinity, or developing algorithms that are optimal in a worst-case sense when parameterized by relevant quantities such as the support size. This chapter, by contrast, considers two standard settings—*learning* a discrete distribution from samples, and *testing* whether a set of samples was drawn from a specific distribution—and develops algorithms that are near optimal *on every instance*.

1.1 Testing and Learning Discrete Distributions

This chapter revisits some of the most basic distributional learning and hypothesis testing problems, with the goal of designing algorithms for these tasks that are optimal in stronger senses than classical worst-case analysis. The first portion of the chapter considers the problem of learning a discretely supported distribution from independent draws. To motivate the results presented here, it is worth first considering the naive approach that simply returns the empirical distribution of the samples. The empirical distribution is optimal in a strong worst-case sense: for every error parameter $\epsilon > 0$, and integer k, given $n = k/\epsilon^2$ independent draws from a distribution supported on $\leq k$ elements, the expected total variation distance between the true distribution and the empirical distribution of the samples is bounded by $O(\epsilon)$. Furthermore, for worst-case distributions supported on k elements, there is no algorithm that can achieve expected error ϵ with $n = o(k/\epsilon^2)$ samples. However, despite this worst-case optimality of the empirical estimator, one might hope to do significantly better than this naive algorithm for the many non-worst-case distributions that have exploitable structure. Indeed, this chapter presents an "instance-optimal" algorithm which, in a concrete sense, opti-

mally leverages whatever structure is present in the distribution in question, even without any prior knowledge of this structure.

The second portion of this chapter considers the following basic hypothesis testing problem, sometimes referred to as "identity testing": given the description of a distribution, p, error tolerance $\epsilon > 0$, and n independent draws from an unknown distribution, q, distinguish the case that $p = q$ versus the case that p and q have total variation distance at least ϵ. Pearson's classical *chi-squared* test is one of the most commonly used algorithms for this problem, though is far from optimal in the regime in which one expects many domain elements to be observed zero or once among the n samples. Beginning in the early 2000's, new algorithms were developed that pinned down the optimal sample size necessary for performing this test, for worst case distributions, p, parameterized in terms of their support size. This chapter instead provides a variant of Pearson's chi-squared test which is optimal for every distribution, p, defined over a discrete countable support, and error parameter ϵ. Analyzing this algorithm yields a clean expression representing the sample complexity of testing the identity of a distribution p, as a function of p and ϵ.

Both the instance optimal learning, and the identity testing portions of this chapter, require some machinery that is conceptually and technically interesting beyond the direct applications to the problems at hand. This chapter provides self-contained treatments of this material. Section 1.2.3 describes an algorithm which, given n samples from distribution p, accurately recovers the multiset of the probabilities of the distribution, essentially as accurately as the empirical estimate would be when given $n \log n$ samples. Section 1.4 describes an efficient algorithm for proving inequalities of a certain form; such inequalities arise in the analysis of the identity testing algorithm and in many other settings in theoretical computer science.

1.2 Instance Optimal Distribution Learning

Given independent draws from an unknown distribution over an unknown discrete support, what is the best way to aggregate those samples into an approximation of the distribution? The most obvious approach is to simply return the empirical distribution of the samples. To what extent can one improve over this naive approach?

If one knew, a priori, that the distribution in question possessed some special structure, that information could plausibly be leveraged to "de-noise" the empirical distribution. For example, if one knew that the distribution is uniform over its domain, then one simply needs to identify the support of the distribution and estimate the support size. Both of these tasks may seem easier than estimating the probability of each element. A more realistic scenario might be where one knows that the distribution has a discrete power-law probability profile (where the ith most likely domain element has probability roughly proportional to $1/i^s$ for some

constant, s, such as in Zipfian distributions); such information could plausibly be leveraged to "correct" the empirical distribution, by nudging empirical probabilities so that they fit with our prior knowledge of the overall power-law shape.

Is there an algorithm which, for *every* input distribution, optimally leverages whatever "structure" is present, *without any prior information about the type of structure*? This chapter shows that the answer is *yes*, up to a subconstant additive term, if one interprets the "structure" of a distribution to mean any property, such as support size or entropy, that is invariant to permuting or relabeling the domain elements—namely any function of the multiset of probabilities with which elements occur.

For the sake of both the construction of this instance-optimal learning algorithm, as well as its analysis, we will first define an unachievably good benchmark that will quantify how helpful the structure in the given instance actually is. This benchmark will correspond to the expected performance of an algorithm that receives extra information about the distribution in addition to the samples, and then uses this extra information and samples optimally. Specifically, this extra information will be the complete description of the distribution in question, with the labels removed. This extra information can be canonically represented as access to the sorted vector of probabilities with which domain elements occur: $p_1 \geq p_2 \geq p_3 \geq \ldots$. As it turns out, it will not be too difficult to reason about the structure of an algorithm which uses this extra information optimally. The analysis proceeds by designing an algorithm that does not receive this extra information, yet still emulates this unachievably good benchmark. The analysis concludes by showing that this algorithm, on every input distribution, performs nearly as well as the unachievably good benchmark.

Definition 1.1. Let $ErrOpt^*(p, n)$ be the minimum expected ℓ_1 error that any algorithm could achieve on the following learning task: given the description of p, and n samples drawn independently from a distribution p' that is identical to p up to an arbitrary relabeling of the domain, learn the distribution p'. Let Opt^* be the corresponding algorithm, which takes as input both the samples, and the vector of (permuted) probabilities.

The following theorem summarizes the sense in which an instance-optimal algorithm will, for every discrete distribution, learn nearly as well as if it already knew the distribution, up to relabeling the domain.

Theorem 1.2. *The Instance Optimal Learning Algorithm, outlined in Algorithm 1.1, when given n independent draws from any distribution p of discrete support, outputs a labeled vector q, such that with probability at least $1 - n^{-\omega(1)}$,*

$$||p - q||_1 \leq ErrOpt^*(p, n) + 1/polylog(n).$$

The proof of Theorem 1.2 boils down to arguing that if there are sufficient samples for Opt^* to accurately assign the labels of domain elements to the ground truth

(unlabeled) vector of probabilities, then one has enough samples to approximately learn the vector of unlabeled probabilities from only the samples. From there the Instance Optimal Learning Algorithm may emulate Opt^*, losing only a $1/polylog(n)$ additive error in comparison to this benchmark. It turns out that the $1/polylog(n)$ additive error term is necessary in the above statement. If, however, one allows a multiplicative constant in front of $ErrOpt^*$, then a slightly different algorithm allows the $1/polylog(n)$ error term to be improved to $O(1/poly(n))$. In both cases, this error term is a function of n only, and in particular is independent of the distribution in question.

Theorem 1.3. *The Good–Turing Denoising Algorithm, described in Algorithm 1.2, when given n independent draws from any distribution p of discrete support, outputs a labeled vector q, such that with probability at least $1 - n^{-\omega(1)}$,*

$$||p - q||_1 \leq 2 \cdot ErrOpt^*(p, n) + \widetilde{O}(1/n^{1/6}).$$

One surprising implication of the above results is that for large sample sizes, n, prior knowledge of the "shape" of the distribution, or knowledge of the rate of decay of the tails of the distribution, cannot significantly improve the accuracy of the learning task. For example, Theorem 1.2 implies that typical Bayesian assumptions that the frequency of words in natural language satisfy Zipf distributions, or that the frequencies of different species of bacteria in the human gut satisfy Gamma distributions or various other power-law distributions, can improve the expected error of the learned distribution by at most a vanishing function of the sample size.

The following two examples highlight the limits and power of these results.

Example 1.4. Let $p = Unif(2)$ denote a distribution supported on two domain elements, with each element having probability $1/2$. The unattainable benchmark Opt^* just needs to learn the support, and hence for $n \geq 1$ the error is simply $1/2$ times the probability of not having observed both elements, namely $ErrOpt^*(Unif(2), n) = \frac{1}{2}\frac{1}{2^{n-1}} = 2^{-n}$. On the other hand, without prior knowledge of the probabilities, no algorithm can use n coin flips to learn the probabilities of a (possibly biased) coin to error $o(1/\sqrt{n})$.

Example 1.4 illustrates that prior knowledge of the vector of probabilities can be very helpful, reducing the expected error from inverse polynomial to inverse exponential. This demonstrates that, in general, one should not hope to achieve a result of the form of Theorem 1.2 without an additive error term. The fact that the error term of Theorem 1.2 vanishes as a function of the sample size, independent of the distribution, implies that for sufficiently large n there is no distribution for which prior knowledge of the probabilities can be leveraged to improve the expected error by a constant. The following example examines a natural extension of the above setting, where both $Opt*$, as well as the instance-optimal algorithm,

achieves significantly less error than the naive algorithm that returns the empirical distribution of the samples.

Example 1.5. Let $p = Unif(k)$ correspond to a uniform distribution over k elements. The Opt^* benchmark needs to learn the support of the distribution, as it knows, *a priori*, that every domain element that occurs with nonzero probability occurs with probability $1/k$. Hence, given n samples, the expected error is $1/k$ times the expected number of unobserved domain elements:

$$ErrOpt^*(Unif(k), n) = \Pr[Binomial(n, 1/k) = 0] \approx e^{-n/k}.$$

In comparison, the empirical distribution of the samples will have expected error

$$ErrEmp(Unif(k), n) = \frac{k}{n} \boldsymbol{E}[|Binomial(n, 1/k) - n/k|] \approx \min(\sqrt{k/n}, 1).$$

When $n = c{\cdot}k$, these two expected errors differ drastically, with $ErrOpt^* \approx exp(-c)$ versus $ErrEmp \approx 1/\sqrt{c}$. For example, if $c = 10$, then each domain element will show up 10 times in expectation; if a domain element shows up 9 or 11 times, then the empirical estimator will over- or underestimate its probability respectively, while the optimal estimator will treat such over- or under representation in the samples identically, making an error only in the exponentially ($\approx e^{-10}$) unlikely case that such a domain element is entirely absent from the samples.

Theorem 1.2 shows that the Instance Optimal Learning Algorithm achieves error $exp(-c) + o_n(1)$, which will be close to $ErrOpt^*$ as long as n is large.

1.2.1 Understanding the Benchmark, Opt^*

Rather than directly trying to understand Opt^*, the analysis will instead consider an algorithm that receives *even more* additional information. Suppose you were given a set of samples, and, for each integer $i \geq 1$, you were told the multiset of probabilities of the domain elements observed exactly i times. With this additional information, the optimal algorithm is easy to describe:

First, for all integers i, the optimal algorithm will assign the same probability to all domain elements that are observed exactly i times. If this was not the case, it is not hard to show that there would exist a labeled distribution on which the algorithm would perform sub-optimally. Given this, the question is what probability to assign? The following fact, whose proof is left as an exercise (Exercise 1.1) provides the answer.

Fact 1.6. *Given a multiset of real numbers, $S = \{s_1, \ldots, s_m\}$, setting $x = median(S)$ minimizes the sum of absolute differences between elements of S and x:*

$$\sum_{i=1}^{m} |s_i - median(S)| = \inf_{x \in \mathbf{R}} \sum_{i=1}^{m} |s_i - x|.$$

Hence we arrive at the following high-level sketch of the instance-optimal algorithm emulating Opt^*:

Algorithm 1.1. *Sketch of the Instance Optimal Learning Algorithm that emulates Opt^*, achieving the instance-optimal guarantees of Theorem 1.2.*

Input: A set of n independent draws from an unknown distribution.
Output: A labeled vector of probabilities.

1. Use the samples to accurately reconstruct the unlabeled vector of probabilities, as described in Section 1.2.3.
2. For each $i \geq 1$, leverage the reconstructed vector to approximate the expected median probability of elements occurring i times, and assign that probability to all domain elements that occurred i times.

The most involved component of the above algorithm is reconstructing the unlabeled vector of probabilities. This is an extremely useful primitive, independent of our current goals of instance-optimal learning. Section 1.2.3 sketches the approach to this reconstruction problem, and describes some of the other applications of this subroutine. The complete proof of Theorem 1.2 is quite involved, though can be interpreted as arguing that, up to the additive $1/\operatorname{polylog} n$ error term, one can always recover an approximation of the unlabeled vector of probabilities more accurately than one can disambiguate and label such a vector. This chapter will not cover the details of this proof, and we refer the interested reader to Valiant and Valiant (2016).

One of the difficulties in proving Theorem 1.2 is that the median is not especially well behaved, and it is tricky to design unbiased (or little-biased) estimators for the median of a distribution. The *mean*, however, is extremely well behaved, and can be leveraged to construct a simple algorithm that is easy to analyze, which achieves the guarantees of Theorem 1.3. The following fact summarizes the crucial property of the mean used in the analysis (see Exercise 1.2 for its proof):

Fact 1.7. *Given a multiset of numbers, $S = \{s_1, \ldots, s_m\}$, setting $x = mean(S)$ is at most a factor of 2 from minimizing the sum of absolute differences between elements of S and x: $\sum_{i=1}^{m} |s_i - mean(S)| \leq 2 \inf_{x \in \mathbf{R}} \sum_{i=1}^{m} |s_i - x|$.*

The following section describes the *Good–Turing* frequency estimation scheme, which provides an extremely simple estimator for the expected average probability of elements observed i times. This estimator leads to the simple algorithm achieving the guarantees of Theorem 1.3.

1.2.2 Good–Turing Frequency Estimation and Proof of Theorem 1.3

In the context of the British WWII code breaking efforts at Bletchley Park, I.J. Good and Alan Turing developed a slick approach to estimating simple functionals

of discrete distributions, including the amount of "missing mass"—the total probability mass comprised by elements that have *not* been observed in a given set of samples. At a high level, their approach is to write an expression for the quantity of interest, and then re-express that as a linear combination of terms, $\sum_{j\geq 1} c_j \boldsymbol{E}[F_j]$, where F_j denotes the number of elements observed exactly j times. Given that F_j will be tightly concentrated about its expectation, this will yield a good estimate, provided the coefficients, c_j, are not too large.

We begin by instantiating the above approach for the task of estimating the expected total probability mass comprised of elements observed exactly i times, yielding a variant of what is often referred to as *Good–Turing Frequency Estimation*.

Proposition 1.8. *Given n independent draws from a distribution p, let F_i denote the number of elements observed exactly i times, and let m_i denote the total probability mass comprised by such elements. Then for $i < n$,*

$$\boldsymbol{E}[m_i] = \frac{i+1}{n-i}\,\boldsymbol{E}[F_{i+1}] + O((i+1)/(n-i)).$$

Proof We begin by rewriting the expression for $\boldsymbol{E}[m_i]$ in terms of $\boldsymbol{E}[F_{i+1}]$. In the following, the summations are over the domain of the distribution, and $p(x)$ denotes the probability that the distribution assigns to element x.

$$\begin{aligned}
\boldsymbol{E}[m_i] &= \sum_x p(x)\Pr[Binomial(n,p(x)) = i] \\
&= \sum_x p(x)(p(x))^i(1-p(x))^{n-i}\binom{n}{i} \\
&= \frac{i+1}{n-i}\sum_x (p(x))^{i+1}(1-p(x))^{n-i}\binom{n}{i+1} \\
&= \frac{i+1}{n-i}\sum_x (1-p(x))\Pr[Binomial(n,p(x)) = i+1] \\
&= \frac{i+1}{n-i}\,\boldsymbol{E}[F_{i+1}] - \frac{i+1}{n-i}\sum_x p(x)\Pr[Binomial(n,p(x)) = i+1].
\end{aligned}$$

The second term on the last line is bounded by 1, since $\sum_x p(x) = 1$ and each binomial probability is at most 1, yielding the proposition. □

We are now prepared to describe the algorithm to which Theorem 1.3 applies. For elements observed many times, it uses their empirical probabilities, and for elements observed few times, it leverages the above Good–Turing estimate for the expected amount of probability mass comprised by such elements.

Algorithm 1.2. *The Good–Turing Denoising Algorithm, achieving the instance-optimal guarantees of Theorem 1.3.*

Input: A set of n independent draws from an unknown distribution.
Output: A labeled vector of probabilities.

1. For each $i \geq n^{1/3}$, for every domain element observed exactly i times in the n samples, assign its empirical probability, i/n.
2. For each $i < n^{1/3}$, assign probability $\frac{i+1}{n-i} \cdot \frac{F_{i+1}}{F_i}$ to each of the F_i elements observed exactly i times, where F_j denotes the number of domain elements observed exactly j times in the n samples.

The proof of Theorem 1.3 will rely on the following intuitive concentration results.

Lemma 1.9. *With probability $1 - n^{-\omega(1)}$, for any distribution, p, we have the following tail bounds:*

- *The contribution to the error due to elements occurring more than $n^{1/3}$ times is small:*

$$\sum_{x:\ freq(x) \geq n^{1/3}} |p(x) - \widehat{p(x)}| \leq \sum_{x:\ freq(x) \geq n^{1/3}} \frac{\sqrt{freq(x)}}{n} \operatorname{polylog} n \leq \tilde{O}(n^{-1/6}),$$

  where $\widehat{p(x)} = freq(x)/n$ denotes the empirical probability of x.
- *For $i \geq 1$, $|F_i - \boldsymbol{E}[F_i]| \leq \sqrt{1 + F_i} \operatorname{polylog} n$, and the mass comprised of elements seen i times, m_i satisfies $|m_i - \boldsymbol{E}[m_i]| \leq \frac{i}{n}\sqrt{1 + \boldsymbol{E}[F_i]} \operatorname{polylog} n$, and the contribution to the error from our approximation of the mean probability $m_i/F_i \approx \frac{i+1}{n-i} \cdot \frac{F_{i+1}}{F_i}$ is bounded as*

$$F_i \left| \frac{m_i}{F_i} - \frac{(i+1)F_{i+1}}{(n-i)F_i} \right| = \left| m_i - \frac{(i+1)F_{i+1}}{n-i} \right| \leq \frac{i}{n}\sqrt{1 + \boldsymbol{E}[F_i]} \operatorname{polylog} n.$$

The proof of the above lemma is left as an exercise. Proving these concentration bounds is complicated by the fact that the quantities in question cannot be easily represented as a sum of independent random variables—even the quantity F_i representing the number of elements observed exactly i times involves dependencies. Hence, instead of using basic Chernoff bounds to analyze this, which only apply to sums of independent random variables, one must instead apply Azuma's inequality—the standard analog for martingales, and analyze the Doob martingale that considers the expectation of the quantities in question as each of the n draws is successively revealed.

We now complete the proof that the Good–Turing Denoising Algorithm of Algorithm 1.2 achieves the guarantees of Theorem 1.3. Specifically, with high probability over the choice of the n independent samples, the error of this algorithm is at most $2 \cdot ErrOpt^* + O(1/n^{1/6})$, where $ErrOpt^*$ is the expected error of the optimal algorithm that receives the description of the true distribution without labels, in addition to the samples. We will prove the slightly stronger statement that with high probability, the algorithm described above achieves an error at most $2 \cdot ErrOpt' + O(1/n^{1/6})$, where $ErrOpt'$ is the expected error of an optimal algorithm that receives a vector of probabilities for each $i \geq 0$, corresponding to the unlabeled vector of probabilities of all elements that occurred exactly i times.

Proof of Theorem 1.3 From Fact 1.6, this even better benchmark algorithm, Opt', simply computes the median of each vector of probabilities and by Fact 1.7, if we instead computed the mean, μ_i of each vector, for $i \geq 1$, and assign μ_i to each element observed i times, we would incur an expected error at most $2ErrOpt'$. The first part of Lemma 1.9 guarantees that the contribution to the error from elements occurring at least $n^{1/3}$ times is bounded by $\tilde{O}(n^{-1/6})$. The second part of Lemma 1.9 shows that the discrepancy between the true and estimated means contribute at most the following quantity to the error: $\sum_{i\in\{1,\ldots,n^{1/3}\}} \frac{i}{n}\sqrt{1+\boldsymbol{E}[F_i]}\,\mathrm{polylog}(n)$. Because of the constraint that $\sum_{i\geq 1} iF_i = n$, this expression is maximized, up to a constant factor, when $\boldsymbol{E}[F_i] \approx n^{1/3}$ for all $i \leq n^{1/3}$, in which case the bound becomes

$$\sum_{i\in\{1,\ldots,n^{1/3}\}} \frac{i}{n}\sqrt{1+\boldsymbol{E}[F_i]}\,\mathrm{polylog}\, n \leq n^{1/3}\frac{n^{1/3}}{n}\sqrt{1+n^{1/3}}\,\mathrm{polylog}\, n = \tilde{O}(n^{-1/6}).$$

□

1.2.3 Estimating the Unseen: Reconstructing a Distribution up to Permutation

In this section we describe an algorithm that accurately approximates the unlabeled vector of probabilities of a distribution, given access to independent samples. This is the main subroutine in the Instance Optimal Learning Algorithm satisfying the guarantees of Theorem 1.2, sketched in Algorithm 1.1. We also briefly discuss some applications of this subroutine beyond its use in instance-optimal learning.

The recovery guarantee for this reconstruction roughly states that, for any distribution, p, given n independent draws, one can accurately recover the portion of the unlabeled vector of true probabilities comprised of probabilities above $c/n\log n$, for a suitable constant c. This is despite the fact that all the empirical probabilities of elements we observe are integer multiples of $1/n$, and for the elements with probability $\ll 1/n$, we cannot hope to learn the labels for most of them, as the vast majority of such elements will not be observed in the n samples. The following theorem formalizes the recovery guarantees.

Theorem 1.10. *Let c denote an absolute constant. For a distribution p, let $p_1 \geq p_2 \geq \ldots$ represent the sorted vector of probabilities assigned to domain elements. For sufficiently large n and any $w \in [1, \log n]$, given n independent draws from p,*

a. One can recover a vector $q = (q_1 \geq q_2 \geq \ldots)$ such that with probability $1-e^{-n^{\Omega(1)}}$

$$\sum_{i:p_i\geq w/(n\log n)} |p_i - q_i| \leq \frac{c}{\sqrt{w}}.$$

b. Letting $cdf_p(v) = \sum_{x:p(x)\leq v} p(x)$ represent that cumulative density function of

p, one can recover a distribution q such that with probability $1 - e^{-n^{\Omega(1)}}$

$$\int_{v=w/(n \log n)}^{1} \frac{1}{v} |cdf_p(v) - cdf_q(v)| \, dv \leq \frac{c}{\sqrt{w}}.$$

In the case that w is a large constant, the above theorem guarantees that one can accurately learn the multi-set of probabilities of the domain elements that occur with probability at least $\Theta(1/n \log n)$. Although many of these elements might not occur in the sample, Theorem 1.10 asserts that one can robustly detect the presence of such elements.

Beyond being interesting in its own right, the ability to reconstruct the unlabeled vector of probabilities as accurately as is guaranteed by Theorem 1.10 has a number of immediate applications for estimating label-invariant properties of the distribution (often referred to as *symmetric* properties). Indeed, any property that is Lipschitz continuous with respect to the distance metrics of the above theorem, can be estimated by evaluating that property value on the recovered distribution, q. Such continuous properties include the expected value of functions of a larger set of independent draws. For example, an easy corollary of Theorem 1.10 is that one can accurately estimate the number of distinct elements that would be observed in a set of $m > n$ independent draws, for m up to $O(n \log n)$:

Corollary 1.11. *Given n samples from an arbitrary distribution p, with probability* $1 - e^{-n^{\Omega(1)}}$ *over the randomness of the samples, one can estimate the expected number of unique elements that would be seen in a set of m samples drawn from p, to within error* $m \cdot c\sqrt{\frac{m}{n \log n}}$ *for some universal constant c.*

From a practical standpoint, this corollary has a number of implication for the many settings where data collection is expensive. In (Zou et al., 2016), for example, this framework was fruitfully used to estimate the number of new, medically relevant mutations that would likely be discovered if larger genetic cohorts were sequenced.

The algorithm for learning the unlabeled vector of probabilities will solve an optimization problem that returns a distribution, q, with the property that if the n samples had been drawn from q, one would expect to see similar statistics to the observed statistics of the actual samples. Specifically, the optimization problem will be a linear program, which returns a distribution with the property that the expected number of elements observed once, twice, etc. in a set of n independent draws will closely match the observed quantities $F_1, F_2, \ldots$. Discrepancies between F_i and the expectation of F_i under the returned distribution, are penalized proportionately to the inverse of the standard deviation, which is approximated as $1/\sqrt{F_i + 1}$. This approximation is reasonable because the variance of F_i is roughly equal to the expectation of F_i, as is the case for Poisson random variables.

The linear program will be described in terms of a fine ϵ-mesh of probability values, $x_1, \ldots, x_\ell \in (0, 1]$ that discretely approximate the potential probability values with which elements of the returned distribution may occur. The variables of

the linear program, $h_1, \ldots, h_\ell$ will be interpreted so that h_i represents the number of domain elements that occur with probability x_i. Because the goal is to return a distribution, the total probability mass is constrained to equal 1, namely $\sum_i h_i x_i = 1$, which is a linear constraint in terms of the variables h_i. Additionally, by linearity of expectation, the expected values of F_j are also linear in the variables h_i, namely $\boldsymbol{E}[F_i] = \sum_j h_j \Pr[Binomial(n, x_j) = i]$, where the expectation is with respect to the distribution represented by the variable $\{h_i\}$. For the time being, we ignore the fact that the linear program is allowed to return non-integral values of h_i—as an additional step, the algorithm could perform a rounding/truncation step to deal with this minor issue.

One final subtlety is that this linear program will only be used for the portion of the distribution corresponding to domain elements that are not seen too many times. For large values of i, one would not actually expect F_i to be concentrated about its expectation. For example, even if there is a domain element that occurs with probability $1/2$, one would not expect $F_{n/2}$ to be too tightly concentrated about its expectation—indeed $F_{n/2}$ will either be 0 or 1 (or 2). Fortunately, for elements that occur often, their empirical probabilities are likely to be accurate. Hence, for elements seen frequently (at least n^α times for some appropriately chosen absolute constant $\alpha > 0$) the algorithm can simply use their empirical probabilities. For the potentially large number of elements each observed few times (at most n^α times), the linear program is used to recover the corresponding portion of the distribution. The fact that the linear program will only be responsible for the small portion of the potential distribution means that the linear program will be small, with only $O(n^\alpha)$ constraints corresponding to enforcing that $\boldsymbol{E}[F_i] \approx F_i$ for $i \leq n^\alpha$. This will ensure that, both in theory and in practice, the linear program could be solved in time sublinear in the number of samples, n. For the purposes of this exposition, however, we omit the minor modifications necessary to achieve sublinear runtime.

The algorithm is presented in terms of two positive constants, B, C, which can be defined arbitrarily provided the following inequalities hold: $0.1 > B > C > \frac{B}{2} > 0$.

Algorithm 1.3. *The Frequency Spectrum Recovery Algorithm for reconstructing the unlabeled vector of probabilities, achieving the guarantees of Theorem 1.10.*

Input: Vector $F_1, F_2, \ldots$ where F_i denotes the number of domain elements observed exactly i times in a set of n samples.
Output: Vector of pairs $(x_1, h_1), \ldots, (x_t, h_t)$.

1. Define the set $X = \{\frac{1}{n^2}, \frac{2}{n^2}, \frac{3}{n^2}, \ldots, \frac{n(n^B+n^C)}{n^2}\}$.
2. For each $x \in X$, define the associated variable h_x, and solve the LP:

$$\text{Minimize} \sum_{i=1}^{n^B} \frac{1}{\sqrt{F_i + 1}} \left| F_i - \sum_{x \in X} h_x \Pr[Binomial(n, x) = i] \right|$$

Subject to:

- $\sum_{x \in X} x \cdot h_x + \sum_{i > n^B + 2n^C}^{n} \frac{i}{n} F_i = 1$ (total prob. mass = 1)
- $\forall x \in X, h_x \geq 0$

3. Return the set of pairs (x_i, h_{x_i}), together with pairs $(i/n, F_i)$ for those domain elements occuring $i > n^B + 2n^C$ times.

The proof of correctness of the above algorithm, establishing Theorem 1.10 is quite involved, and proceeds by directly relating the objective value of the linear program to an appropriate notion of distance between the distribution represented by the returned (x_i, h_i) pairs, and the true vector of probabilities. We refer the reader to the treatment in Valiant and Valiant (2017b) for the details of this analysis.

1.3 Identity Testing

We now turn to a basic distributional hypothesis testing problem: given the description of a distribution p over a discrete support, error tolerance $\epsilon > 0$, and n independent draws from an unknown distribution, q, distinguish the case that $p = q$ versus the case that p and q have total variation distance at least ϵ, with probability of success at least $2/3$ over the randomness of the samples. This success probability of $2/3$ is standard in this literature, and can be exponentially amplified if needed by repeating the test with new samples and returning the majority outcome.

1.3.1 Overview

In contrast to the results of Section 1.2 in which the algorithms presented were instance optimal in terms of the unknown distribution from which the samples were drawn, in this section we will strive for an algorithm that is optimal in terms of the known distribution, p, in a worst-case sense over unknown distributions, q. Since distribution p is known to the algorithm, we will assume, without loss of generality, that it is supported on the positive integers, and will use p_i to denote the probability assigned to element i.

The classic approach to this hypothesis testing problem is via Pearson's chi-squared test. Letting X_i denote the number of times element i appears in the set of n samples, the chi-squared test accepts or rejects according to whether the following quantity exceeds a given threshold: $\sum_i \frac{(X_i - np_i)^2}{p_i}$. For distributions p with large support, this test is far from optimal. For example, if p is a uniform distribution over k elements, for constant ϵ, the chi-squared test requires k samples as compared to the optimal $\sqrt{k}$ samples (see Exercise 1.5). *Can one develop an identity test that is optimal for every distribution, p, and every error parameter, ϵ?*

The answer is yes, and the optimal algorithm is a modification of the chi-squared test. As a bonus, the analysis of this algorithm yields an expression, as a function

of p and ϵ, characterizing the necessary and sufficient amount of data required to perform this hypothesis test. Before summarizing this result, we introduce the following notation that will be used for the rest of this section.

Notation: Given a distribution p, let $p^{-\max}$ refer to the vector of probabilities of the distribution after removing the single highest-probability element; let $p_{-\epsilon}$ refer to the distribution after removing the lowest-probability elements, one-by-one, stopping just before ϵ total probability mass has been removed. We use standard notation for ℓ_p norms, where for real vector, v, and real number a, the ℓ_a norm of v is $\|v\|_a = (\sum_i |v_i|^a)^{1/a}$; however, unusually, instead of the standard ℓ_1 or ℓ_2 norms, the $a = 2/3$ norm is crucial to the analysis. We slightly abuse notation by using p both to refer to the distribution and to its vector of probabilities, $p = (p_1, p_2, \ldots)$.

Theorem 1.12. *Define the function* $f(p, \epsilon) = \max\{\frac{1}{\epsilon}, \frac{\|p_{-\epsilon}^{-\max}\|_{2/3}}{\epsilon^2}\}$. *There exists a tester and constants* $c_1, c_2 > 0$ *such that for any* $\epsilon > 0$ *and any distribution* p, *given samples from any unknown distribution* q,

a. *The tester will distinguish* $q = p$ *from* $\|p - q\|_1 \geq \epsilon$ *with probability* $\geq 2/3$ *when run on a set of at least* $f(p, c_1\,\epsilon)$ *samples drawn from* q; *and*
b. *No tester can accomplish this task with a set of fewer than* $f(p, c_2\,\epsilon)$ *samples.*

1.3.2 Interpretation of the Sample Complexity, $f(p, \epsilon)$

The function $f(p, \epsilon)$ defined in Theorem 1.12 expresses the optimal sample complexity of hypothesis testing distribution p. While the expression may look odd, the fact that it is constant-factor optimal for each and every distribution p means each of the quirks of $f(p, \epsilon) = \max\{\frac{1}{\epsilon}, \frac{\|p_{-\epsilon}^{-\max}\|_{2/3}}{\epsilon^2}\}$ represents a real phenomenon, and this definition is essentially a law of nature.

The $2/3$ norm of p is perhaps the most mysterious part of this expression, though is natural in light of the fact that when p is the uniform distribution on k elements, $||p||_{2/3} = \sqrt{k}$, matching the tight bounds for uniformity testing (Batu et al., 2000; Paninski, 2008). Further, for distributions of support at most k, the $2/3$ norm attains its maximum for the uniform distribution, and the "$-max$", "$-\epsilon$" modifiers can only decrease its value meaning that $f(p, \epsilon) \leq \frac{\sqrt{k}}{\epsilon^2}$ for all such distributions, which is a tight worst-case bound for distributions supported on at most k elements.

The $\frac{1}{\epsilon^2}$ multiplier in $f(p, \epsilon)$ shows up repeatedly in statistics, representing the fact that one needs $\frac{1}{\epsilon^2}$ coin flips to estimate the bias of a coin to accuracy $O(\epsilon)$, because the standard deviation of a sample mean decreases with the square root of the number of samples. The maximum with $\frac{1}{\epsilon}$ reflects the fact that, no matter what distribution we start with, it is impossible to distinguish a discrepancy of ϵ probability mass based on fewer than $\Omega(\frac{1}{\epsilon})$ samples; this term becomes relevant only in the "edge case" when $\|p_{-\epsilon}^{-\max}\|_{2/3} < \epsilon$, which can only happen when the maximum probability element has mass at least $1 - 2\epsilon$.

1.3.3 An Instance Optimal Algorithm

The testing algorithm satisfying Theorem 1.12, makes three crucial modifications, term-by-term, to the quantities computed in Pearson's chi-squared test, $\sum_i (X_i - np_i)^2/p_i$: 1) subtract X_i from the numerator to reduce the variance due to rare elements; 2) modify the denominator from p_i to $p_i^{2/3}$ to reduce the penalty for discrepancies in small probabilities; and 3) examine the smallest probability domain elements only in aggregate, while also ignoring the single largest domain element. Before formally stating the algorithm, we briefly motivate the first two of these modifications, which result in the expression $\sum_i \frac{(X_i - np_i)^2 - X_i}{p_i^{2/3}}$, which we refer to as the (instance optimal) test statistic.

The numerator of the ith term, $(X_i - np_i)^2 - X_i$, has two useful properties. First, it gives an *almost* unbiased estimate of $(p_i - q_i)^2$, after scaling: since $\boldsymbol{E}[X_i] = nq_i$ and $\boldsymbol{E}[X_i^2] = n^2 q_i^2 + nq_i(1 - q_i)$, we have

$$\begin{aligned}\boldsymbol{E}[(X_i - np_i)^2 - X_i] &= \boldsymbol{E}[X_i^2] - 2np_i \boldsymbol{E}[X_i] + (np_i)^2 - \boldsymbol{E}[X_i] = n^2(q_i - p_i)^2 - nq_i^2 \\ &\approx n^2(q_i - p_i)^2.\end{aligned}$$

Second, domain elements i for which $np_i, nq_i \ll 1$ will contribute very little variance to this expression. For such elements, $(X_i - np_i)^2 - X_i \approx X_i^2 - X_i$, which evaluates to 0 when $X_i = 0$ *and* when $X_i = 1$. Phrased differently, this expression is essentially agnostic to whether a rare element occurs zero times, versus once. The standard chi-squared statistic, by comparison, incurs a significant variance from such elements.

There is not an entirely clean motivation for scaling the ith term by $1/p_i^{2/3}$. Scaling by $1/p_i$, as in the chi-squared test, compensates for the fact that the expectation of the numerator is $(p_i - q_i)^2$, instead of the desired $|p_i - q_i|$. For example, if p_i and q_i differed by a constant factor, then the expected contribution after this scaling would also be proportional to $|p_i - q_i|$. The intuition for scaling by $1/p_i^\alpha$ for some $\alpha < 1$ is that even when $p = q$, we expect proportionately larger deviations between np_i and X_i for smaller values of p_i, and hence we must penalize such deviations less.

The lower bound construction, showing the optimality of this testing algorithm, yields a different perspective on the $1/p_i^{2/3}$ scaling of each term. Roughly speaking, for any distribution p, the most difficult instance of this hypothesis test is distinguishing whether $p = q$, versus a distribution where each element of q has a randomly perturbed probability $q_i = p_i \pm \delta_i$ for some choice of perturbations δ_i that sum to ϵ. From a lower bound standpoint, the question is how to allocate the δ deviation to the different elements. Setting $\delta_i = \epsilon \cdot p_i$, which would be the proportionate allocation, is clearly suboptimal, since the larger q_i is, the more accurate a multiplicative estimate of q_i will be. This motivates setting the magnitude of $\delta_i = |q_i - p_i|$ to be proportional to p_i^α for some $\alpha < 1$. As it turns out, setting $\delta_i = p_i^{2/3}$ is optimal, matching the $1/p_i^{2/3}$ scaling in the statistic $\sum_i \frac{(X_i - np_i)^2 - X_i}{p_i^{2/3}}$.

We conclude by formally describing the testing algorithm in Algorithm 1.4, and saying a word about the proof of Theorem 1.12.

Algorithm 1.4. *The 2/3-Norm Testing Algorithm, which optimally tests whether* $p = q$ *versus* $\|p - q\|_1 \geq \epsilon$.

Input: Distribution $p = (p_1, p_2, \ldots)$, parameter $\epsilon > 0$ and a vector $X_1, X_2, \ldots$ where X_i denotes the number of times domain element i occurs in a set of n samples from an unknown distribution q.
Output: Either "$\|p - q\|_1 \geq \epsilon$" or "$p = q$".

0. Assume wlog that the domain elements of p are sorted in non-increasing order of probability. Define $s = \min\{i : \sum_{j>i} p_j \leq \epsilon/8\}$, and let $S = \{s+1, s+2, \ldots\}$ (the "small" elements) and $M = \{2, \ldots, s\}$ (the "medium" elements).
1. Threshold the test statistic: if $\sum_{i \in M} \frac{(X_i - np_i)^2 - X_i}{p_i^{2/3}} > 4n\|p_M\|_{2/3}^{1/3}$ output "$\|p - q\|_1 \geq \epsilon$".
2. If $\sum_{i \in S} X_i > \frac{3}{16}\epsilon n$ output "$\|p - q\|_1 \geq \epsilon$".
3. Otherwise, output "$p = q$".

The proof of 1.12(a), establishing the performance guarantees of the above testing algorithm, is conceptually simple. The core of the analysis is to apply Chebyshev's inequality to the expressions computed in Steps 1 and 2 of the algorithm to show that—with the claimed probability—we accept true hypotheses and reject hypotheses that are far from true. (See Exercise 1.6 for analysis of the complementary roles of these two tests in the algorithm.) Chebyshev's inequality states that a random variable will be more than c standard deviations from its mean with probability at most $\frac{1}{c^2}$. The brunt of the algorithm analysis consists of showing that the expectations of the expressions computed by the algorithm differ significantly when $p = q$ versus when $\|p - q\|_1 \geq \epsilon$, and that this difference is large in comparison to the standard deviation of these quantities. Unfortunately, this simple approach reduces (after some straightforward but tedious algebra that we omit for clarity) to the problem of proving that an extremely messy inequality holds for all distributions p, and all discrepancies from the hypothesis $\Delta = (p_1 - q_1, p_2 - q_2, \ldots,)$:

$$\sum_{i \in M} \left(\frac{p_i^{2/3} ||\Delta_M||_1^4}{||p_M||_{2/3}^2} + 2\frac{\Delta_i ||\Delta_M||_1^3}{||p_M||_{2/3}^{4/3}} + \frac{p_i^{-2/3}\Delta_i^2 ||\Delta_M||_1^2}{||p_M||_{2/3}^{2/3}} \right. \tag{1.1}$$
$$\left. + 2\frac{p_i^{-1/3}\Delta_i^2 ||\Delta_M||_1^2}{||p_M||_{2/3}} + 2\frac{p_i^{-1}\Delta_i^3 ||\Delta_M||_1}{||p_M||_{2/3}^{1/3}} \right) \leq 8 \left(\sum_{i \in M} \Delta_i^2 p_i^{-2/3} \right)^2$$

In Section 1.4 below, we describe a way of *automating* the proofs of such inequalities: this yields both a complete characterization of when such inequalities

are true, along with a polynomial time algorithm that either produces a proof if the inequality is true, or a refutation if the inequality is not true.

1.4 Digression: An Automatic Inequality Prover

Given a sequence of triples, (a_i, b_i, c_i), is it true that for all positive vectors $x = (x_1, \ldots), y = (y_1, \ldots)$ the following inequality holds?

$$\prod_{i=1}^{r} \left(\sum_j x_j^{a_i} y_j^{b_i} \right)^{c_i} \geq 1. \tag{1.2}$$

Several familiar inequalities, including Cauchy-Schwarz, Hölder, and the monotonicity of ℓ_p norms, can be expressed in this form, as illustrated below. Additionally, the proof of the inequality of Equation 1.1 corresponds to proving five inequalities of the above form—each inequality bounding one of the terms on the left hand side by the right hand side.

$$\left(\sum_j x_j^2 \right)^{1/2} \left(\sum_j y_j^2 \right)^{1/2} \left(\sum_j x_j y_j \right)^{-1} \geq 1 \qquad \text{(Cauchy-Schwarz)}$$

$$\left(\sum_j x_j^{1/\lambda} \right)^{\lambda} \left(\sum_j y_j^{1/(1-\lambda)} \right)^{1-\lambda} \left(\sum_j x_j y_j \right)^{-1} \geq 1 \qquad \text{(Hölder)}$$

$$\left(\sum_j x_j^{1/\lambda} \right)^{-\lambda} \left(\sum_j x_j \right) \geq 1 \qquad (\ell_p \text{ monotonicity})$$

In this section, we show that an inequality of the form of Equation 1.2 is true, if and only if it is expressible as the product of positive powers of Hölder, and ℓ_p monotonicity inequalities. Furthermore, there is an efficient algorithm for automatically proving or disproving such an inequality: given the triples, (a_i, b_i, c_i), the algorithm either produces a derivation of the inequality, or produces a counter-example pair of sequences x, y which falsify the inequality.

Theorem 1.13. *For a sequence of triples $(a, b, c)_i = (a_1, b_1, c_1), \ldots (a_r, b_r, c_r)$, the inequality $\prod_{i=1}^{r} \left(\sum_j x_j^{a_i} y_j^{b_i} \right)^{c_i} \geq 1$ holds for all finite sequences of positive numbers $(x)_j, (y)_j$ if and only if it can be expressed as a finite product of positive powers of Hölder inequalities of the form $\left(\sum_j x_j^{a'} y_j^{b'} \right)^{\lambda} \cdot \left(\sum_j x_j^{a''} y_j^{b''} \right)^{1-\lambda} \geq \sum_j x_j^{\lambda a' + (1-\lambda) a''} y_j^{\lambda b' + (1-\lambda) b''}$, and ℓ_p monotonicity inequalities of the form $\left(\sum_j x_j^{a} y_j^{b} \right)^{\lambda} \leq \sum_j x_j^{\lambda a} y_j^{\lambda b}$, where $\lambda \in [0, 1]$. Such a derivation can be found in polynomial time via*

linear programming whenever the inequality is true; and a compact representation of a refutation can be found whenever the inequality is false.

Example 1.14. Consider for some $\epsilon \geq 0$ the single-sequence inequality

$$\Big(\sum_j x_j^{-2}\Big)^{-1}\Big(\sum_j x_j^{-1}\Big)^{3}\Big(\sum_j x_j^{0}\Big)^{-2-\epsilon}\Big(\sum_j x_j^{1}\Big)^{3}\Big(\sum_j x_j^{2}\Big)^{-1} \geq 1,$$

which can be expressed in the form of Equation 1.2 via the triples $(a_i, b_i, c_i) = (-2, 0, -1), (-1, 0, 3), (0, 0, -2-\epsilon), (1, 0, 3), (2, 0, -1)$. This inequality is true for $\epsilon = 0$ but false for any positive ϵ. However, the shortest counterexample sequences have length that grows as $exp(\frac{1}{\epsilon})$ as ϵ approaches 0. Counterexamples are thus hard to write down, though easy to express—for example, letting $n = 64^{1/\epsilon}$, the sequence x of length $2 + n$ consisting of $n, \frac{1}{n}$, followed by n ones violates the inequality.

1.4.1 Proving inequalities without math: a peg game

Theorem 1.13 argues that there is a linear-programming based algorithm for efficiently proving or refuting inequalities of the specified form. The intuition underlying the proof of Theorem 1.13, however, can be used to formulate the task of proving such an inequality as a simple and intuitive "peg game" played on a 2-d board. This peg game interpretation allows one to use basic geometric intuitions to easily derive a proof of many of these inequalities, using only a little bit of pencil and paper!

We describe this peg game in the concrete setting of proving the following inequality (which corresponds to the 4^{th} component of Equation 1.1 from Section 1.3 where Δ has been replaced by x and p has been replaced by y):

$$\left(\sum_j x_j^2 y_j^{-2/3}\right)^{2}\left(\sum_j x_j^2 y_j^{-1/3}\right)^{-1}\left(\sum_j x_j\right)^{-2}\left(\sum_j y_j^{2/3}\right)^{3/2} \geq 1, \qquad (1.3)$$

Expressing this inequality in the form of Theorem 1.13, we have the triples $(a_i, b_i, c_i) = (2, -\frac{2}{3}, 2), (2, -\frac{1}{3}, -1), (1, 0, -2), (0, \frac{2}{3}, \frac{3}{2})$. The peg game—as illustrated in Figure 1.5—begins by representing each triple (a_i, b_i, c_i) as the number c_i written at location (a_i, b_i) in the plane. At any moment, the game board consists of some numbers written on the plane (with the convention that every point without a number is interpreted as having a 0), and you "win" if you can remove all the numbers from the board via a combination of "moves" of the following two types:

1. (Hölder) Any two positive numbers can be moved to the weighted mean of their locations. (For example, we can subtract 1 from one location in the plane, subtract 3 from a second location in the plane, and add 4 to a point $\frac{3}{4}$ of the way from the first location to the second location.)

2. (ℓ_p monotonicity) Any negative number can be moved towards the origin by a factor $\lambda \in (0,1)$ and scaled by $\frac{1}{\lambda}$. (For example, we can add 1 to one location in the plane, and subtract 2 from a location halfway to the origin.)

The rules of the game allow just these two types of moves: you can push positive numbers together, and push negative numbers towards the origin (scaling them). Theorem 1.13 translates into the claim that this peg game can be won if, and only if, the corresponding inequality is true; additionally, a small linear program can either produce a winning combination of moves, or present a certificate that the game is unwinnable. Nevertheless, our geometric intuition is quite good at solving these types of puzzles, even for intricate counterintuitive inequalities like the current example. (Try it!)

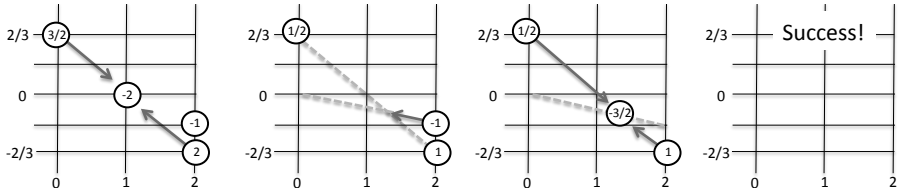

Figure 1.5 Depiction of a successful sequence of "moves" in the game corresponding to the inequality $\left(\sum_j x_j^2 y_j^{-2/3}\right)^2 \left(\sum_j x_j^2 y_j^{-1/3}\right)^{-1} \left(\sum_j x_j\right)^{-2} \left(\sum_j y_j^{2/3}\right)^{3/2} \geq 1$, showing that the inequality is true. The first diagram illustrates the initial configuration of positive and negative weights, together with the "Hölder-type move" that takes one unit of weight from each of the points at $(0, 2/3)$ and $(2, -2/3)$ and moves it to the point $(1, 0)$, canceling out the weight of -2 that was initially at $(1, 0)$. The second diagram illustrates the resulting configuration, together with the "ℓ_p monotonicity move" that moves the -1 weight at location $(2, -1/3)$ towards the origin by a factor of $2/3$ while scaling it by a factor of $3/2$, resulting in a point at $(4/3, -2/9)$ with weight $-3/2$, which is now collinear with the remaining two points. The third diagram illustrates the final "Hölder-type move" that moves the two points with positive weight to their weighted average, zeroing out all weights.

The intuition behind one winning sequence for the game corresponding to Equation 1.3, illustrated in Figure 1.5, is to first realize that three of the points lie on a line, with the "-2" halfway between the "$\frac{3}{2}$" and the "2". Thus we take 1 unit from each of the endpoints and cancel out the "-2" via a "Hölder" move. Now, no three points are collinear, so we need to move one point onto the line formed by the other two: "-1", being negative, can be moved towards the origin, so we move it until it crosses the line formed by the two remaining numbers. This moves it $\frac{1}{3}$ of the way to the origin, thus increasing it from "-1" to "$-\frac{3}{2}$"; amazingly, this number, at position $\frac{2}{3}(2, -\frac{1}{3}) = (\frac{4}{3}, -\frac{2}{9})$ is now $\frac{2}{3}$ of the way from the remaining "$\frac{1}{2}$" at $(0, \frac{2}{3})$ to the number "1" at $(2, -\frac{2}{3})$, meaning that we can remove the final three numbers from the board in a single move, winning the game. We thus made three moves total, two of the Hölder type, one of the ℓ_p monotonicity type. Reexpressing these

moves as inequalities yields the desired derivation of our inequality (Equation 1.3) as a product of powers of Hölder and ℓ_p monotonicity inequalities, explicitly, as the product of the following three inequalities, which are respectively 1) the square of a Cauchy-Schwarz inequality, 2) the 3/2 power of an ℓ_p monotonicity inequality for $\lambda = 2/3$, and 3) the 3/2 power of a Hölder inequalty for $\lambda = 2/3$:

$$\begin{aligned}\left(\sum_j x_j^2 y_j^{-2/3}\right)\left(\sum_j x_j^0 y_j^{2/3}\right)\left(\sum_j x_j^1 y_j^0\right)^{-2} &\geq 1\\ \left(\sum_j x_j^{4/3} y_j^{-2/9}\right)^{3/2}\left(\sum_j x_j^2 y_j^{-1/3}\right)^{-1} &\geq 1\\ \left(\sum_j x_j^2 y_j^{-2/3}\right)\left(\sum_j x_j^0 y_j^{2/3}\right)^{1/2}\left(\sum_j x_j^{4/3} y_j^{-2/9}\right)^{-3/2} &\geq 1.\end{aligned}$$

1.5 Beyond Worst Case Analysis for Other Testing Problems

There are a wide variety of testing and learning problems that can be considered from perspectives other than worst case analysis, beyond the two settings highlighted in this chapter. In many cases, a significant part of the challenge is defining a reasonable benchmark or notion of optimality that yields clean, conceptually appealing results and practically meaningful algorithms.

To briefly describe one example, Section 1.3 considers the question of distinguishing whether two distributions, p and q, are equal versus have significant distance, given a description of p and samples drawn from q. The analogous question can also be asked where both distributions p and q are unknown, and one wishes to deduce if $p = q$ versus $\|p - q\|_1 \geq \epsilon$ given samples from both distributions. If p and q are supported on at most n elements, this hypothesis can be tested using $O\left(\max(n^{2/3}/\epsilon^{4/3}, n^{1/2}\epsilon^2)\right)$, which is optimal *in the worst case* (Batu et al., 2000; Chan et al., 2014).

Going beyond worst-case analysis, the works Acharya et al. (2011, 2012) apply the perspective of *competitive analysis* to this question. Instead of bounding the sample size required for this task in terms of the support size of the distributions, this work bounds the sample size as a (super-linear) function of the sample size that would be required if distributions p and q were known to the algorithm, and the algorithm needed to distinguish whether two sets of samples were drawn from the pair p, q versus both drawn from a single distribution.

The work Lam-Weil et al. (2019) takes a quite different approach towards this problem of identity testing with two unknown distribution. They develop an algorithm which, for every p, q, uses as few samples as would be necessary even if one "approximately" knows distribution q. Specifically, given a vector of probabilities,

π, they ask how difficult it is to distinguish $p = q$ versus $\|p - q\|_1 \geq \epsilon$ where distribution q is obtained via the random process of sampling the probabilities of its elements uniformly at random from the multiset π, and p is a worst-case distribution with distance ϵ from q. Here, the goal is to get an optimal sample complexity as a function of π, achieved via an algorithm that does not require knowledge of π.

1.6 References and Bibliographic Notes

Section 1.2 is based on results from Valiant and Valiant (2016), and Sections 1.3 and 1.4 are based on Valiant and Valiant (2017a). For the problem of instance optimal learning discussed in Section 1.2, the work of Orlitsky and Suresh (2015) which appeared contemporaneously with Valiant and Valiant (2016), considered the problem of learning with respect to KL-divergence, instead of total variation distance (L1 distance). In that setting, they showed a variant of the Good–Turing Denoising Algorithm (Algorithm 1.2) is instance optimal for learning with respect to KL-divergence in an analogous sense to the results discussed in Section 1.2.

For additional intuition on how the 2/3-norm arises in the sample complexity of instance optimal testing (Theorem 1.12), we refer the reader to Diakonikolas and Kane (2016), who obtained a similar expression with extra polylogarithmic factors, via a general framework for reducing such hypothesis testing questions to the easier task of performing analogous tests in terms of ℓ_2 distance.

For a general introduction to modern questions and perspectives on distributional property testing and estimation, we refer the reader to the survey Canonne (2015), or the slightly older survey Rubinfeld and Shapira (2011). These surveys also provide some historical context for how these fundamental statistical questions came to be studied by the theoretical computer science community, first in the context of testing graph expansion—essentially the question of identity testing with respect to the uniform distribution (Goldreich and Ron, 2011)—and subsequently abstracted and generalized to hypothesis tests and estimates of ℓ_1 and ℓ_2 norms between distributions (Batu et al., 2000).

References

Acharya, J., Das, H., Jafarpour, A., Orlitsky, A., and Pan, S. 2011. Competitive closeness testing. In: *Conference on Learning Theory (COLT)*.

Acharya, J., Das, H., Jafarpour, A., Orlitsky, A., and Pan, S. 2012. Competitive classification and closeness testing. *Proc. 25th Conference on Learning Theory (COLT)*, **23**, 22.1–22.18.

Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., and White, P. 2000. Testing that distributions are close. In: *IEEE Symposium on Foundations of Computer Science (FOCS)*.

Canonne, Clément L. 2015. A survey on distribution testing: Your data is big. but is it blue? In: *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 22.

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

Diakonikolas, Ilias, and Kane, Daniel M. 2016. A new approach for testing properties of discrete distributions. Pages 685–694 of: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE.

Goldreich, Oded, and Ron, Dana. 2011. On testing expansion in bounded-degree graphs. Pages 68–75 of: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer.

Lam-Weil, Joseph, Carpentier, Alexandra, and Sriperumbudur, Bharath K. 2019. Local minimax rates for closeness testing of discrete distributions. *arXiv preprint arXiv:1902.01219*.

Orlitsky, Alon, and Suresh, Ananda Theertha. 2015. Competitive Distribution Estimation: Why is Good-Turing Good. Pages 2143–2151 of: Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds), *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc.

Paninski, L. 2008. A coincidence-based test for uniformity given very sparsely-sampled discrete data. *IEEE Transactions on Information Theory*, **54**, 4750–4755.

Rubinfeld, Ronitt, and Shapira, Asaf. 2011. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, **25**(4), 1562–1588.

Valiant, Gregory, and Valiant, Paul. 2016. Instance optimal learning of discrete distributions. Pages 142–155 of: *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*. STOC '16. New York, NY, USA: ACM.

Valiant, Gregory, and Valiant, Paul. 2017a. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, **46**(1), 429–455.

Valiant, Gregory, and Valiant, Paul. 2017b. Estimating the Unseen: Improved Estimators for Entropy and Other Properties. *J. ACM*, **64**(6), 37:1–37:41.

Zou, James, Valiant, Gregory, Valiant, Paul, Karczewski, Konrad, Chan, Siu On, Samocha, Kaitlin, Lek, Monkol, Sunyaev, Shamil, Daly, Mark, and MacArthur, Daniel G. 2016. Quantifying unobserved protein-coding variants in human populations provides a roadmap for large-scale sequencing projects. *Nature communications*, **7**, 13293.

Exercises

1.1 Prove Fact 1.6, that for any multiset of real numbers $S = \{s_1, \ldots, s_m\}$, the *median* minimizes the sum of the absolute distances to elements of S:

$$\sum_{i=1}^{m} |s_i - median(S)| = \inf_{x \in \mathbb{R}} \sum_{i=1}^{m} |s_i - x|.$$

1.2 Prove Fact 1.7, that for any multiset of real numbers $S = \{s_1, \ldots, s_m\}$, the sum of the absolute differences between the *mean* and elements of S is at most a factor of two larger than the sum of distances to the median:

$$\sum_{i=1}^{m} |s_i - mean(S)| \leq 2 \cdot \sum_{i=1}^{m} |s_i - median(S)| = 2 \cdot \inf_{x \in \mathbb{R}} \sum_{i=1}^{m} |s_i - x|.$$

1.3 Given n independent draws from a distribution with discrete support, for integers $i \geq 1$, let F_i represent the number of domain elements that each appear exactly i times in the samples. Prove that F_i is tightly concentrated about its mean, namely for any $c > 0$, $\Pr[|F_i - \boldsymbol{E}[F_i]| \geq c\sqrt{n}] \leq O(\exp(-\Omega(c^2)))$. (**Hint:** Letting x_i denote the ith indepedent draw, consider the Doob martingale: $X_0 = \boldsymbol{E}[F_i], X_1 = \boldsymbol{E}[F_i|x_1], X_2 = \boldsymbol{E}[F_i|x_1, x_2], \ldots, X_n = \boldsymbol{E}[F_i|x_1, \ldots, x_n] = F_i$, and apply Azuma's martingale concentration inequality.)

1.4 Show that the concentration bound of the previous exercise can be improved if $\boldsymbol{E}[F_i] \ll n$: show that $\Pr[|F_i - \boldsymbol{E}[F_i]| \geq c\sqrt{1 + \boldsymbol{E}[F_i]}] = O(\exp(-\Omega(c^2)))$.

1.5 Let $p = (1/2, \frac{1}{2k}, \frac{1}{2k}, \ldots, \frac{1}{2k})$ denote the distribution that puts mass $1/2$ on element 1, and distributes the remaining mass among elements $2, \ldots, k+1$; let $q = (1/2, \frac{1}{k}, \ldots, \frac{1}{k})$ denote an analogous distribution that distributes the remaining mass among $2, \ldots, k/2+1$. Consider using the chi-squared statistic $\sum_i (X_i - np_i)/p_i$ to distinguish the case where n samples were drawn from p versus n samples were drawn from q. Prove that this distinguisher would require $n = \Omega(k)$ samples to have success probability at least $2/3$.

1.6 This exercise motivates the two steps of the algorithm of Algorithm 1.4, with Step 1 detecting discrepancies in "medium" probability elements, and Step 2 detecting if the "small" probability elements have too much total probability mass. Recall that the set S of small elements is constructed so that $\sum_{i \in S} p_i \leq \frac{\epsilon}{8}$, and the set M consists of the remaining elements, with the exception of p_{max}. Prove that if $\|p - q\|_1 \geq \epsilon$ then at least one of the following must hold:

- $\sum_{i \in M} |p_i - q_i||_1 \geq \frac{\epsilon}{8}$ (which will likely trigger Step 1 of the algorithm), or
- $\sum_{i \in S} q_i \geq \frac{\epsilon}{4}$ (which will likely trigger Step 2 of the algorithm).

1.7 Show the monotonicity of ℓ_p norms: for a vector x and $\lambda \in (0, 1)$, $||x||_1 \leq ||x||_\lambda$.

1.8 Win the "peg game" of Figure 1.5 in a *different* way, where the first move is different from zeroing out the -2 at location $(1, 0)$. Express your winning strategy as a combination of Hölder and ℓ_p monotonicity inequalities.

1.9 Prove the inequality of Example 1.14 for $\epsilon = 0$—or more generally, for any $\epsilon \leq 0$—via the "peg game" techniques of Section 1.4.1.