Uncertainty about Uncertainty: Optimal Adaptive Algorithms for Estimating Mixtures of Unknown Coins[*]

Jasper C.H. Lee
Brown University
jasperchlee@brown.edu

Paul Valiant
IAS and Purdue University
pvaliant@gmail.com

Abstract

Given a mixture between two populations of coins, "positive" coins that each have—unknown and potentially different—bias $\geq \frac{1}{2} + \Delta$ and "negative" coins with bias $\leq \frac{1}{2} - \Delta$, we consider the task of estimating the fraction ρ of positive coins to within additive error ϵ. We achieve an upper and lower bound of $\Theta(\frac{\rho}{\epsilon^2\Delta^2} \log \frac{1}{\delta})$ samples for a $1-\delta$ probability of success, where crucially, our lower bound applies to all *fully-adaptive* algorithms. Thus, our sample complexity bounds have tight dependence for every relevant problem parameter. A crucial component of our lower bound proof is a decomposition lemma (Lemma 5.2) showing how to assemble partially-adaptive bounds into a fully-adaptive bound, which may be of independent interest: though we invoke it for the special case of Bernoulli random variables (coins), it applies to general distributions. We present simulation results to demonstrate the practical efficacy of our approach for realistic problem parameters for crowdsourcing applications, focusing on the "rare events" regime where ρ is small. The fine-grained adaptive flavor of both our algorithm and lower bound contrasts with much previous work in distributional testing and learning.

1 Introduction

We consider a natural statistical estimation task, motivated by a practical setting, with an intriguing adaptive flavor. We provide a new adaptive algorithm and a matching fully adaptive lower bound, tight up to multiplicative constants.

In our problem setting, there is a universe of coins of two types: positive coins each have a (potentially different) probability of heads that lies in the interval $[\frac{1}{2} + \Delta, 1]$, while negative coins lie in the interval $[0, \frac{1}{2} - \Delta]$, where $\Delta \in (0, \frac{1}{2}]$ parameterizes the "quality" of the coins. Our only access to the coins is by choosing a coin and then flipping it, without access to the true biases of the coins. An algorithm in this setting may employ arbitrary adaptivity—for example, flipping three different coins in sequence and then flipping the first coin 5 more times if and only if the results of the first 3 flips were heads, tails, heads. The challenge is to estimate the *fraction* ρ of coins that are of positive type, to within a given additive error ϵ, using as few coin flips (samples) as possible. We assume because of the symmetry of the problem (between positive and negative coins) that $\rho \leq \frac{1}{2}$.

This model arose from a collaboration with colleagues in data science and database systems, about harnessing paid crowdsourced workers to estimate the "quality" of a database. Our model is a direct theoretical analog of the following practical problem, where sample complexity linearly translates into the amount of money that must be paid to workers, and thus even multiplicative factors crucially affect the usefulness of an algorithm. Given a set of data and a predicate on the data, the task is to estimate what fraction of the data satisfies the predicate—for example, estimating the proportion of records in a large database that contain erroneous data. After automated tools have labeled whatever portion of the data they are capable of dealing with, the remaining data must be processed via *crowdsourcing*, an emerging setting that potentially offers sophisticated capabilities but at the cost of unreliability. Namely, for each data item, one may ask many human users/workers online whether they think the item satisfies the predicate, with the caveat that the answers returned could be noisy. In the case that the workers have no ability to distinguish the predicate, we cannot hope to succeed; however, if the histograms of detection probabilities for positive versus negative data

[*]We thank Tim Kraska and Yeounoh Chung for bringing these problems to our attention in the data analytics setting, and for contributing to the simulation results in this work. We thank an anonymous reviewer for asking about the δ dependence in lower bounds, which led to the current tight results. This work is partially supported by NSF award IIS-1562657. In addition, Paul Valiant is partially supported by NSF award DMS-1926686, and indirectly supported by NSF award CCF-1900460.

have a gap between them (the gap is 2Δ in the model above), then the challenge is to estimate ρ as accurately as possible, from a limited budget of queries to workers [15].

A key feature that makes this estimation problem distinct from many others studied in the literature is the richness of adaptivity available to the algorithm. Achieving a tight lower bound in this setting requires considering and bounding all possible uses of adaptivity available to an algorithm; and achieving an optimal algorithm requires choosing the appropriate adaptive information flow between different parts of the algorithm. Much of the previous work in the area of statistical estimation is focused on non-adaptive algorithms and lower bounds; however see [7], and in particular, Sections 4.1 and 4.2 of that work, for a survey of several distribution testing models that allow for adaptivity. In our setting there are two distinct kinds of adaptivity that an algorithm can leverage: 1) single-coin adaptivity, deciding how many times a particular coin should be flipped—a per-coin stopping rule—in terms of the results of its previous flips, and 2) cross-coin adaptivity, deciding which coin to flip next in terms of the results of previous flips across *all* coins. Our final optimal algorithm (Section 3) leverages both kinds of adaptivity. In our tight lower bound analysis (Section 5), we overcome the technical obstacles presented by the richness of adaptivity by giving a reduction (Section 5.1) from fully-adaptive algorithms that leverage both kinds of adaptivity to single-coin adaptive algorithms that process each coin independently, valid for our specific lower bound instance. We discuss the approaches and challenges of our lower bound in more detail in Section 1.1.2.

The main *algorithmic* challenge in this problem is what we call "uncertainty about uncertainty": we make no assumptions about the quality of the coins beyond the existence of a gap 2Δ between biases of the coins of different types (centered at $\frac{1}{2}$). If we relaxed the problem, and assumed (perhaps unrealistically) that we know 1) the conditional distribution of biases of positive coins, and 2) the same for negative coins, and 3) an initial estimate of the mixture parameter ρ between the two distributions, then this allows us to use mathematical programming techniques to construct an estimation algorithm with sample complexity that is optimal *by construction* up to a multiplicative constant. We show this construction in the full version of this paper on arXiv [22]. In our original setting, however, our algorithm must return estimates with small bias, and be sample-efficient at the same time, regardless of the bias of the coins, whether they are all deterministic, or all maximally noisy as allowed by the Δ parameter, or some quality in between. While intuitively the hardest settings to distinguish information theoretically involve coins with biases as close to each other as possible (and indeed our lower bound relies on mixtures of only $\frac{1}{2} \pm \Delta$ coins), settings with biases near but not equal to $\frac{1}{2} \pm \Delta$ introduce "uncertainty about uncertainty" challenges. The two kinds of adaptivity available to the algorithm allow us to meet these challenges by trading off, optimally, between 1) investigating a single coin to reduce uncertainty about its bias, and 2) apportioning resources between different coins to reduce uncertainty about the ground truth fraction ρ, which is the objective of the problem.

1.1 Our Approaches and Results

To motivate the new algorithms of this paper, we start by describing the straightforward analysis of perhaps the most natural approach to the problem, which is non-adaptive, based on subsampling.

Example 1.1. *Recall that it takes $\Omega(\frac{1}{\Delta^2})$ samples to distinguish a coin of bias $\frac{1}{2} - \Delta$ from a coin of bias $\frac{1}{2} + \Delta$. We can therefore imagine an algorithm that chooses a random subset of the coins, and flips each coin $\Omega(\frac{1}{\Delta^2})$ many times. Asking for $\Theta(\frac{1}{\Delta^2} \log \frac{1}{\epsilon})$ flips from each coin guarantees that all but ϵ fraction of the coins in the subset will be accurately classified. Given an accurate classification of m randomly chosen coins, we use the fraction of these that appear positive as an estimate on the overall mixture parameter ρ. Estimating ρ to within error ϵ requires $m = O(\frac{\rho}{\epsilon^2})$ randomly chosen coins. Overall, taking $\Theta(\frac{1}{\Delta^2} \log \frac{1}{\epsilon})$ samples from each of $m = \Theta(\frac{\rho}{\epsilon^2})$ coins uses $\Theta(\frac{\rho}{\epsilon^2 \Delta^2} \log \frac{1}{\epsilon})$ samples.*

As we will see, the above straightforward algorithm is potentially wasteful in samples by up to a $\log \frac{1}{\epsilon}$ factor, since it makes $\Theta(\frac{1}{\Delta^2} \log \frac{1}{\epsilon})$ flips for every single coin, yet—since $\Omega(\frac{1}{\Delta^2})$ samples suffices to label a coin with constant accuracy—each sample beyond the first $\Theta(\frac{1}{\Delta^2})$ samples from a single coin gives increasing certainty yet diminishing information-per-coin. If we can save on this $\log \frac{1}{\epsilon}$ factor without sacrificing impractical constants, then our approach leads to significant practical savings in samples, and thus monetary cost—in regimes, such as crowdsourcing, where gathering data is by far the most expensive part of the estimation process.

1.1.1 Algorithmic Construction

We give two algorithmic constructions. Algorithm 2, which we call the Triangular Walk algorithm, is single-coin adaptive, and is theoretically almost-tight in sample complexity. Second, Algorithm 5 has the optimal sample complexity, by combining the Triangular Walk algorithm with a new (and surprisingly) non-adaptive component (Algorithm 3).

The Triangular Walk algorithm (Algorithm 2) is designed for the specific *practical* parameter regime where ρ is small: in our earlier crowdsourcing example, practitioners typically preprocess data items by using automated techniques and heuristics to classify a majority of the items, before leaving to crowdsourced workers a small number of items that cannot be automatically classified. These automated filtering techniques usually flag significantly more "negative" items than "positive" items as "unclassifiable automatically", resulting in a small fraction ρ of positive items among the ones selected for crowdsourced human classification. The intuition behind our approach, then, is to try to abandon sampling (frequent) negative coins as soon as possible, after $\Theta(\frac{1}{\Delta^2})$ samples, while being willing to investigate (infrequent) positive coins up to depth $\Theta(\frac{1}{\Delta^2}\log\frac{1}{\epsilon})$. Thus we disproportionately bias our investment of resources towards the rare and valuable regime. Using techniques from random walk theory, we design a linear estimator based on this behavior (Algorithm 1), whose expectation across many coins yields a robust estimator, Algorithm 2, as shown in Theorem 1.1 (restated and proved in Section 2).

THEOREM 1.1. *Given coins where a ρ fraction of the coins have bias $\geq \frac{1}{2}+\Delta$, and $1-\rho$ fraction have bias $\leq \frac{1}{2}-\Delta$, then running Algorithm 2 on $t = \Theta(\frac{\rho}{\epsilon^2}\log\frac{1}{\delta})$ randomly chosen coins will estimate ρ to within an additive error of $\pm\epsilon$, with probability at least $1-\delta$, with an expected sample complexity of $O(\frac{\rho}{\epsilon^2\Delta^2}(1+\rho\log\frac{1}{\epsilon})\log\frac{1}{\delta})$.*

The analysis of Algorithm 2 uses only standard concentration inequalities, and thus the big-O notation for the sample complexity does not hide large constants. As further evidence of the good practical performance of Algorithm 2, Section 6 shows simulation-based experimental results, run on settings with practical problem parameters for crowdsourcing applications. These results demonstrate the advantages of our algorithm as compared with the straightforward majority vote algorithm as well as the state-of-the-art algorithm [15] (which does not enjoy any theoretical guarantees).

As for our second, optimal, algorithmic construction (Algorithm 5 in Section 3), we combine the adaptive techniques from the Triangular Walk algorithm with a non-adaptive estimation component. More concretely, in the regimes where Algorithm 2 is not optimal, Algorithm 5 uses Algorithm 2 to first give a 2-approximation of ρ, before using this information to non-adaptively estimate ρ much more accurately, while keeping the variance of the estimate small, to control the sample complexity. The theoretical guarantees of Algorithm 5 are shown in Theorem 1.2 (restated and proved in Section 3).

THEOREM 1.2. (INFORMAL) *Given coins where a ρ fraction of the coins have bias $\geq \frac{1}{2}+\Delta$, and $1-\rho$ fraction have bias $\leq \frac{1}{2}-\Delta$, then for large enough constant c, running Algorithm 5 on a budget of $B \geq c\frac{\rho}{\Delta^2\epsilon^2}$ coin flips will estimate ρ to within an additive error of $\pm\epsilon$, with probability at least $2/3$. If the algorithm is repeated $\Theta(\log\frac{1}{\delta})$ times, and the median estimate is returned, then the probability of failure is at most δ.*

1.1.2 Lower Bounds and Discussion

Complementary to our algorithm, we show a matching lower bound of $\Omega(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ samples for a success probability of $1-\delta$ for the problem. Crucially, our bounds match across choices of all four parameters, $\rho, \epsilon, \Delta, \delta$. To show the lower bound, we use the following setup: consider a scenario where all positive coins have bias *exactly* $\frac{1}{2}+\Delta$ and all negative coins have bias *exactly* $\frac{1}{2}-\Delta$.

The overall intuition for our lower bound is that, for each coin, even flipping it enough to learn whether it is a positive or negative coin will tell us little about whether the true fraction of positive coins is ρ versus $\rho+\epsilon$, and thus the flow of information to our algorithm is at most a slow trickle. To capture this intuition, we aim to decompose the analysis into a sum of coin-by-coin bounds; however, the key challenge is the *cross-coin adaptivity* that is available to the algorithm.

To demonstrate the challenge of tightly analyzing cross-coin adaptivity, consider the following natural attempt at a lower bound.

1. Consider flipping a fair coin S to choose between a universe with ρ fraction of positive coins, versus $\rho+\epsilon$ fraction.
2. The aim is to bound the amount of mutual information that the entire transcript of an adaptive coin-flipping algorithm can have with the coin S.
3. Suppose this mutual information can be bounded by the mutual information of the sub-transcript of the i^{th} coin with S, summed over all i.
4. Thus consider and bound the amount of mutual information between the sub-transcript of just coin i alone, with S; and sum these bounds over all coins at the end.

While one would intuitively expect the bounds of Step 4 to be small for each single coin, cross-coin adaptivity allows for each single-coin sub-transcript to encode a lot of mutual information via its *length*, which may be adaptively chosen by the algorithm in light of information gathered across all other coins. The amount of mutual information about S in a sub-transcript may be linear in the number of times *other* coins have been flipped, implying that summing up such mutual information across all coins would yield a bound that uselessly grows quadratically with the number of flips, instead of linearly.

Our approach: We show that no fully-adaptive algorithm can distinguish the following two scenarios with probability at least $1-\delta$, using $o(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ samples: 1) when a ρ fraction of the coins are positive, and 2) when a $\rho+\epsilon$ fraction of the coins are positive. This is formalized as the following theorem (Theorem 1.3), and proved in Section 5.

THEOREM 1.3. *For $\rho \in [0, \frac{1}{2})$ and $\epsilon \in (0, 1-2\rho]$, the following two situations are impossible to distinguish with at least $1-\delta$ probability using an expected $o(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ samples: A) ρ fraction of the coins have probability $\frac{1}{2}+\Delta$ of landing heads and $1-\rho$ fraction of the coins have probability $\frac{1}{2}-\Delta$ of landing heads, versus B) $\rho+\epsilon$ fraction of the coins have probability $\frac{1}{2}+\Delta$ of landing heads and $1-(\rho+\epsilon)$ fraction of the coins have probability $\frac{1}{2}-\Delta$ of landing heads. This impossibility crucially includes fully-adaptive algorithms.*

In Section 5.1, we capture rather generally via Lemmas 5.1 and 5.2 the above intuitive decomposition of a many-coin adaptive algorithm into its single-coin contributions, but via a careful simulation argument that precludes the kind of information leakage between coins that we described above. More explicitly, instead of decomposing a single transcript into many (possibly correlated) sub-transcripts, we relate an n-coin transcript to n *separate* runs of the algorithm (each on freshly drawn random coins), where in the i^{th} run, coin i is authentically sampled (from either the ρ scenario or the $\rho+\epsilon$ scenario), while all the remaining coins are simulated by the algorithm. Crucially, since the remaining simulated coins do not depend on the "real" scenario, no cross-coin adaptivity can leak any information about the real world to coin i, beyond the information gained from flipping coin i itself.

Furthermore, Lemmas 5.1 and 5.2 apply to a broad variety of problem settings, where the population of random variables can be arbitrary and not necessarily Bernoulli coins. We believe these lemmas are of independent interest beyond this work, and can be a useful tool for proving lower bounds for other problem settings, for example a Gaussian variant of the current problem, where instead of being input a noisy yes/no answer on the positivity of an item, we instead receive a numerical Gaussian-distributed score with mean, say, >1 for positive items and <0 for negative items.

Given the decomposition lemmas (Lemmas 5.1 and 5.2), completing the lower bound analysis for the current problem requires upper bounding the squared Hellinger distance between running any single-coin adaptive algorithm on the two coin populations described earlier, with slightly different positive-to-negative mixture ratios. This forms the bulk (and technical parts) of the proof of Theorem 1.3.

Non-adaptive bounds: As motivation for the algorithmic results of this paper, it is reasonable to ask, given Theorem 1.3's lower bound of $\Omega(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ on the number of samples for our problem, is it possible that a *non-adaptive* algorithm can approach this performance, or is the adaptive flavor of Algorithms 2 or 5 required? In our full paper [22], we describe how the framework of the "natural attempt" (the numbered list above) in fact yields a constant probability sample lower bound for *non-adaptive* algorithms that is a $\log\frac{1}{\rho}$ factor higher than that of Theorem 1.3: namely the analog of Theorem 1.3 for non-adaptive algorithms holds for $\Omega(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\rho})$ samples, in the regime of constant δ, and when $\rho \geq \epsilon^2$.

In summary, we have the adaptive and non-adaptive bounds in Table 1. As shown in Table 1, the non-adaptive bounds match each other and the adaptive bounds only in the regime where $\rho = \Theta(1)$ (and in the trivial $\epsilon = \Theta(1)$ regime). In the non-constant ρ regime, the non-adaptive lower bound is asymptotically larger than the adaptive lower bound, demonstrating the need for adaptivity in the design of our final optimal algorithm.

1.1.3 Practical Considerations

The keen-eyed reader might notice that the algorithmic results in Theorems 1.1 and 1.2 both depend on the unknown ground truth ρ, so thus these bounds are not immediately invokable by a user. We present two approaches to address this issue.

The first approach is to note that Algorithm 2 can be interpreted as an *anytime* algorithm: it can produce an estimate at any point in its execution. As more coins are used in Algorithm 2, the estimate simply gains accuracy. Our full paper [22] discusses this approach in more detail, and our experiments in Section 6 are also run using this approach. Because of its simplicity, we recommend this method in practice.

A complication arising from this approach is the fact the sample complexity bound of Theorem 1.1 is an *expected* sample complexity bound. Thus there are potential issues introduced by abruptly stopping the algorithm after a fixed budget of samples, which might inadvertently introduce bias to the estimate. In our full paper [22], we also show how to analyze and address this issue.

The second, theoretically more interesting approach

	Upper Bound	Lower Bound
Adaptive	$O(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ (Algorithm 5)	$\Omega(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ (Section 5)
Non-adaptive	$O(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\epsilon}\log\frac{1}{\delta})$ (Trivial, Example 1.1) $O(\frac{1}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ (Algorithm 3 for $\rho = \Theta(1)$)	$\Omega(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\rho})$ (Full paper, for $\rho \geq \epsilon^2$ and constant δ)

Table 1: Sample Complexity Upper and Lower Bounds

is to fix a total budget of allowable coin flips, and have the algorithm "discover" the optimal achievable accuracy ϵ just from interacting with the different coins. Our presentation and analysis of Algorithm 5, in Section 3, follows this approach. We point out that Algorithm 2 can also be made to have this theoretical guarantee, as demonstrated by the invocation of Algorithm 2 in Algorithm 5.

1.2 Related Work

A related line of work considers the scenario where all positive coins have identical bias (not necessarily greater than 1/2), and negative coins also have identical bias (strictly less than the positive coins' bias), with the ultimate goal of *identifying* any single coin that is positive (or "heavy" in the terminology of these works). The problem has been studied and solved optimally in the context where the biases and positive-negative proportions are known [13], and also when none of this information is known [27, 21]. Such problems may be seen as a special case of bandit problems.

Another related line of work concerns the *learning* of distributions of (e.g. coin) parameters over a population, which arises in various scientific domains [25, 26, 28, 29, 16, 3]. In particular, the works of Lord [25], and Kong et al. [34, 37] consider a model similar to ours, with the crucial difference that each coin is sampled a fixed number t many times—instead of allowing adaptive sampling as in the current work—with the objective of learning the distribution of biases of the coins in the universe.

Since an earlier version of this paper was posted on arXiv, more recent work by Brennan et al. [6] considers a generalization of our setting, but because of different motivation and parameterization, both their upper and lower bounds are not directly comparable with ours.

Our problem also sits in the context of estimation and learning tasks with *noisy* or *uncalibrated* queries. The noiseless version of our problem would be when $\Delta = \frac{1}{2}$ and thus $\frac{1}{2} \pm \Delta$ equals either 0 or 1. That is, all coins are either deterministically heads or deterministically tails, and thus estimating the mixture parameter ρ is equivalent to estimating the parameter of a *single* coin with bias ρ, which has a standard analysis. Prior works have considered noisy versions of well-studied computational problems, such as (approximate) sorting and maximum selection under noisy access to pairwise comparisons [19, 17] and maximum selection under access to uncalibrated numerical scores that are consistent with some global ranking [38].

Furthermore, our problem can be interpreted as a special case of the "testing collections of distributions" model introduced by Levi, Ron and Rubinfeld [23, 24], modulo the distinction between testing and parameter estimation. In their model, a collection of m distributions $(D_1, \ldots, D_m)$ (over the same domain) is given to the tester, and the task is to test whether the collection satisfies a particular *property*, where a property in this case is defined as a subset of m-tuples of distributions. In the *query* access model, one is allowed to name an index $i \in \{1, \ldots, m\}$ and get a fresh sample from the distribution D_i. Our problem can be analogously phrased in this model, where the distributions are over the domain $\{0, 1\}$, and the property in question is whether the fraction ρ of distributions in the collection having bias $\geq 1/2$ is greater than some threshold τ.

We highlight other distribution testing models that allow for adaptive sampling access. For example, in testing contexts, conditional sampling oracles have been considered [12, 8, 11, 10, 18, 1], where a subset of the domain is given as input to the oracle, which in turn outputs a sample from the underlying unknown distribution conditioned on the subset. Evaluation oracles have also been considered [30, 2, 20, 9], where the testing algorithm has access to an oracle that evaluates the probability mass function or the cumulative mass function of the underlying distribution. See the survey by Canonne [7] for detailed comparisons between the different standard and specialized access models, along with a discussion of recent results.

Adaptive lower bounds of problems related to testing monotonicity of high-dimensional Boolean functions have a somewhat similar setup to ours, where binary decisions adaptively descend a decision tree according to probabilities that depend both on the algorithm and its (unknown) input that it seeks to categorize [4, 14]. Lower bounds in these works rely on showing that the probabilities of reaching any leaf in the decision tree under the two scenarios that they seek to distinguish are either exponentially small or within a constant factor of each other. This proof technique is powerful yet does not work in our setting, as many adaptive algorithms have high-probability outcomes that yield non-negligible insight into which of the two scenarios we are in. By contrast, our proof technique involves showing that, while such "insightful" outcomes may be realized with high probability, in these cases we must pay a correspondingly high sample complexity cost somewhere *else* in the adaptive tree.

A crucial part of our lower bound proof, Lemma 5.2, involves carefully "decomposing" fully-adaptive (multi-coin) algorithms into their single-coin components. Work by Braverman et al. [5] gives a data processing inequality in the context of communication lower bounds, whose proof uses similar ideas to how we prove Lemma 5.2.

As described at the beginning of the introduction, results of this work have practical applications in *crowd-sourcing* algorithms in the context of data science and beyond. Theoretical studies with similar aims to our own have been undertaken on handling potentially noisy answers from crowdsourced workers due to lack of expertise [33, 31], (including this work); in practice it is also crucial to understand how to incentivize workers to answer truthfully [32]. Our work also addresses directly the practical problem proposed by Chung et al. [15], to issue queries to potentially unreliable crowdsourced workers in order to estimate the fraction of records containing "wrong" data within a database; here adaptive queries are a natural capability of the model.

2 The Triangular Walk Algorithm

In this section, we present the *Triangular Walk* algorithm (Algorithm 2) for the problem, in the regime where both ρ and the coin biases are unknown. This is an important subroutine of our main, optimal algorithm; and the Triangular Walk algorithm itself can be used as an estimator in its own right. We demonstrate later in Section 6, with simulation results, that this algorithm offers practical advantages over the straightforward majority vote estimator mentioned in the introduction, as well as the state-of-the-art method used in practice.

The Triangular Walk algorithm leverages only single-coin adaptivity, and makes no use of cross-coin adaptivity. At the heart of our algorithm is an estimator (Algorithm 1) that works coin-by-coin, in the regime $\Delta \geq \frac{1}{4}$; subsequently we show how to use this estimator to solve the general problem, with an arbitrary (but known) Δ.

We describe an asymmetric estimator (Algorithm 1) that, given sampling access to a single coin of bias p, returns a real number whose expectation is in $[1 \pm \frac{\epsilon}{2}]$ if $p \geq \frac{3}{4}$, and whose expectation is in $[\pm\frac{\epsilon}{2}]$ if $p \leq \frac{1}{4}$. The estimator is asymmetric in the sense that it will quickly "give up on" coins with $p \leq \frac{1}{4}$, taking only a constant number of samples from them in expectation, while it will more deeply investigate the rare and interesting case of $p \geq \frac{3}{4}$. Below, c will be a constant that emerges from the analysis, where $c \log \frac{1}{\epsilon}$ coin flips suffice to yield an empirical fraction of heads within $poly(\epsilon)$ of the ground truth, p.

Algorithm 1 Single-coin estimate

```
Given: a coin of bias $p$, error parameter $\epsilon$
1. Let $n \leftarrow 0$   (representing the total number of coin flips so far)
2. Let $k \leftarrow 0$   (representing the total number of observed heads so far)
3. Repeat:
   (a) Flip the coin, and increment $n \leftarrow n + 1$
   (b) If heads, increment $k \leftarrow k + 1$
   (c) If $2k \leq n$, return 0 and halt   (majority of flips are tails, evidence that p is small)
   (d) If $n = c \log \frac{1}{\epsilon}$, return $\min(4, \frac{n}{2k-n})$ and halt   (enough flips for concentration)
```

Our overall algorithm robustly combines estimates from running Algorithm 1 on many coins via the standard median-of-means technique. To deal with the general case when Δ might be much smaller than $\frac{1}{4}$, we "simulate a $\frac{1}{4}$-quality coin" by running Algorithm 1 not on individual flips, but rather on the majority vote of blocks of $\Theta(\frac{1}{\Delta^2})$ flips; this majority vote will convert a coin of bias $\leq \frac{1}{2} - \Delta$ to a simulated coin of bias $\leq \frac{1}{4}$, and symmetrically, convert a coin of bias $\geq \frac{1}{2} + \Delta$ to a simulated coin of bias $\geq \frac{3}{4}$.

Theorem 1.1. *Given coins where a ρ fraction of the coins have bias $\geq \frac{1}{2} + \Delta$, and $1 - \rho$ fraction have bias $\leq \frac{1}{2} - \Delta$, then running Algorithm 2 on $t = \Theta(\frac{\rho}{\epsilon^2} \log \frac{1}{\delta})$ randomly chosen coins will estimate ρ to within an additive error of $\pm\epsilon$, with probability at least $1 - \delta$, with an expected sample complexity of $O(\frac{\rho}{\epsilon^2\Delta^2}(1 + \rho \log \frac{1}{\epsilon}) \log \frac{1}{\delta})$.*

The rest of this section concerns the (relatively straightforward) proof of Theorem 1.1, via an analysis of Algorithms 1 and 2; Section 4 instead formulates

```
Algorithm 2 Triangular walk algorithm
Given: $t$ coins, quality parameter $\Delta$, error parameter $\epsilon$, and failure probability parameter $\delta$
  1. For each coin: simulate a new "virtual" coin by computing the majority of $\Theta(\frac{1}{\Delta^2})$ flips each time a "virtual" flip is requested; run Algorithm 1 on each virtual coin, using, inputting $\epsilon$ unchanged, and record the returned estimates.
  2. Partition the returned estimates into $\Theta(\log \frac{1}{\delta})$ groups and compute the mean of each group.
  3. Return the median of the $\Theta(\log \frac{1}{\delta})$ means, or 0 if any of the groups in step 2 are empty.
```

a more general algorithmic framework that adds some perspective to Algorithm 1, and whose abstractions will be crucial to the lower bound analysis in Section 5.

Intuition and analysis of Algorithm 1: Recall that Algorithm 1 is designed to work for coins of constant noise-quality Δ, namely, coins have bias either $\leq \frac{1}{4}$ or $\geq \frac{3}{4}$, and nothing in between. Algorithm 1 halts under two conditions: either the majority of observed flips have been tails—Step 3(c)—or our budget of coin flips (for that coin) is exhausted—Step 3(d). The first stopping condition is designed to make it more likely to halt early for negative coins (coins with bias $p \leq \frac{1}{4}$), even though *all* coins may have a significant chance of halting early. Importantly, the chance of Algorithm 1 halting early depends on the coin's bias p, which is a priori unknown. The output coefficients in Step 3(d) are designed so that the expected output, given any negative coin (of bias $\leq \frac{1}{4}$), is close to 0, and similarly close to 1 given a positive coin (of bias $\geq \frac{3}{4}$). Furthermore, the output coefficients are all bounded by a constant, which gives a constant bound on the variance of the estimate.

Lemma 2.1 captures the guarantees we need from Algorithm 1 in order to analyze the triangular walk algorithm, Algorithm 2.

LEMMA 2.1. *If Algorithm 1 is run with a sufficiently large universal constant c, then the following statements hold.*

1. *Given an arbitrary negative coin (having bias $p \leq \frac{1}{4}$), the output of Algorithm 1 has expectation in $[\pm\frac{\epsilon}{2}]$ and variance upper bounded by ϵ^2. Furthermore, the expected sample complexity in this case is upper bounded by a constant.*

2. *Given an arbitrary positive coin (having bias $p \geq \frac{3}{4}$), the output of Algorithm 1 has expectation in $[1 \pm \frac{\epsilon}{2}]$ and variance upper bounded by a constant. The expected sample complexity in this case is (trivially) upper bounded by $c \log \frac{1}{\epsilon}$.*

The overall expected sample complexity, when the fraction of positive coins is ρ, is $O(1 + \rho \log \frac{1}{\epsilon})$.

Proof. (Sketch) The only nontrivial element of the analysis is understanding the probability that n coin flips of a p-biased coin will produce k heads, without any initial sequence of flips having at least as many tails as heads. This is the product of the Binomial probability that n flips of a p-biased coin will produce k heads, together with the random walk fact known as the "Ballot Theorem", which states that the fraction of sequences of length n containing k heads whose initial segments are all majority-heads equals $\frac{2k-n}{n}$.

From this result, the claims of the lemma all follow from trivial calculations, given a large enough constant c is chosen so as to guarantee that $n = c \log \frac{1}{\epsilon}$ flips of a coin with bias $p \geq \frac{3}{4}$ has at most $O(\epsilon^2)$ probability of having fewer than $\frac{5}{8}n$ heads. For the complete calculations, please refer to our full paper [22]. □

Analyzing Algorithm 2: We conclude by proving Theorem 1.1, which analyzes Algorithm 2.

Proof. [Theorem 1.1] For this proof, we assume that $\rho = \Omega(\epsilon^2)$. Otherwise, the case is handled in Step 3 of Algorithm 2, which returns the valid estimate of 0.

At a high-level, Algorithm 2 runs Algorithm 1 repeatedly on independently chosen coins.

Observe that in Step 1 of Algorithm 2, for each coin we simulate a new "virtual" coin, by using the majority vote of $\Theta(1/\Delta^2)$ coin flips to compute each requested coin flip. By Chernoff bounds, if each given coin has bias either $p \leq \frac{1}{2} - \Delta$ or $p \geq \frac{1}{2} + \Delta$, then the corresponding virtual coin will have bias $p \leq \frac{1}{4}$ and $p \geq \frac{3}{4}$ respectively. Therefore, by Lemma 2.1, the output of Step 1 for each coin is a random variable with expectation in $[\rho \pm \frac{\epsilon}{2}]$. As for the variance of the output, we do the following calculation. Let X_0 denote the random variable that is the output of Algorithm 1 when given a random *negative* coin, and similarly X_1 for a random positive coin. The output of Algorithm 1, which we call Y, is thus distributed as X_1 with ρ probability and as X_0 with $1 - \rho$ probability. The variance of Y is

$$\begin{aligned}\operatorname{Var}[Y] &= \rho \operatorname{Var}[X_1] + (1-\rho)\operatorname{Var}[X_0] + \operatorname*{Var}_{i \leftarrow \text{Bernoulli}(\rho)}[\mathbb{E}[X_i]] \\ &\leq O(\rho) + (1-\rho)\epsilon^2 + \rho(\mathbb{E}[X_1])^2 + (1-\rho)(\mathbb{E}[X_0])^2 \\ &\leq O(\rho) + \epsilon^2 + O(\rho) + O(\epsilon^2) \\ &= O(\rho)\end{aligned}$$

Steps 2 and 3 of Algorithm 2 are the median-of-means method for estimating the mean of a (real-valued) random variable. Using $t = \Theta(\frac{\rho}{\epsilon^2} \log \frac{1}{\delta})$ coins, each of the $\Theta(\log \frac{1}{\delta})$ groups will have $\Theta(\frac{\rho}{\epsilon^2})$ coins and hence outputs from Algorithm 1. By Chebyshev's inequality, with constant probability, the sample mean of each group's estimates will be within $O(\sqrt{\frac{\epsilon^2}{\rho}})$ standard deviations of the expected output of Algorithm 1. The estimation error is therefore equal to $O(\epsilon)$, with a multiplicative constant that can be made arbitrarily small by adjusting the constant in the choice of the number of coins t. Step 3 computes the median of $\Theta(\log \frac{1}{\delta})$ such sample means, which boosts the success probability from constant to $1-\delta$, via standard uses of Chernoff bounds.

Lastly, the total expected sample complexity is the product of 1) the choice of t in the theorem statement, 2) $\Theta(1/\Delta^2)$ which is the number of coin flips used for each majority vote in Step 1, and 3) the sample complexity of Algorithm 1 as stated in Lemma 2.1, yielding $O(\frac{\rho}{\epsilon^2\Delta^2}(1+\rho\log\frac{1}{\epsilon})\log\frac{1}{\delta})$. □

While Theorem 1.1 gives ϵ as input to Algorithm 2 and then asks how many coins are needed to achieve this ϵ error, it will be useful as a preliminary step of our optimal Algorithm 5 to consider the performance of Algorithm 2 where these two roles for ϵ are decoupled. Explicitly, how many coins or samples does it take for Algorithm 2 to achieve error ϵ_1, when Algorithm 2 is given ϵ_2 as input? We will use this result in the regime where the failure probability for Algorithm 2 should be a constant, and thus for simplicity we omit δ from the following statement.

Corollary 2.1. *Given coins where a ρ fraction of the coins have bias $\geq \frac{1}{2}+\Delta$, and $1-\rho$ fraction have bias $\leq \frac{1}{2}-\Delta$, then, for parameters $\epsilon_1, \epsilon_2 > 0$, running Algorithm 2 on $t = \Theta(\frac{\rho}{\epsilon_1^2})$ randomly chosen coins with parameter $\epsilon = \epsilon_2$ will estimate ρ to within an additive error of $\pm\epsilon_1$, with failure probability at most $0.1 + O(t \cdot \text{poly}(\epsilon_2))$, with an expected sample complexity of $O(\frac{\rho}{\epsilon_1^2\Delta^2}(1+\rho\log\frac{1}{\epsilon_2}))$. Note that the degree of the polynomial term (in ϵ_2) in the failure probability can be made arbitrarily high, by choosing a large constant c in Step 3(d) of Algorithm 1.*

The proof is essentially the same as that of Theorem 1.1. For a given ρ and target accuracy ϵ_1, it is easy to see that we need $t = \Theta(\frac{\rho}{\epsilon_1^2})$ coins to estimate ρ to within $\pm\epsilon_1$, even if all examined coins are correctly classified. The parameter ϵ_2 controls, via its logarithm, the maximum number of flips with which we examine any single coin. Namely, an increase in the number of coin flips will exponentially reduce the bias of the per-coin estimator, with the corollary aimed at the regime where the ϵ_1 error from the process of sampling coins (as opposed to flipping coins once they are sampled) dominates the error.

3 The Main Algorithm

Here we present our main algorithm, Algorithm 5, analyzed in Theorem 1.2, which uses $O(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ samples, matching the fully-adaptive lower bound we prove in Section 5.

Algorithm 5 uses the Triangular Walk estimator as a subroutine and has a hybrid flavor, combining both (single-coin) adaptive and non-adaptive techniques, where the algorithm is increasingly adaptive for smaller values of ρ. Crucially, in the adaptive component of Algorithm 5, we use the Triangular Walk estimator to provide a 2-approximation to ρ, and a variant of the algorithm to "filter" out most negative coins such that we get a constant ratio of positive vs negative coins, to reduce variance. The coins "surviving" the filter are then fed into a new, non-adaptive algorithm (Algorithm 3) that we call "refined sampling", which like Algorithm 1 flips different coins a *different* number of times, yet the number of flips is chosen *non-adaptively*; the information from different coins is combined in a subtle way.

As a general motivation, consider taking t coins, flipping them n times each, and trying to estimate the fraction of positive coins. For a slightly different setting that may have cleaner intuition, consider having sample access to many univariate Gaussian distributions of bounded variance, some of which have mean ≤ 0 and some of which have mean ≥ 1, where the goal is to estimate the fraction of "positive" Gaussians with as few samples as possible. If we take n samples from a given distribution, then testing whether the sample mean is $> \frac{1}{2}$ lets us correctly determine its identity with probability $1-\exp(-n)$, incentivizing us to choose a large n. However, for a fixed budget on the total number of samples across all distributions, choosing many samples per distribution means we can only sample from a limited number of distributions, introducing sampling errors across distributions (as opposed to within each distribution), and thus introducing a variance into our estimate inversely proportional to the number of coins sampled, and thus $O(n/T)$ for a total budget of T. This is the classic bias-variance tradeoff, where larger n induces a better bias but worse variance.

While in many settings, one might try to find an optimal n that balances these two concerns, the right answer here is instead to combine the two approaches: sample some distributions many times, to get a low-bias signal, and also sample many distributions a few times, to get a low-variance signal; and combine these two signals with care. Explicitly, the coefficients in Step

3 of Algorithm 3 are carefully chosen so that their contributions "telescope" in expectation between distributions sampled different numbers of times, allowing, essentially, all the high-variance terms to cancel out without worsening the bias.

We first present the non-adaptive component (Algorithm 3) of Algorithm 5 for estimating smooth functions f on the underlying coin bias p, which has constant expected sample complexity, with *zero bias*, at the cost of $O(1)$ variance instead of $O(\rho)$ variance as in Algorithm 2. This will be combined with a single-coin adaptive "filtering" component such that only an $O(\rho)$ fraction of coins will be used in running Algorithm 3, giving an overall $O(\rho)$ variance in Algorithm 5.

Think of the function f as being analogous to the output coefficient $\frac{n}{2k-n}$ of Algorithm 1—correcting for a probabilistic filtering mechanism, such that the expected output of f for those coins that survive filtering will be essentially 0 for negative coins ($p \le \frac{1}{4}$), 1 for positive coins ($p \ge \frac{3}{4}$), and smoothly transitions between 0 and 1 in between. See later in Definition 3.1 for the precise instantiation of $f(p)$ we need.

Let $\text{Bin}(n, p, k)$ denote the probability that a Binomial distribution with n trials and bias p outputs k.

Algorithm 3 Refined Sampling

```
Input: sample access to a coin of bias $p$; target function $f : [0,1] \to \mathbb{R}$
1. Choose a number of coin flips $n$ that is a power of 2, choosing $2^i$ with probability $\frac{\sqrt{8}-1}{\sqrt{8}}(2^i)^{-1.5}$,
   where $\frac{\sqrt{8}-1}{\sqrt{8}}$ is the normalizing constant so that the probabilities sum to 1.
2. Flip the coin $n$ times, and let $k$ be the number of observed heads.
3. Return $\frac{n^{1.5}\sqrt{8}}{\sqrt{8}-1}\left(f(\frac{k}{n}) - \sum_{i=0}^{n/2} f(\frac{i}{n/2}) \cdot \binom{n/2}{i}\binom{n/2}{k-i}/\binom{n}{k}\right)$
```

The sum in Step 3 of the algorithm is omitted if the power of 2 chosen for the number of coin flips is $n = 1$, in which case $\binom{n/2}{i}$ would be undefined. We now describe the properties of Algorithm 3 in Lemma 3.1.

LEMMA 3.1. *Given a coin of bias p, and given a function $f : [0,1] \to \mathbb{R}$ that is bounded by a universal constant, and has 2nd derivative bounded by a universal constant, then Algorithm 3 will return an estimate of $f(p)$ that has bias 0, variance $O(1)$, and uses $O(1)$ samples in expectation.*

Proof. The expected number of coin flips taken by Algorithm 3 is the sum of a fixed geometric series, and is thus $O(1)$ as desired.

We bound the variance of the algorithm by showing that, for each depth n, the values returned in Step 3 will have magnitude $O(n^{0.5})$. Consider the sum in the second term of the expression of Step 3. The expression $\binom{n/2}{i}\binom{n/2}{k-i}/\binom{n}{k}$ can be interpreted as: given a sequence of n coin tosses of which k were heads, if a random subsequence of length $n/2$ is chosen, what is the probability that i heads are chosen. This distribution has expectation $\frac{k}{2}$, and variance $< n$. Since f has second derivative bounded by a constant, the difference of f from $f(\frac{k}{n})$ is upper and lower bounded by quadratics centered at $\frac{k}{n}$. Thus the difference between $f(\frac{k}{n})$ and the expected value of $f(\frac{i}{n/2})$ when i is drawn from the distribution with pmf $\binom{n/2}{i}\binom{n/2}{k-i}/\binom{n}{k}$ is bounded by a constant times the variance of the random variable $\frac{i}{n/2}$, namely $O(\frac{1}{n})$. Therefore, when multiplied by $\frac{n^{1.5}}{\sqrt{8}-1}$, the output of Step 3 will be bounded by $O(n^{0.5})$ as desired. Since in Step 1, n is chosen with probability $\frac{\sqrt{8}-1}{n^{1.5}}$, the contribution to the variance from a particular n is at most $\frac{\sqrt{8}-1}{n^{1.5}}O(n^{0.5})^2 = O(n^{-0.5})$; summing this bound over all n that are powers of 2 yields a constant, $O(1)$, variance, since geometric series converge.

To analyze the expectation of the values returned in Step 3 of Algorithm 3, we show that it telescopes across the different depths n. Namely, consider the expected contribution just of the second (sum) term at level n, $-\sum_{k=0}^{n} \text{Bin}(n,p,k) \sum_{i=0}^{n/2} f\left(\frac{i}{n/2}\right) \cdot \binom{n/2}{i}\binom{n/2}{k-i}/\binom{n}{k}$. The coefficient in this expression of a given $f(\frac{i}{n/2})$ equals $-\sum_{k=0}^{n} \text{Bin}(n,p,k)\binom{n/2}{i}\binom{n/2}{k-i}/\binom{n}{k}$; from the discussion at the start of the proof, the k^{th} term of this sum can be reinterpreted as the probability that, in n tosses of a coin of bias p, we have k heads total, and i heads among the first $n/2$ tosses; summed over all k this is clearly just the probability that i heads will be observed among $n/2$ tosses, namely $\text{Bin}(\frac{n}{2}, p, i)$. Thus the expected value of the sum term of Step 3 at level n is $-\sum_{i=0}^{n/2} \text{Bin}(\frac{n}{2}, p, i) f(\frac{i}{n/2})$, which is exactly the negation of the expectation of the first term of Step 3, at level $n/2$. (The multiplier $\frac{n^{1.5}\sqrt{8}}{\sqrt{8}-1}$ in Step 3 is exactly canceled out by the probability of choosing n in Step 1.)

Thus the expected output of the algorithm, considering only contributions up to some depth $n = 2^i$, collapses to just the expectation of the first term of Step 3 at the deepest level, n. This expected output is thus $\sum_{k=0}^{n} f(\frac{k}{n}) \cdot \text{Bin}(n,p,k)$, namely the expected value of $f(\frac{k}{n})$ when k is drawn from a binomial distribution with n trials and bias p. Since the binomial distribution $\text{Bin}(n,p,\cdot)$ has expectation pn and variance $< n$, and since f has 2nd derivative bounded by a constant, we have that this expectation converges to $f(p)$ for large n;

namely, $|f(p) - \sum_{k=0}^{n} f(\frac{k}{n}) \cdot \mathrm{Bin}(n,p,k)| = O(\frac{1}{n})$. Thus, as n goes to infinity, we see that the expected output of Algorithm 3 converges to $f(p)$, as claimed. ∎

We now give a new non-adaptive algorithm, Algorithm 4, in order to motivate the choice of $f(p)$ that we use for Algorithm 3 within Algorithm 4. Algorithm 4 will be a major component of our final algorithm, Algorithm 5.

Algorithm 4 Optimal Algorithm given an estimate $\hat{\rho}$

```
Given: A total budget $B$ of coin flips, quality parameter $\Delta$, and an estimate $\hat{\rho}$ that is within a factor of 2 of $\rho$
1. Run the following on $t = \Theta(\Delta^2 B)$ randomly drawn coins. For each coin: simulate a new "virtual" coin by computing the majority of $\Theta(\frac{1}{\Delta^2})$ flips each time a "virtual" flip is requested, so that each virtual coin will have probability either $p \leq \frac{1}{4}$ or $p \geq \frac{3}{4}$.
    (a) For each virtual coin, flip it at most $d = \Theta(\log \frac{1}{\hat{\rho}})$ times but stop if at any point the majority of flips are tails.
    (b) If the previous step did not stop early, then run Algorithm 3 for the function $f_d(p)$ of Definition 3.1.
2. Return $\frac{1}{t}$ times the sum of all the values output by Algorithm 3 in Step 3(b).
```

As mentioned above, the choice of $f(p)$ is a correction for the filtering mechanism. Concretely, in Algorithm 4, Step 2(a) will stop early on negative coins with probability that is increasingly high for smaller ρ, significantly reducing the number of coin flips; and in Step 2(b) we exactly compensate for this (a priori) unknown early stopping probability by running the unbiased Algorithm 3 on an appropriately chosen function $f_d(p)$ that is exactly the inverse of this early stopping probability, for positive coins, and 0 for negative coins:

Definition 3.1. *Given a depth d, let $f_d(p) : [0,1] \to \mathbb{R}$ be defined to equal 0 for $p \leq \frac{1}{4}$; and for $p \geq \frac{3}{4}$, let $f_d(p)$ equal 1 divided by the probability that a sequence of d flips of a coin of bias p never has a majority-tails initial sequence; for $\frac{1}{4} < p < \frac{3}{4}$, let $f_d(p)$ be chosen so as to smoothly connect the regions $p \leq \frac{1}{4}$ and $p \geq \frac{3}{4}$ so that $f_d(p)$ has second derivative bounded by a universal constant (independent of d).*

With this choice of $f(p)$, we state and prove Proposition 3.1, which gives the soundness and sample complexity bounds for Algorithm 4.

Proposition 3.1. *On input 1) a budget $B = O(\frac{\rho}{\epsilon^2 \Delta^2})$ of coin flips, 2) the quality parameter Δ and 3) a 2-approximation $\hat{\rho}$ of ρ, Algorithm 4 returns an estimate of ρ that has additive error at most ϵ with probability at least 0.99, using at most B coin flips.*

Proof. We first show the expected output of Algorithm 4 equals ρ. For each positive coin, Step 1 transforms it into a "virtual" coin of probability $p \geq \frac{3}{4}$; this coin will "survive" Step 1(a) with probability exactly $1/f_d(p)$, by definition of $f_d(p)$ in Definition 3.1. Thus Algorithm 3 will return an estimate of $f_d(p)$, with bias 0. Multiplying through by the survival probability $1/f_d(p)$, and by the probability ρ that a positive coin will be drawn, we see that, over t coins, the expected contribution to the estimate from Step 2 of the *positive* coins will be ρ. For each negative coin, by definition $f_d(p) = 0$, so the expected contribution from these coins, added over all $\leq t$ of them, and scaled by $\frac{1}{t}$ in Step 2, will be 0.

To bound the variance of the output of Step 2, we note that at most a 2ρ fraction of the coins reach Step 1(b): a ρ fraction of the coins are positive; meanwhile, negative coins, where $p \leq \frac{1}{4}$, have $\exp(-d)$ probability of surviving Step 1(a), which can be made $\leq \rho$ since $d = \Theta(\log \frac{1}{\hat{\rho}})$. Thus the output returned in Step 2 is $\frac{1}{t}$ times the sum of t independent trials of a process that, with probability $\leq 2\rho$ outputs a random variable whose expected squared magnitude is bounded by a constant (by Lemma 3.1). For $t = \Theta(\frac{\rho}{\epsilon^2})$, the expected squared magnitude—and hence the variance—of the output of the algorithm is thus bounded by $O(\frac{t\rho}{t^2}) = O(\epsilon^2)$. Thus by Chebyshev's inequality, Step 2 will return an estimate accurate to within $O(\epsilon)$, with constant probability.

Lastly, we need to verify that Step 1 will exceed the coin flip budget only with small constant probability. It suffices, using a Markov's inequality argument, to bound the expected number of coin flips used in the steps. We consider the number of ("virtual") flips from Step 1(a), and also Step 1(b), and then multiply by $\Theta(\frac{1}{\Delta^2})$ as described in Step 1. For a negative coin, the expected number of flips until a majority-tails initial sequence is observed in Step 1(a) is constant by standard random walk analysis, leading to an $O(\Delta^2 B)$ term; for positive coins, there are on average $O(\rho\Delta^2 B)$ of them, so we could afford to flip each $O(\frac{1}{\rho})$ times, but Step 1(a) uses only at most $d = \Theta(\log \frac{1}{\hat{\rho}})$ flips. Step 1(b) is run on an expected $\leq 2\rho$ fraction of the coins, as explained above; and by Lemma 3.1, Algorithm 3 takes $O(1)$ expected samples, for a total bound of $O(\rho\Delta^2 B)$ virtual flips from Step 1(b). (Algorithm 3 could thus afford to take up to $O(\frac{1}{\rho})$ samples on average, so, interestingly, there is a lot of slack here.)

Thus in total we use $O(\Delta^2 B) = O(\frac{\rho}{\epsilon^2})$ virtual flips, each requiring $\Theta(1/\Delta^2)$ real flips, corresponding to expected sample complexity of $O(B) = O(\frac{\rho}{\epsilon^2\Delta^2})$. ∎

Having analyzed Algorithm 4, we can now present our final optimal algorithm, stated as Algorithm 5. The theoretical guarantees are given in Theorem 1.2, restated and proved below.

We stress again that, in our presentation of Algorithm 5, the error parameter ϵ (of Theorem 1.2) is not known, since it depends on the budget B and the unknown ground truth ρ, yet the returned estimate will have this optimal ϵ accuracy regardless. This is achieved by Algorithm 5's calls to Algorithm 2 and Algorithm 4, which collectively cover all regimes of how ρ and ϵ relate to each other, yielding optimal error guarantees in each case.

Algorithm 5 Optimal Algorithm

```
Given: A total budget $B$ of coin flips and quality parameter $\Delta$
  1. Use Algorithm 2 in Section 2 on $O(\Delta^2 B)$ many coins (a small fraction of $B$), using an "$\epsilon$" that is $\Theta(1/(\Delta^2 B))$, and a constant $\delta$. Let $\hat{\rho}$ be the returned estimate of $\rho$.
      (a) If Algorithm 2 ever tries to use more than $B/4$ coin flips total, then terminate Algorithm 2 and move onto the next step.
      (b) Otherwise, return the estimate produced by Algorithm 2.
  2. Use Algorithm 2 on $\Theta(\sqrt{\Delta^2 B})$ freshly drawn coins, using again an "$\epsilon$" that is $\Theta(1/(\Delta^2 B))$, and a constant $\delta$. The returned estimate $\hat{\rho}$ will be a 2-approximation to $\rho$. If in this step, Algorithm 2 tries to use more than $B/4$ coin flips, terminate and fail, which happens only with small constant probability.
  3. Run Algorithm 4 on input $B/2$, $\Delta$, and $\hat{\rho}$, and return its answer.
  4. (If a sub-constant failure probability $\delta$ is desired, then repeat the entire algorithm $\Theta(\log \frac{1}{\delta})$ times and return the median of the outputs, ignoring invocations that failed.)
```

THEOREM 1.2. *Given coins where a ρ fraction of the coins have bias $\geq \frac{1}{2} + \Delta$, and $1-\rho$ fraction have bias $\leq \frac{1}{2} - \Delta$, then running Algorithm 5 on a budget of B coin flips will estimate ρ to within an additive error of $\pm\epsilon$, with probability at least $2/3$, where ϵ is implicitly defined by the relation $B = \Theta(\frac{\rho}{\Delta^2\epsilon^2})$ based on the unknown ground truth ρ. If the algorithm is repeated $\Theta(\log \frac{1}{\delta})$ times, and the median estimate is returned, then the probability of failure is at most δ.*

Proof. Given the fixed total sample complexity budget of B coin flips, and fixing the unknown ground truth ρ, the target additive error parameter ϵ is defined by the sample complexity equation $B = \Theta(\frac{\rho}{\epsilon^2\Delta^2})$. There are two cases, either $\log \frac{1}{\epsilon} \leq c/\rho$ for some sufficiently small universal constant c (in which case we show that, with high probability, Step 1 will output a correct answer and then halt), or the inequality is in the opposite direction (in which case, with high probability, either Step 1 still produces a correct answer and halts, or Steps 2 and 3 will output a correct answer).

In the case where $\log \frac{1}{\epsilon} \leq c/\rho$, we use Corollary 2.1 with parameters $\epsilon_1 = \epsilon$ and $\epsilon_2 = \Theta(\frac{1}{\Delta^2 B}) = \Theta(\frac{1}{t})$: Algorithm 2 will have error $\pm\epsilon$, except with failure probability $0.1 + O(t \cdot \text{poly}(\epsilon_2)) = 0.1 + O(t \cdot \text{poly}(\frac{1}{t}))$, where, as noted in Corollary 2.1, we may make the polynomial superlinear to make this failure probability $0.1 + o(1)$. Further, the expected sample complexity is $O(\frac{\rho}{\epsilon_1^2\Delta^2}) = O(B)$ in the case where $\log \frac{1}{\epsilon} \leq c/\rho$, so by Markov's inequality, for appropriate constants we can ensure that Algorithm 2 uses $\leq B/4$ samples with high constant probability. Thus in this case, the algorithm will correctly terminate in Step 1(b) with high probability.

Next, we analyze the case where $\log \frac{1}{\epsilon} \geq c/\rho$. By Corollary 2.1, as above, if Step 1(b) is reached then its answer will be ϵ-accurate except with some small constant probability. Otherwise, since Steps 2 and 3 are statistically independent of Step 1, we can just analyze these steps for the case $\log \frac{1}{\epsilon} \geq c/\rho$, ignoring what happened in Step 1.

We first claim that Step 2 will return a 2-approximation $\hat{\rho}$ of ρ with high constant probability. As before, we use Corollary 2.1 with $\epsilon_2 = \epsilon$; since (from the algorithm and the parameters of the theorem) this step uses $t = \Theta(\sqrt{\Delta^2 B}) = \Theta(\frac{\sqrt{\rho}}{\epsilon})$ coins, solving the equation $t = \Theta(\frac{\rho}{\epsilon_1^2})$ of the Corollary yields $\epsilon_1 = \Theta(\sqrt{\epsilon\sqrt{\rho}}) = O(\sqrt{\epsilon})$. Since we are in the regime where $\log \frac{1}{\epsilon} \geq c/\rho$, we have that $\epsilon_1 = O(\sqrt{\epsilon}) \leq O(e^{-c/\rho}) \ll \rho/2$ for sufficiently small ρ, meaning that we will approximate ρ to within $\pm\rho/2$, giving us a 2-approximation. The failure probability is $0.1 + o(1)$ as above. From Corollary 2.1, the expected sample complexity, in our case $\log \frac{1}{\epsilon} \geq c/\rho$ will be $O(\frac{\rho}{\epsilon_1^2\Delta^2}\rho \log \frac{1}{\epsilon_2})$; substituting in the definitions of ϵ_1, ϵ_2 yields $O(\frac{\rho^{3/2}}{\epsilon\Delta^2} \log \frac{1}{\epsilon})$. Since $\rho = O(1)$ and $\log \frac{1}{\epsilon} = o(\frac{1}{\epsilon})$ this expected sample complexity is thus $O(\frac{\rho}{\epsilon^2\Delta^2}) = O(B)$ and Markov's inequality implies the algorithm exceeds its sample bound in Step 2 with an arbitrarily small constant probability.

We conclude by invoking Proposition 3.1 to show that the estimate returned in Step 3 by Algorithm 4 is accurate to within additive error ϵ except with small constant probability. ∎

4 Characterizing Single-Coin Algorithms

As a crucial first step towards the lower bounds of Section 5 that analyze how information from many different coins may interact, in this section we describe a unified framework for characterizing (adaptive) algorithms that flip only a single coin. Section 5.1 will then show a general structural result describing how any adaptive multi-coin algorithm may be broken into single-coin subroutines that may then be analyzed in light of the characterization of this section.

The most general form of an adaptive single-coin algorithm is a decision tree, where each node is a coin flip, and has two outgoing edges denoting the outcome of the coin flip, heads or tails; the current node captures the outcome of the entire sequence of coin flips so far, and thus for each node, a generic algorithm specifies a probability of halting, versus continuing from that node.

Via a (standard) symmetrization argument, instead of considering the state of the algorithm to be an arbitrary sequence of coin flips, we instead aggregate this information into a pair (n, k) representing the number of coin flips, and the number of heads observed so far. In outline, one may prove by induction on the number of coin flips n that any such decision tree may be "symmetrized" so that its stopping probability at each node $(n' \leq n, k)$ depends only on n' and k, while preserving, for any underlying coin bias p, the total probability of hitting the set of decision tree nodes that represent observing k total heads out of n' flips. The inductive step relies on the fundamental property that, conditioned on observing exactly k heads out of n' coin flips, the distribution over all such sequences of coin flips is independent of the coin bias p, and depends only on the stopping probabilities along each of the $\binom{n}{k}$ paths in the decision tree. This is a direct generalization of the analogous observation in the triangular walk algorithm section (Section 2), and is analyzed in slightly different form in Equation 4.1 below.

We thus consider single-coin algorithms as random walks (Algorithm 6) on the structure of the Pascal Triangle, in which the states are represented by pairs (n, k), where n is the total number of flips of the coin so far, and $k \leq n$ is the number of "heads" responses. At each state (n, k), the algorithm terminates with some probability $\gamma_{n,k}$, else the algorithm may request a further coin flip and continue the walk. The collection of parameters $\gamma_{n,k}$ we call a *stopping rule*, and specifies that algorithm's behavior.

Algorithm 6 Triangular Walk

```
Input: a coin of bias $p$
  1. Initialize state $(n, k)$ to $(0, 0)$.
  2. Repeat until termination:
      (a) With probability $\gamma_{n,k}$, terminate and output $(n, k)$.
      (b) Otherwise, sample one more coin flip. Increment $n$, and increment $k$ by the result of the flip (0 or 1).
```

This formulation of single-coin algorithms, which we call a *triangular walk*, reveals structure that will be useful to the rest of the analysis of this paper. In particular, since the overall objective of running an adaptive coin-flipping algorithm is to recover information about the bias p of the coin (while minimizing expected sample complexity), it is fortuitous (as we will see) that the outcome of such an algorithm depends on p in an unexpectedly transparent way. This is given in Definition 4.1.

Definition 4.1. *Given a stopping rule $\{\gamma_{n,k}\}$, we define coefficients $\{\alpha_{n,k}\}$, $\{\beta_{n,k}\}$, and $\{\eta_{n,k}\}$, so that, for any $p \in [0, 1]$, the triangular walk with stopping rule $\{\gamma_{n,k}\}$ on a coin of bias p, the coefficients have the semantics: $\alpha_{n,k}p^k(1-p)^{n-k}$ represents the probability that the walk terminates at (n, k), with all such probabilities summing to 1; $\beta_{n,k}p^k(1-p)^{n-k}$ represents the probability that the triangular walk encounters (n, k), whether or not it terminates there, and $\eta_{n,k}p^k(1-p)^{n-k}$ is the probability that the triangular walk encounters (n, k) but does* not *terminate there. Each of these reparameterizations of the stopping rule may be derived from $\{\gamma_{n,k}\}$ using the following relations.*

$$\begin{aligned}
(4.1) \quad \beta_{0,0} &= 1 \\
\beta_{n+1,k+1} &= \beta_{n,k+1} \cdot (1 - \gamma_{n,k+1}) + \beta_{n,k} \cdot (1 - \gamma_{n,k}) \\
\alpha_{n,k} &= \beta_{n,k} \cdot \gamma_{n,k} \\
\eta_{n,k} &= \beta_{n,k} - \alpha_{n,k} \; (= \beta_{n,k} \cdot (1 - \gamma_{n,k})\,).
\end{aligned}$$

Consider the original setting, where one has a universe of (different) coins; one might repeatedly run a single-coin algorithm on coins drawn from the universe, and somehow combine their outputs into a final answer. There are many conceivable ways of aggregating the outputs of single-coin algorithms into an estimate, and the lower bounds of Section 5 consider them all. However, a particularly natural and powerful approach is to construct a linear estimator, namely to have the single-coin algorithm output a real number coefficient $v_{n,k}$ at each termination node, with the overall algorithm estimating the expected output of the single-coin algorithm, across the coins in the universe. Algorithm 2 works this way, using the median-of-means method (instead of taking the sample mean) to estimate the expected output of Algorithm 1. Such linear estimators are surprisingly flexible, and are known to be optimal in certain classes of estimation tasks [36].

5 Fully-adaptive lower bounds

We show in this section that Algorithm 5 is optimal in all four problem parameters ρ, ϵ, Δ and δ, even when compared to all fully-adaptive algorithms that are adaptive across different coins. In particular, we show the following indistinguishability result (Theorem 1.3).

Theorem 1.3. *For $\rho \in [0, \frac{1}{2})$ and $\epsilon \in (0, 1-2\rho]$, the following two situations are impossible to distinguish with at least $1-\delta$ probability using an expected $o(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ samples: A) ρ fraction of the coins have probability $\frac{1}{2}+\Delta$ of landing heads and $1-\rho$ fraction of the coins have probability $\frac{1}{2}-\Delta$ of landing heads, versus B) $\rho+\epsilon$ fraction of the coins have probability $\frac{1}{2}+\Delta$ of landing heads and $1-(\rho+\epsilon)$ fraction of the coins have probability $\frac{1}{2}-\Delta$ of landing heads. This impossibility crucially includes fully-adaptive algorithms.*

With the algorithmic result of Theorem 1.2, this lower bound is therefore tight to within a constant factor. We note that the restrictions $\rho < \frac{1}{2}$ and $\epsilon \leq 1-2\rho$ reflect the symmetry of the problem, where the pair $\rho, \rho+\epsilon$ is exactly as hard to distinguish as the pair $1-\rho-\epsilon, 1-\rho$, yielding analogous results for the symmetric parameter regime.

Example 5.1. *Even in the constant failure probability regime, the $\Omega(\frac{\rho}{\epsilon^2\Delta^2})$ lower bound requires significant analysis, forming the bulk of the remainder of this paper, but two special cases have direct proofs. When $\Delta = \Theta(1)$ we can prove a $\Omega(\frac{\rho}{\epsilon^2})$ lower bound without the Δ dependence: consider the case where all coins are unbiased and perfect, meaning that the only source of randomness is from the mixture of coins, which is itself a Bernoulli distribution of bias either ρ or $\rho+\epsilon$. We quote the standard fact that, in order to estimate a Bernoulli coin flip of bias ρ to up to additive ϵ, we need $\Omega(\frac{\rho}{\epsilon^2})$ samples to succeed with constant probability; this can be proven by a standard (squared) Hellinger distance argument. On the other hand, it is also straightforward to prove a $\frac{1}{\Delta^2}$ lower bound (covering the regime where ρ and ϵ are constant): consider the easiest regime for ρ and ϵ, where $\rho = 0$ and $\epsilon = 1$; thus coins either all have $\frac{1}{2}+\Delta$ bias or all have $\frac{1}{2}-\Delta$ bias. To distinguish whether we have access to positive coins or negative coins requires $\Omega(\frac{1}{\Delta^2})$ samples.*

In order to show Theorem 1.3, we use the Hellinger distance and KL-divergence between probability distributions as proxies for bounding the total variation distance.

Definition 5.1. (Hellinger Distance) *Given two discrete distributions P and Q, the* Hellinger distance *$H(P,Q)$ between them is*

$$\frac{1}{\sqrt{2}}\sqrt{\sum_i(\sqrt{p_i}-\sqrt{q_i})^2} = \sqrt{1-\sum_i\sqrt{p_iq_i}}$$

Definition 5.2. (KL-divergence) *Given two discrete distributions P and Q, the* KL-divergence *$D_{\mathrm{KL}}(P||Q)$ between them is*

$$\sum_i p_i \log\frac{p_i}{q_i}$$

The following facts capture how the Hellinger distance and KL-divergence can be used to show sample complexity lower bounds.

Fact 5.1. (Chapter 2.4, [35]) *For any two distributions P and Q over the same domain, we have*

$$\ell_1(P,Q) \leq \sqrt{2}H(P,Q)$$

and furthermore, for any event E,

$$P(E) + Q(\bar{E}) \geq \frac{1}{2}e^{-D_{\mathrm{KL}}(P||Q)}$$

The second inequality is also known as the high-probability Pinsker inequality.

Recall from the introduction that, the main challenge in proving a general lower bound for our problem lies in analyzing the two kinds of adaptivity that algorithms may employ that were both absent in the special cases of Example 5.1. Explicitly, when taking samples from a given coin, we can choose whether to ask for another sample based on A) previous results of this coin, and also B) previous results of all the other coins. This first kind of adaptivity, “single-coin adaptivity”, is crucially used in the algorithms presented in the rest of the paper (e.g. the “shape” of the stopping rule for our triangular-walk algorithms); in Proposition 5.1 we analyze the best possible performance of such triangular stopping rules. The most interesting part of the proof of Theorem 1.3 consists of showing that the second kind of adaptivity (cross-coin adaptivity) cannot help in the lower bound setting, which we analyze via general Hellinger distance/KL-divergence inequalities (Lemmas 5.1 and 5.2) in Section 5.1.

5.1 Reduction to Single-Coin Adaptive Algorithms In this section, we give two related but distinct reductions to single-coin adaptive algorithms. The first is a general decomposition ("direct sum") inequality that decomposes the squared Hellinger distance of running a fully-adaptive algorithm on two different coin populations into the sum of squared Hellinger distances of running single-coin adaptive algorithms on the two coin populations. This inequality will lead to a constant probability sample complexity lower bound. The second inequality instead decomposes the KL divergence into (a constant times) a sum of squared Hellinger distances, however with an additional slight restriction that the two coin populations being considered must be very close to each other. The upside to using this second inequality is that, an upper bound on the KL divergence combined with the high probability Pinsker inequality allows us to obtain a *high probability* sample complexity lower bound, which in particular is tight in *all* parameters of the problem, up to a multiplicative constant.

Both of the following inequalities are applicable to populations of variables *beyond* Bernoulli coins. We believe that the general inequalities are of independent interest to the community, since they would be applicable and useful for proving lower bounds on a variety of scenarios involving, for example, a Gaussian variant of the current problem, where instead of getting yes/no answers on the positivity of an item, one gets a real-valued score which correlates with the positivity of the item.

We phrase both lemmas as upper bounds on distances between distributions of the *transcript* of an algorithm, which when combined with the data processing inequality immediately yields upper bounds on distances between distributions of the algorithm's *output*. See, for example, the very end of the proof of Theorem 1.3.

Lemma 5.1. *Consider a problem setting where there is a collection of random variables, and an adaptive algorithm can draw variables from the collection and draw independent samples from the drawn variables. Now consider an arbitrary algorithm that iteratively samples from random variables drawn from the collection, choosing each subsequent variable to sample in an arbitrary adaptive manner based on the results of previous sample outcomes. Suppose the algorithm terminates almost surely. Consider two arbitrary collections of random variables, denoted by distributions $\mathcal{A}$ and $\mathcal{B}$ over the set of possible random variables. Let H^2_{full} be the squared Hellinger distance between the transcript of a single run of the algorithm where 1) the random variables are drawn from $\mathcal{A}$ versus where 2) the random variables are drawn from $\mathcal{B}$. Furthermore, let H^2_i be the squared Hellinger distance between the two scenarios, but instead of running the algorithm as is, we only use random variable i (as drawn either from $\mathcal{A}$ or $\mathcal{B}$ depending on the scenario) and simulate all other random variables as independent random variables that are themselves drawn from the mixture distribution $\frac{\mathcal{A}}{2} + \frac{\mathcal{B}}{2}$. Then*

$$H^2_{\text{full}} \leq \sum_{\text{variable } i} H^2_i$$

Proof. It suffices to prove the result for deterministic algorithms, since squared Hellinger distance is linear with respect to mixtures of distributions *with distinct outcomes*, and a randomized algorithm is simply a mixture of deterministic algorithms which also records which of the algorithms the random coins picked. Furthermore, the following proof is phrased in terms of the special case where the collection of random variables are Bernoulli coins (which is the setting considered in this paper). Barring measure-theoretic formalization issues that we do not discuss, the proof generalizes directly to populations of arbitrary random variables.

A deterministic fully-adaptive algorithm is a decision tree, where each node is labeled by the identity of the coin the algorithm chooses to flip next conditioned on reaching this node, and each edge out of a node is labeled by a heads or tails result for this coin. We can view a run of the algorithm as follows: 1) first draw all the random coins from either $\mathcal{A}$ or $\mathcal{B}$ depending on the scenario, and then 2) flip these coins according to this fully-adaptive algorithm—we view choosing the coins from $\mathcal{A}$ or $\mathcal{B}$ as happening at the beginning since all these samples are free and only the coin flips themselves are counted. After step 1, fixing the bias of each coin, the probability of ending up at the i^{th} leaf of the decision tree is simply the probability (over coin flips) that every edge along the path from the root to that leaf is followed. Note that each edge is a probabilistic event depending on only one coin. Therefore, this probability can be factored into a product of probabilities, one term for each of the coins. For example, suppose the path to leaf i involves coin j returning 5 heads in a row, then getting some particular sequence from flipping some *other* coins, then coin j returning another 2 heads followed by 3 tails. Then, if coin j has bias p_j, it contributes $p_j^{5+2}(1-p_j)^3$ to the probability product.

We denote by $q^{\mathcal{A}}_{j,i}$ the *expected* contribution of coin j to the probability product for leaf i, over the randomness of $\mathcal{A}$ on the bias of coin j. In the previous example, $q^{\mathcal{A}}_{j,i}$ would be equal to $\mathbb{E}_{p\leftarrow\mathcal{A}}[p^7(1-p)^3]$. We similarly define $q^{\mathcal{B}}_{j,i}$. Explicitly, for leaf i and coin j, $q^{\mathcal{A}}_{j,i}$ is the expectation (over p drawn from $\mathcal{A}$) of p to the exponent of the number of "heads" edges on the path from the root to node i in the decision tree, times $(1-p)$

to the exponent of the number of "tails" edges on this path.

Using this notation, the probability of the algorithm reaching leaf i, when the coins are sampled from distribution $\mathcal{A}$, would be $\prod_{\text{coin } j} q^{\mathcal{A}}_{j,i}$, since each coin is sampled from $\mathcal{A}$ independently; let $\prod_{\text{coin } j} q^{\mathcal{B}}_{j,i}$ be the respective probability for sampling from $\mathcal{B}$.

Since the total probability of reaching all leaves i must equal 1, this expression yields the immediate corollary, that for any distribution $\mathcal{A}$ over $[0,1]$,

$$\sum_{\text{leaf } i}\prod_{\text{coin } j} q^{\mathcal{A}}_{j,i} = 1 \tag{5.2}$$

We can now express the squared Hellinger distance with this notation. For any two distributions $\mathbf{a}$ and $\mathbf{b}$, 1 minus their squared Hellinger distance can be rewritten as $\sum_i \sqrt{a_i b_i}$. In our context, the summation is over leaves i, and thus the squared Hellinger distance between the two scenarios in question is

$$H^2_{\text{full}} = 1 - \sum_{\text{leaf } i}\sqrt{\prod_{\text{coin } j} q^{\mathcal{A}}_{j,i}\prod_{\text{coin } j} q^{\mathcal{B}}_{j,i}} \tag{5.3}$$

Since $q^{\mathcal{A}}_{j,i}$ and $q^{\mathcal{B}}_{j,i}$ are both non-negative, we simplify the summand as

$$\begin{aligned}&\sqrt{\prod_{\text{coin } j} q^{\mathcal{A}}_{j,i}\prod_{\text{coin } j} q^{\mathcal{B}}_{j,i}}\\ &= \left(\prod_{\text{coin } j}\frac{q^{\mathcal{A}}_{j,i}+q^{\mathcal{B}}_{j,i}}{2}\right)\left(\prod_{\text{coin } j}\frac{2\sqrt{q^{\mathcal{A}}_{j,i}q^{\mathcal{B}}_{j,i}}}{q^{\mathcal{A}}_{j,i}+q^{\mathcal{B}}_{j,i}}\right)\\ &\geq \left(\prod_{\text{coin } j}\frac{q^{\mathcal{A}}_{j,i}+q^{\mathcal{B}}_{j,i}}{2}\right)\left[1-\sum_{\text{coin } j}\left(1-\frac{2\sqrt{q^{\mathcal{A}}_{j,i}q^{\mathcal{B}}_{j,i}}}{q^{\mathcal{A}}_{j,i}+q^{\mathcal{B}}_{j,i}}\right)\right]\end{aligned} \tag{5.4}$$

where the inequality holds because each $\frac{2\sqrt{q^{\mathcal{A}}_{j,i}q^{\mathcal{B}}_{j,i}}}{q^{\mathcal{A}}_{j,i}+q^{\mathcal{B}}_{j,i}}$ is less than or equal to 1 by the AM-GM inequality (and at least 0), and therefore we can apply the union bound by treating each term as a probability—namely, for any $x_j \in [0,1]$ we have $\prod_j x_j \geq 1 - \sum_j (1-x_j)$.

Observe that our definition of q, being an expectation, is thus linear in the distribution in its superscript, and thus $\frac{1}{2}(q^{\mathcal{A}}_{j,i}+q^{\mathcal{B}}_{j,i}) = q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}$, and therefore the right hand side of the inequality can be rewritten as

$$\left(\prod_{\text{coin } j} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right)\left[1-\sum_{\text{coin } j}\left(1-\frac{\sqrt{q^{\mathcal{A}}_{j,i}q^{\mathcal{B}}_{j,i}}}{q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}}\right)\right] \tag{5.5}$$

Thus the sum of Equation 5.5 over all leaves is at most $1 - H^2_{\text{full}}$. We simplify the summation by changing the summation variable in Equation 5.5 from j to k, and distributing the initial product so as to form three additive terms (the "$j \neq k$" in the bounds of the last product below is because the $j = k$ term gets canceled by the denominator from the last term in Equation 5.5):

$$\begin{aligned}&\left(\sum_{\text{leaf } i}\prod_{\text{coin } j} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right) - \sum_{\text{coin } k}\left(\sum_{\text{leaf } i}\prod_{\text{coin } j} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right)\\ &- \sum_{\text{coin } k}\left(\sum_{\text{leaf } i}\left(\prod_{\text{coin } j \neq k} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right)\sqrt{q^{\mathcal{A}}_{k,i}q^{\mathcal{B}}_{k,i}}\right)\end{aligned}$$

We know by Equation 5.2 that $\left(\sum_{\text{leaf } i}\prod_{\text{coin } j} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right) = 1$, and so the sum can be written as

$$1 - \sum_{\text{coin } k}\left(1 - \sum_{\text{leaf } i}\left(\prod_{\text{coin } j \neq k} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right)\sqrt{q^{\mathcal{A}}_{k,i}q^{\mathcal{B}}_{k,i}}\right)$$

which by definition of H_k is equal to $1 - \sum_{\text{coin } k} H^2_k$: by Equation 5.3, 1 minus the squared Hellinger distance between the view of the algorithm when the k^{th} coin is from $\mathcal{A}$ versus from $\mathcal{B}$, where all remaining coins are drawn from the mixture $\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}$ equals $\sum_{\text{leaf } i}\left(\prod_{\text{coin } j \neq k} q^{\frac{\mathcal{A}}{2}+\frac{\mathcal{B}}{2}}_{j,i}\right)\sqrt{q^{\mathcal{A}}_{k,i}q^{\mathcal{B}}_{k,i}}$.

Summarizing, we have shown that $1 - H^2_{\text{full}} \geq 1 - \sum_{\text{coin } k} H^2_k$, from which the lemma statement follows. □

We now give the KL-divergence decomposition lemma (Lemma 5.2) which will yield a tight *high probability* sample complexity lower bound, but makes a further assumption than Lemma 5.1 that the two coin populations are close to each other. As a note, the definition of H^2_i is slightly different in this lemma from the definition in Lemma 5.1, and is not a typographical mistake.

LEMMA 5.2. *Consider an arbitrary algorithm that iteratively flips coins from a collection of coins, choosing each subsequent coin to flip in an arbitrary adaptive manner based on the results of previous flips. Suppose the algorithm terminates almost surely. Consider two arbitrary mixtures of coins, denoted by distributions $\mathcal{A}$ and $\mathcal{B}$ over the coin bias $[0,1]$. Let D_{full} be the KL-divergence between the transcript of a single run of the algorithm where 1) the coins are drawn from the mixture $\rho\mathcal{A}+(1-\rho)\mathcal{B}$ versus where 2) the coins are drawn from $(\rho+\epsilon)\mathcal{A}+(1-\rho-\epsilon)\mathcal{B}$, where $\rho \in [0,\frac{1}{2})$, $\epsilon \in (0, 1-2\rho]$ and $\epsilon < \rho$. Furthermore, let H^2_i be the squared Hellinger distance between the two scenarios, but instead of running* *the algorithm as is, we only use coin i (as drawn either from the ρ-fraction mixture or the $(\rho+\epsilon)$-fraction mixture depending on the scenario) and simulate all other coins as independent coins drawn from the ρ-fraction mixture. Then*

$$D_{\text{full}} = O\left(\sum_{\text{coin } i} H_i^2\right)$$

The proof of Lemma 5.2 is similar to that of Lemma 5.1 by viewing algorithms as decision trees, with the crucial difference that, rather than using the AM-GM inequality, Lemma 5.2 instead bounds the KL-divergence via a quadratic bound $\log(1+x) \geq x - x^2$, valid for $x \in [-\frac{1}{2}, 1]$.

For the lower bound proof at hand, we show Corollary 5.1 in the next subsection, which upper bounds the squared Hellinger distance for single-coin adaptive algorithms by a quantity that is proportional to the expected number of samples taken by the algorithm.

Corollary 5.1. *Consider an arbitrary single-coin adaptive algorithm. Let H^2 be the squared Hellinger distance between a single run of the algorithm where 1) a coin with bias $\frac{1}{2}+\Delta$ is used with probability ρ and a coin with bias $\frac{1}{2}-\Delta$ is used otherwise, versus a run of the algorithm where 2) a coin with bias $\frac{1}{2}+\Delta$ is used with probability $\rho+\epsilon$ and a coin with bias $\frac{1}{2}-\Delta$ is used otherwise. Furthermore, let $\mathbb{E}_\rho[n]$ and $\mathbb{E}_{\rho+\frac{\epsilon}{2}}[n]$ be the expected number of coin flips during a run of the algorithm where we use a $\frac{1}{2}+\Delta$ coin with probability ρ and $\rho+\frac{\epsilon}{2}$ respectively, and a $\frac{1}{2}-\Delta$ coin otherwise. If all of ρ, ϵ, Δ and ϵ/ρ are smaller than some universal absolute constant, then*

$$\max\left[\frac{H^2}{\mathbb{E}_\rho[n]}, \frac{H^2}{\mathbb{E}_{\rho+\frac{\epsilon}{2}}[n]}\right] = O\left(\frac{\epsilon^2\Delta^2}{\rho}\right)$$

Using Corollary 5.1 and Lemma 5.2, we now complete the proof of the main high probability indistinguishability result (Theorem 1.3) for fully-adaptive algorithms. We note again that Lemma 5.1, which is applicable to more general coin populations with fewer restrictions than Lemma 5.2, can be used to derive a constant probability sample complexity lower bound with essentially the same proof as follows, with the exception that we would use the Hellinger distance inequality in Fact 5.1 instead of the high-probability Pinsker inequality.

Proof. [Theorem 1.3]
Letting $\mathcal{A}$ of be a population of coins that all have $\frac{1}{2}+\Delta$ probability, with $\mathcal{B}$ a population of coins that all have $\frac{1}{2}-\Delta$ probability, our goal is to show the indistinguishability of $\rho\mathcal{A}+(1-\rho)\mathcal{B}$ from $(\rho+\epsilon)\mathcal{A}+(1-\rho-\epsilon)\mathcal{B}$. We apply Lemma 5.2 and use the lemma's conclusion, that $D_{\text{full}} = O\left(\sum_{\text{coin } i} H_i^2\right)$.

Next, for each i, the quantity H_i^2 of Lemma 5.2 describes the squared Hellinger distance between an induced *single-coin* algorithm run on a single coin from scenario A versus B respectively (with the remaining coins being simulated, from scenario A with a ρ-fraction mixture). We thus bound H_i^2 from Corollary 5.1. As in the corollary, let $\mathbb{E}_{i,\rho}[n]$ denote the expected number of samples from coin i when running the induced algorithm (for coin i) on a mixture that uses a $\frac{1}{2}+\Delta$ coin with probability ρ and a $\frac{1}{2}-\Delta$ coin otherwise. Thus Corollary 5.1 yields that $H_i^2 = O(\frac{\epsilon^2\Delta^2}{\rho})\cdot\mathbb{E}_{i,\rho}[n]$. Summing, combined with the result from Lemma 5.2 above, yields

$$D_{\text{full}} \leq O\left(\frac{\epsilon^2\Delta^2}{\rho}\right)\cdot\sum_{\text{coin } i}\mathbb{E}_{i,\rho}[n]$$

Crucially, the sum (over choice of coin i) of the expected number of flips $\mathbb{E}_{i,\rho}[n]$ (when running the algorithm induced for coin i) can be viewed in a different way: the i^{th} term is exactly the expected number of times that coin i is flipped when running the overall algorithm where every coin is drawn from the ρ-fraction mixture in scenario A. Namely, this sum counts the total expected number of coin flips (across all coins i), for the algorithm run in the setting where all coins are drawn from the ρ-fraction mixture. Thus, for an algorithm that uses $o(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta})$ flips in expectation, we conclude that

$$D_{\text{full}} \leq O\left(\frac{\epsilon^2\Delta^2}{\rho}\right)\cdot o\left(\frac{\rho}{\epsilon^2\Delta^2}\log\frac{1}{\delta}\right) = o\left(\log\frac{1}{\delta}\right)$$

We conclude by using the high-probability Pinsker inequality. In the notation of the inequality, given an algorithm that attempts to classify whether it is in scenario A or B, let P, Q respectively be the distributions of its output in scenarios A, B respectively; let E be the event that the algorithm outputs "scenario B". Then the probability that the algorithm is wrong is $P(E)+Q(\bar{E})$. By the high-probability Pinsker inequality this failure probability is at least $\frac{1}{2}e^{-D_{\text{KL}}(P||Q)} \geq \frac{1}{2}e^{-D_{\text{full}}} \geq \frac{1}{2}e^{-o(\log\frac{1}{\delta})} = \frac{1}{2}\delta^{o(1)} \gg \delta$ as desired, where the first inequality is the data processing inequality for KL-divergence. $\square$

5.2 Upper Bounding the Squared Hellinger Distance for Single-Coin Adaptive Algorithms

In this section, we prove Corollary 5.1, though significant technical details are deferred to the full paper [22]. Explicitly, we analyze a simplified scenario in Proposition 5.1, after discussing why each of the simplifying assumptions does not give up generality, and cannot affect the key "squared Hellinger distance per sample" quantity by more than a constant factor.

1. Consider a single-coin algorithm. We restrict our attention to algorithms that only stop once they have seen a number of coin flips that is exactly a power of 2. Any stopping rule S that potentially stops in between powers of 2 could be converted into an almost-equivalent rule S' by collecting coin flips up to the next power of 2 and discarding them as necessary so as to simulate S: this will sacrifice at most a factor of 2 in sample complexity, and can only increase our Hellinger distance (since discarding data is a form of "data processing" and thus we may apply the data processing inequality). Thus the new S' will have "squared Hellinger distance per sample" at least half that of S.

2. By standard symmetrization arguments, a single-coin algorithm can always be implemented such that decisions only depend on the *number* of flips for a coin as well as the *number of observed "heads"*, as opposed to the explicit *sequence* of heads/tails observations. Thus we restrict our attention to stopping rules in the sense of Algorithm 6, specified in full generality by a triangle of stopping coefficients $\{\gamma_{n,k}\}$.

3. There is in some sense a "phase change" once an algorithm has received $\Omega(\frac{1}{\Delta^2})$ samples from a single coin: after this point, the algorithm might have good information about whether the coin is of type $\frac{1}{2}+\Delta$ versus type $\frac{1}{2}-\Delta$, and might productively make subtle adaptive decisions after this point. We restrict our analysis to the regime where *no coin is flipped more than* $10^{-8}/\Delta^2$ *times*: formally, we show an impossibility result in the following stronger setting, where we assume that whenever a single coin is flipped $10^{-8}/\Delta^2$ times, then the coin's true bias (either $\frac{1}{2}+\Delta$ or $\frac{1}{2}-\Delta$) is *immediately* revealed to the algorithm. Thus any coin flips beyond $10^{-8}/\Delta^2$ that an algorithm desires can instead be simulated at no cost.

   Formally, an impossibility result in this setting with "advice" (Proposition 5.1) implies the analogous result in the original setting (Corollary 5.1) by the data processing inequality for Hellinger distance (since Hellinger distance is an f-divergence): simulating additional coin flips in terms of "advice" is itself "data processing", and thus can only decrease the Hellinger distance. Thus the setting without advice has smaller-or-equal Hellinger distance, and uses greater-or-equal number of samples, and hence the bound on their ratio in Proposition 5.1 implies the corresponding bound in Corollary 5.1.

PROPOSITION 5.1. *Consider an arbitrary stopping rule $\{\gamma_{n,k}\}$ that 1) is non-zero only for n that are powers of 2, and 2) $\gamma_{10^{-8}/\Delta^2,k}=1$ for all k, that is the random walk always stops if it reaches $10^{-8}/\Delta^2$ coin flips. Suppose that given a coin, after a random walk on the Pascal triangle according to the stopping rule, the position (n,k) that the walk ended at is always revealed, and furthermore, if $n=10^{-8}/\Delta^2$, then the bias of the coin is also revealed. Let H^2 be the squared Hellinger distance between a single run of the above process where 1) a coin with bias $\frac{1}{2}+\Delta$ is used with probability ρ and a coin with bias $\frac{1}{2}-\Delta$ is used otherwise versus 2) a coin with bias $\frac{1}{2}+\Delta$ is used with probability $\rho+\epsilon$ and a coin with bias $\frac{1}{2}-\Delta$ is used otherwise. Furthermore, let $\mathbb{E}_\rho[n]$ and $\mathbb{E}_{\rho+\frac{\epsilon}{2}}[n]$ be the expected number of coin flips during a run of the algorithm where we use a $\frac{1}{2}+\Delta$ coin with probability ρ and $\rho+\frac{\epsilon}{2}$ respectively, and a $\frac{1}{2}-\Delta$ coin otherwise. If all of ρ, ϵ, Δ and ϵ/ρ are smaller than some universal absolute constant, then*

$$\max\left[\frac{H^2}{\mathbb{E}_\rho[n]},\frac{H^2}{\mathbb{E}_{\rho+\frac{\epsilon}{2}}[n]}\right]=O\left(\frac{\epsilon^2\Delta^2}{\rho}\right)$$

It remains to prove Proposition 5.1. For the rest of the section, we shall use the notation $h^+_{n,k}=(\frac{1}{2}+\Delta)^k(\frac{1}{2}-\Delta)^{n-k}$ and $h^-_{n,k}=(\frac{1}{2}-\Delta)^k(\frac{1}{2}+\Delta)^{n-k}$ for convenience. The proofs for upper bounding $\frac{H^2}{\mathbb{E}_\rho[n]}$ and $\frac{H^2}{\mathbb{E}_{\rho+\frac{\epsilon}{2}}[n]}$ are essentially the same, and here we give a high-level description for bounding the former.

The first step in the proof is the following lemma that writes out the squared Hellinger distance induced by a given stopping rule $\{\gamma_{n,k}\}$, whose proof can be found in the full paper. The expression in Lemma 5.3 avoids square roots and in other ways simplifies aspects of the squared Hellinger distance by estimating terms to within a constant factor, which is folded into a multiplicative "big-Θ" term at the start of the expression. The two lines in the expression below capture the different forms of the Hellinger distance for stopping *before* the last row versus *at* the last row—recall that we prove impossibility under the stronger model where, upon reaching the last row the algorithm receives the true bias of the coin (as "advice"). Thus the squared Hellinger distance coefficients from elements of the last row are typically much larger than for other rows, capturing the cases when this advice is valuable. Recall from Definition 4.1 that $\{\alpha_{n,k}\}$ is defined from the stopping rule $\{\gamma_{n,k}\}$, so that when multiplied by $h^+_{n,k}$ or $h^-_{n,k}$ respectively, it equals the probability of encountering (n,k) without necessarily stopping there, in the cases of positive and negative bias respectively.

LEMMA 5.3. *Consider the two probability distributions in Proposition 5.1 over locations (n,k) in the Pascal triangle of depth $10^{-8}/\Delta^2$ and bias $p \in \{\frac{1}{2} \pm \Delta\}$, generated by the given stopping rule $\{\gamma_{n,k}\}$ in the two cases 1) a coin with bias $\frac{1}{2}+\Delta$ is used with probability ρ and a coin with bias $\frac{1}{2}-\Delta$ is used otherwise versus 2) a coin with bias $\frac{1}{2}+\Delta$ is used with probability $\rho+\epsilon$ and a coin with bias $\frac{1}{2}-\Delta$ is used otherwise. If ϵ/ρ is smaller than some universal constant, then the squared Hellinger distance between these two distributions can be written as*

$$\Theta(\epsilon^2)\cdot \left[\sum_{\substack{n<\frac{10^{-8}}{\Delta^2}\\ k\in[0..n]}} \alpha_{n,k}(\rho h^+_{n,k}+(1-\rho)h^-_{n,k})\frac{(h^+_{n,k}-h^-_{n,k})^2}{(\rho h^+_{n,k}+(1-\rho)h^-_{n,k})^2} + \sum_{\substack{n=\frac{10^{-8}}{\Delta^2}\\ k\in[0..n]}} \alpha_{n,k}(\rho h^+_{n,k}+(1-\rho)h^-_{n,k})\frac{\frac{h^+_{n,k}}{\rho}+h^-_{n,k}}{\rho h^+_{n,k}+(1-\rho)h^-_{n,k}}\right]$$

Intuitively, Lemma 5.3 breaks up the squared Hellinger distance into its contributions from each location (n,k) in the triangle, with the coefficient $\alpha_{n,k}$ depending on the stopping rule (proportional to the algorithm's probability of stopping at location (n,k)), and the remaining portion of the expression depending only on n,k,Δ,ρ, with the ϵ dependence already factored out in the initial $\Theta(\epsilon^2)$ term.

The rest of the analysis uses the above tools to upper bound the squared Hellinger distance per sample. For concrete details and calculations, please refer to full paper [22]. The high level idea of the analysis is to split the expression of Lemma 5.3 for the total squared Hellinger distance per sample into three components, with the contribution from each location (n,k) assigned to either 1) the last row $n=\frac{10^{-8}}{\Delta^2}$, 2) a "high discrepancy region" where $h^+_{n,k}/h^-_{n,k} \geq 1/\rho^{0.1}$ which is towards the right of the triangle, potentially contributing large amounts to the squared Hellinger distance and 3) a "central" region that is the rest of the triangle. The last row, because of the nature of "advice", clearly needs its own analysis. As for the rest of the triangle, we divide it into the "central" and "high discrepancy" regions, and bound their contributions to the squared Hellinger distance per sample using different strategies. For the central region, the key insight is that the squared Hellinger distance term is bounded by a well-behaved quadratic function in that region. On the other hand, for the high discrepancy region, the key observation is that the region is defined such that it is a large number of standard deviations away from where a non-stopping random walk on the Pascal triangle should concentrate, and thus it is very unlikely for the algorithm to enter that region. We take additional care to show that, for any stopping rule used by any algorithm, it *cannot* sufficiently skew the distribution of where the walk ends up—for example, while the distribution might skew to the right if the algorithm stops whenever it enters the "left" side of the triangle, we show that this cannot significantly save on expected sample complexity nor substantially increase the squared Hellinger distance per sample. The analysis for the high discrepancy region makes crucial use of our simplifying assumption that the stopping rule *only* stops at powers of 2 coin flips, letting us analyze large sequences of coin flips at a time, where we may take advantage of the tight concentration of the Binomial distribution over sufficiently many coin flips to bound the effect of any skewing-towards-the-right that can be introduced by the stopping rule.

In the full paper [22], we show that for each of the respective regions, their contribution to the squared Hellinger distance, divided by the expected sample complexity, is at most $O(\epsilon^2\Delta^2/\rho)$. Summing up the three terms is an upper bound on the *total* squared Hellinger distance per expected sample of $O(\epsilon^2\Delta^2/\rho)$, completing the proof of Proposition 5.1.

6 Experimental Results

We give simulation results to demonstrate the practical efficacy of our proposed algorithm. In our experimental setups, we compare the convergence rates of 1) our algorithm ("T-WALK (15)" on the plots), 2) the natural majority vote method mentioned in the Introduction ("VOTING" on the plots) and 3) the "SWITCH" method proposed in previous work by Chung et al. [15] which has been observed to perform well in practice, but does not have a theoretical analysis. For our algorithm, we choose the maximum number of flips for a single coin to be 15 ($= c\log\frac{1}{\epsilon}$) in Algorithm 1. We also make the assumption that the noise parameter satisfies $\Delta \geq 0.3$, meaning that we can use Algorithm 1 directly instead of using Algorithm 2 to simulate virtual coins before feeding them into Algorithm 1. To further improve the practical performance of Algorithm 1, we ran a local search method to improve on the non-zero output coefficients in Step 3(d) of Algorithm 1, using the assumption that $\Delta \geq 0.3$. Concretely, recall that we output a non-zero coefficient when the maximum number of coin flips (15) has occurred and the majority of coin flips has been heads. Thus for $k \in \{8,\ldots,15\}$, we output 8: 0, 9: 6.913, 10: 5.032, 11: 2.101, 12: 0.636, 13: 1.965, 14: 1.016, 15: 1.009. We note again

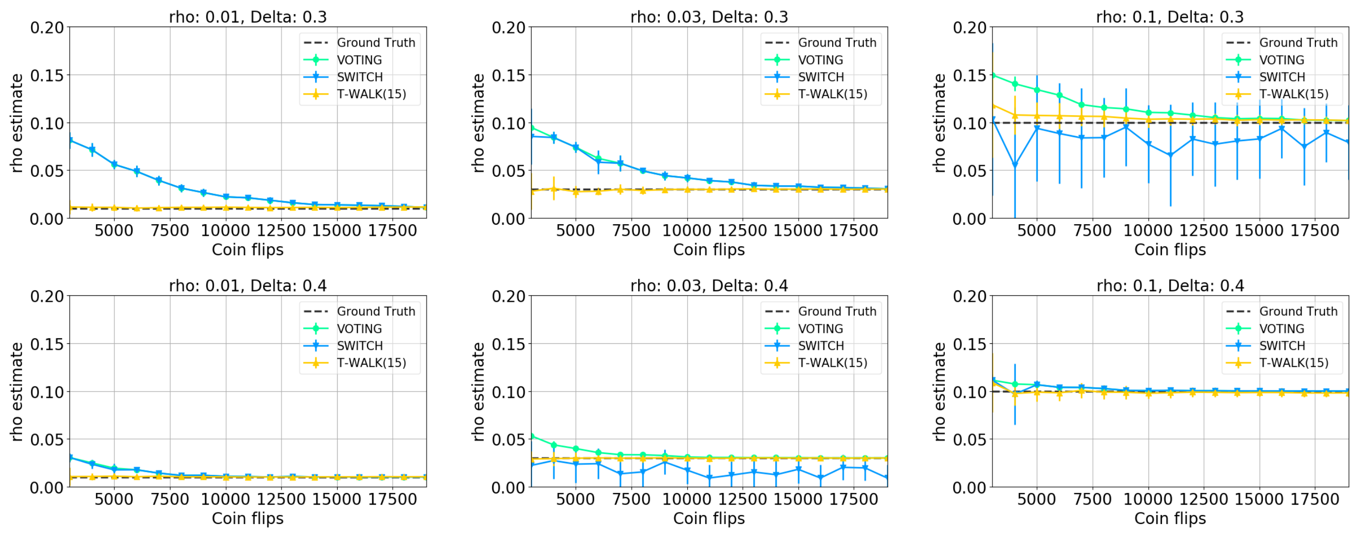

Figure 1: Experimental Results

that these coefficients are *reusable* in practice, as long as the $\Delta \geq 0.3$ assumption can be made.

For a more detailed discussion concerning the implementation of Algorithm 1 in practice, please refer to our full paper [22].

Figure 1 presents the experimental results, for "coin quality" $\Delta = 0.3$ or 0.4, and ground truth fraction of positive coins ρ taking representative values $0.01, 0.03$, or 0.1. For each plot, the x-axis corresponds to the number of coin flips, with all algorithms eventually converging to the ground truth for enough coin flips. Standard deviation bars are computed over 10 runs of each different setting.

In all cases, our algorithm (plotted in yellow) performs close to the ground truth (horizontal black line), while the alternative algorithms take longer to converge, or have high variance, as depicted by the error bars. In particular, as discussed in the introduction, our adaptive methods have the most potential for improvement in the more challenging and more practical regime where ρ is small (left plots), and where Δ is smaller (top row).

References

[1] Jayadev Acharya, Clément L Canonne, and Gautam Kamath. A chasm between identity and equivalence testing with conditional queries. *arXiv preprint arXiv:1411.7346*, 2014.

[2] Tugkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM J. Comput*, 35(1):132–150, 2005.

[3] Graham Bell, Martin J Lechowicz, and Marcia J Waterway. Environmental heterogeneity and species diversity of forest sedges. *Journal of Ecology*, 88(1):67–87, 2000.

[4] Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proc. STOC '16*, pages 1021–1032, 2016.

[5] Mark Braverman, Ankit Garg, Tengyu Ma, Huy L. Nguyen, and David P. Woodruff. Communication lower bounds for statistical estimation problems via a distributed data processing inequality. In *Proc. STOC'16*, page 1011–1020, 2016.

[6] Jennifer Brennan, Ramya Korlakai Vinayak, and Kevin Jamieson. Estimating the number and effect sizes of non-null hypotheses. In *Proc. ICML'20*, 2020.

[7] Clément Canonne. A Survey on Distribution Testing. Your Data is Big. But is it Blue? 2017.

[8] Clément Canonne, Dana Ron, and Rocco A Servedio. Testing equivalence between distributions using conditional samples. In *Proc. SODA '14*, pages 1174–1192, 2014.

[9] Clément Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *Proc. ICALP '14*, pages 283–295, 2014.

[10] Clément L Canonne. Big data on the rise? In *Proc. ICALP '15*, pages 294–305, 2015.

[11] Clément L Canonne, Dana Ron, and Rocco A Servedio. Testing probability distributions using conditional samples. *SIAM J. Comput*, 44(3):540–616, 2015.

[12] Sourav Chakraborty, Eldar Fischer, Yonatan Goldhirsh, and Arie Matsliah. On the power of conditional samples in distribution testing. *SIAM J. Comput*, 45(4):1261–1296, 2016.

[13] Karthekeyan Chandrasekaran and Richard Karp. Finding a most biased coin with fewest flips. In *Proc. COLT'14*, pages 394–407, 2014.

[14] Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond Talagrand Functions: New Lower Bounds for Testing

Monotonicity and Unateness. In *Proc. STOC '17*, pages 523–536, 2017.

[15] Yeounoh Chung, Sanjay Krishnan, and Tim Kraska. A Data Quality Metric (DQM): How to Estimate the Number of Undetected Errors in Data Sets. *Proc. VLDB '17*, 10(10):1094–1105, 2017.

[16] Robert K Colwell and Jonathan A Coddington. Estimating terrestrial biodiversity through extrapolation. *Phil. Trans. R. Soc. Lond. B*, 345(1311):101–118, 1994.

[17] Moein Falahatgar, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. Maxing and ranking with few assumptions. In *Proc. NeurIPS '17*, pages 7060–7070, 2017.

[18] Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Faster algorithms for testing under conditional sampling. In *Proc. COLT '15*, pages 607–636, 2015.

[19] Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. In *Proc. ICML '17*, pages 1088–1096, 2017.

[20] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *Proc. SODA '06*, pages 733–742, 2006.

[21] Kevin Jamieson, Daniel Haas, and Ben Recht. The power of adaptivity in identifying statistical alternatives. In *Proc. NIPS'16*, pages 775–783, 2016.

[22] Jasper C.H. Lee and Paul Valiant. Uncertainty about uncertainty: Optimal adaptive algorithms for estimating mixtures of unknown coins. *arXiv:1904.09228*, 2020.

[23] Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing properties of collections of distributions. *Theory of Computing*, 9(1):295–347, 2013.

[24] Reut Levi, Dana Ron, and Ronitt Rubinfeld. Testing similar means. *SIAM Journal on Discrete Mathematics*, 28(4):1699–1724, 2014.

[25] Frederic M Lord. A strong true-score theory, with applications. *Psychometrika*, 30(3):239–270, 1965.

[26] Frederic M Lord and Noel Cressie. An empirical bayes procedure for finding an interval estimate. *Sankhyā: The Indian Journal of Statistics, Series B*, pages 1–9, 1975.

[27] Matthew L Malloy, Gongguo Tang, and Robert D Nowak. Quickest search for a rare distribution. In *Proc. CISS'12*, pages 1–6, 2012.

[28] Wayne J Millar. Distribution of body weight and height: comparison of estimates based on self-reported and observed measures. *Journal of Epidemiology & Community Health*, 40(4):319–323, 1986.

[29] Michael W Palmer and Philip M Dixon. Small-scale environmental heterogeneity and the analysis of species distributions along gradients. *Journal of Vegetation Science*, 1(1):57–65, 1990.

[30] Ronitt Rubinfeld and Rocco A Servedio. Testing monotone high-dimensional distributions. *Random Struct Algor*, 34(1):24–44, 2009.

[31] N. B. Shah, S. Balakrishnan, and M. J. Wainwright. Feeling the bern: Adaptive estimators for bernoulli probabilities of pairwise comparisons. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1153–1157, 2016.

[32] Nihar Shah, Dengyong Zhou, and Yuval Peres. Approval voting and incentives in crowdsourcing. In *Proc. ICML '15*, pages 10–19, 2015.

[33] Nihar B Shah and Martin J Wainwright. Simple, robust and optimal ranking from pairwise comparisons. *JMLR*, 18(1):7246–7283, 2017.

[34] Kevin Tian, Weihao Kong, and Gregory Valiant. Learning populations of parameters. In *Proc. NeurIPS '17*, pages 5778–5787, 2017.

[35] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008.

[36] Gregory Valiant and Paul Valiant. The power of linear estimators. In *Proc. FOCS '11*, pages 403–412, 2011.

[37] Ramya Korlakai Vinayak, Weihao Kong, Gregory Valiant, and Sham Kakade. Maximum likelihood estimation for learning populations of parameters. In *International Conference on Machine Learning*, pages 6448–6457, 2019.

[38] Jingyan Wang and Nihar B Shah. Your 2 is my 1, your 3 is my 9: Handling arbitrary miscalibrations in ratings. In *Proc. AAMAS '19*, 2019. To appear.