TWO PROBLEMS IN NOISE TOLERANT COMPUTING

BY SIJIAN TANG

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Mathematics

Written under the direction of

Michael Saks

and approved by

New Brunswick, New Jersey

October, 2018

ABSTRACT OF THE DISSERTATION

Two Problems in Noise Tolerant Computing

by Sijian Tang

Dissertation Director: Michael Saks

This thesis consists of 2 main results about computations under random noise. In both problems we consider the discrete input picked from the hamming cube $\{0,1\}^n$. Noise is introduced by flipping each input bit randomly with some fixed probability.

In Chapter 2 we provide the first polynomial algorithm for noisy population recovery problem with finite support. This result directly implies a reverse Bonami-Beckner type inequality for sparse functions.

In Chapter 3 we study the noisy broadcast model and the generalized noisy decision tree (gnd-tree) model under noise cancellation adversary. Here noise cancellation adversary is a type of adversary that can correct the random noise. Under the noise cancellation adversary, we show an $\Omega(\varepsilon^5 \cdot n \log n)$ lower bound for the function OR in the non-adaptive gnd-tree model. This implies an $\Omega(\log(1/\varepsilon)^{-1} \cdot n \log\log n)$ lower bound for a special kind of noisy broadcast model which we call the 2-Phase noisy broadcast model.

Acknowledgements

I would first like to thank my advisor, Michael Saks. His constant encouragement and brilliant insights always inspire me towards the goal. This thesis could never be accomplished without his support and guidance, from the topic of the thesis, all the way to the details of every proof. It was a great honor to work with him. I will never be able to thank him enough for all the help he gave me on this project.

I also want to thank our coauthors, Anindya De and Noga Zewi. It is a great experience to work with you and I really appreciate all your contributions.

Finally, I would like to thank my parents and my girlfriend Ruiheng for always believing in me and supporting me. Thanks for all your love!

Chapter 2 of this dissertation was a joint work with Anindya De and Michael Saks and is adapted from a conference version [10]. Chapter 3 was a joint work with Noga Zewi and Michael Saks.

The work in this thesis was supported in part by NSF grant CCF-1218711 and by Simons Foundation award 332622

Table of Contents

Chapter 1

Introduction

In this section we explain a bit more details on the problems and results we get in each chapter.

In Chapter 2 we study the noisy population recovery problem introduced by Dvir et al. [14]. The goal is to learn an unknown distribution f on binary strings of length n from noisy samples. A noisy sample with parameter $\mu \in [0, 1]$ is generated by selecting a sample from f, and independently flipping each coordinate of the sample with probability $(1 - \mu)/2$. The goal is to estimate the probability of any string to within some given error ε. It is known that the algorithmic complexity and sample complexity of this problem are polynomially related to each other.

In Chapter 2 we consider the special case where the size of the support of the distribution is upper bounded by a known parameter k. (A recent result [12] shows that the general problem (without a bound on the size of the support) has a quasi-exponential lower bound in terms of n) We describe an algorithm that for each $\mu > 0$, provides the desired estimate of the distribution in time bounded by a polynomial in k, n and $1/\varepsilon$ improving upon the previous best result of $\mathsf{poly}(k^{\log \log k}, n, 1/\varepsilon)$ due to Lovett and Zhang [31].

Our proof combines ideas from [31] with a *noise attenuated* version of Möbius inversion. The latter crucially uses the *robust local inverse* construction of Moitra and Saks [32].

Chapter 2 was joint work with Anindya De and Michael Saks.

In Chapter 3 we study the generalized noisy decision tree (gnd-tree) model under noise cancellation adversary. Where the generalized noisy decision tree model, introduced by [24], is a computation model for boolean functions. In this model, instead of getting the true input, the algorithm only gets a collection of noisy copies of the true input, which is obtained by flipping each coordinate independently with probability ε. Computation in the model is a decision tree, where at each node the algorithm evaluates and branches on the value of an arbitrary 2-valued boolean function of one of the noisy copies. This generalizes the noisy decision tree model, in which the function at each node is just the value of a single bit in the noisy copy. The noise cancellation adversary, which was first raised by Feige and Kilian [20], is a type of adversary that can correct the random noise.

We proved that if we restrict the gnd-tree to be non-adaptive, then any such tree that computes the function OR under any noise cancellation adversaries must have depth $\Omega(\varepsilon^5 \cdot n \log n)$. While there is a known $O(n)$ upper bound for this function in gnd-tree model without adversary [24], our result gives the first separation between the random noise model and the model under noise cancellation adversary.

This result also implies a similar result in the noisy broadcast model. We prove that for a special kind of noisy broadcast model which we call the 2-Phase noisy broadcast model, any protocol that computes OR under any noise cancellation adversaries must have depth $\Omega(\log(1/\varepsilon)^{-1} \cdot n \log\log n)$.

Chapter 3 was joint work with Noga Zewi and Michael Saks.

Chapter 2

Polynomial Algorithm for Noisy Population Recovery Problem

2.1 Introduction

The population recovery problem was first introduced by Dvir, Rao, Wigderson and Yehudayoff in [14], which related it to the problem of learning DNF from restrictions. The problem, which can be viewed as a noisy unsupervised learning problem, tries to learn an unknown distribution from noisy samples. Formally speaking, assume we have an unknown distribution f over binary strings of length n, and a noise parameter $0 < \mu < 1$. However, instead of getting samples directly from f, a certain random error occurs to each coordinate independently. There are two main error models: the lossy model and the noisy model. In each model, the samples are generated in the following way:

Lossy Sample:

- Choose a string x according to f.
- Replace each coordinate with a question character "?" independently with probability $1 - \mu$.
- output the resulting string $\tilde{x}$.

Noisy Sample:

- Choose a string x according to f.
- Choose a binary string N according to the distribution η_μ in which each coordinate is independently set to 1 with probability $(1 - \mu)/2$.

- Output $x \oplus N$, where $\oplus$ denotes bitwise sum modulo 2.

In both models, given access to these noisy/lossy samples of f and error parameter ε, the goal of the learner is to output an estimate of the function f (denoted by $\tilde{f}$), which it does by specifying the set S of strings for which the estimate is nonzero, and an estimate $\tilde{f}(x)$ for each $x \in S$. The algorithm is said to succeed provided that $|\tilde{f}(x) - f(x)| \leq \varepsilon$ for all $x \in \{0,1\}^n$. If the algorithm succeeds with probability at least $1 - \delta$ we say that it is an (ε, δ)-estimation algorithm for f.

It is easy to see the lossy model is easier than the noisy model since the learner can simulate samples from the noisy model given samples from the lossy model by replacing each '?' by a random bit. In both models, for $\mu = 1$, there is no noise and the problem is easy to solve, whereas for $\mu = 0$, the distribution f cannot be recovered with any number of samples. As μ becomes smaller, the learning problem becomes harder.

The complexity of an algorithm for this problem depends on four parameters, namely, μ, n, ε, $\frac{1}{\delta}$. As usual, the value of δ is not very significant for the complexity; if we have an algorithm that works for $\delta = 1/4$, we can improve it to an arbitrary δ by repeating the algorithm $\log(1/\delta)$ times and assigning to each $x \in \{0,1\}^n$ the median of the estimates of $f(x)$ from the different runs. We generally think of μ and δ as constants and focus on expressing the running time as a function of n and $1/\varepsilon$. An interesting feature of this problem is that the algorithmic complexity of the problem is polynomial in the sample complexity of the problem. This seems to have been first explicitly mentioned in [31] though they refer to [4, 32]. Thus we will only use the algorithmic complexity as our complexity measure here.

A lot of study have been done for both models. For the lossy model, Dvir et al. [14] gave an algorithm with run time polynomial in n and $1/\varepsilon$ provided that $\mu \gtrsim 0.365$. Their analysis was improved by Batman, et al. [4] who showed that the same algorithm is polynomial time for any $\mu > 1 - 1/\sqrt{2} \approx 0.293$. Subsequently, Moitra and Saks [32] gave a polynomial time algorithm for population recovery in the lossy model for any $\mu > 0$.

For the noisy model, De, et al. [12] gave a quasi-exponential lower bound and they also provided an algorithm that meets this bound. However, we may still hope to find efficient algorithms under some reasonable assumptions. One may assume that every element in the support has non-trivial weight. Precisely speaking, we define the following assumption:

> **Lower Bound Assumption** $LBA(k)$: $f(x) \geq 1/k$ for all $x \in \operatorname{supp}(f)$.

Here k will be another parameter for the complexity measure. Usually we think of it as polynomial in n. It is easy to see this assumption directly implies $|\operatorname{supp}(f)| \leq |k|$. Formally speaking, we define:

> **Bounded Support Assumption** $BSA(k)$: the size of $\operatorname{supp}(f)$ is at most k.

Then we know $LBA(k)$ implies $BSA(k)$. Also, it is natural to consider the following assumption:

> **Known Support Assumption** $KSA(k)$: The algorithm knows a set $X \subset \{0,1\}^n$ with $|X| \leq k$ such that $\operatorname{supp}(f) \subseteq X$.

Clearly $KSA(k)$ is stronger than $BSA(k)$, and therefore solving population recovery under $KSA(k)$ can be reduced to solving it under $BSA(k)$. Though $KSA(k)$ and $LBA(k)$ are incomparable, KSA seems like a much stronger assumption. However it is not known whether solving noisy population recovery under $KSA(k)$ can be reduced to solving it under $LBA(k)$. Surprisingly, the reverse is known to hold: Dvir, et al. [14] showed that solving noisy population recovery under $LBA(k)$ can be reduced to solving it under $KSA(k)$. Precisely speaking, they proved the following lemma:

Lemma 2.1.1. *Assume that we have access to a population recovery algorithm $A(k)$ that operates under the assumption $KSA(k)$. Then there is a population recovery algorithm B that operates under the assumption $LBA(k)$ which on size parameter n, makes n calls to algorithm $A(2k)$ one for each size parameter $i \in [n]$ with error parameter* $\min(1/3k, \varepsilon)$*, such that B is correct provided that all calls to A are correct.*

Actually, in [14] Dvir, et al. claim that the algorithm constructed in Lemma 2.1.1 correctly solves noisy population recovery under assumption $BSA(k)$. However, it turns out the proof of correctness they gave requires the stronger assumption $LSA(k)$, and it seems that $BSA(k)$ is not sufficient for the algorithm to work. The correct proof of Lemma 2.1.1 is given in Section 2.8.

Under $KSA(k)$, a few quasi-polynomial algorithms are known. Wigderson and Yehudayoff [42] developed a framework called "partial identification" and used this to give an algorithm that runs in time $\mathsf{poly}(k^{\log k}, n, 1/\varepsilon)$ for any $\mu > 0$. They also showed that their framework cannot obtain algorithms running in time better than $\mathsf{poly}(k^{\log\log k})$.

Lovett and Zhang [31] gave an algorithm under $KSA(k)$ with a better time complexity of $\mathsf{poly}(k^{\log\log k}, n, 1/\varepsilon)$ for any $\mu > 0$. Interestingly, while their algorithm matches the lower bound in [42], their algorithm departs from the framework of [42], and thus is not subject to the same lower bound.

In this article we will focus on finding an efficient algorithm under $KSA(k)$ (which directly gives an algorithm under $LBA(k)$ by Lemma 2.1.1). Under $KSA(k)$, it is easy to see estimating f is equivalent to estimating $f(x)$ for all x in X. We focus on constructing an algorithm that recovers f on a particular point x. Then we can recover f by running the algorithm k times, once for each $x \in X$. Also, by shifting the samples on the hamming cube, without loss of generality we may assume x is the origin $\mathbf{0} = 0^n$. In other words, we consider the following problem:

Noisy Population Point Recovery ($NPPR(n, \varepsilon)$) Given access to noise samples of f in $\{0,1\}^n$ with noise parameter μ, output an estimate of $f(\mathbf{0})$ that has additive error at most ε.

By the analysis above, if we find an algorithm for $NPPR$ with running time $T(n,k)$ under $KSA(k)$, then we can get an algorithm for the original noisy population recovery problem under $KSA(k)$ with running time $T(n,k) \cdot \mathsf{poly}(kn)$. In particular, if $T(n,k) = \mathsf{poly}(k,n)$, then apply Lemma 2.1.1 we get an algorithm for $NPPR$ under $LBA(k)$ that runs in time $\mathsf{poly}(k,n)$.

2.1.1 Our result

Here we show that for any $\mu > 0$, the time complexity of noisy population recovery problem is at most $\mathsf{poly}(k, n, \frac{1}{\varepsilon}, \log(\frac{1}{\delta}))$

Theorem 2.1.2. *For any $\mu > 0$, there is an algorithm for $NPPR(n, \varepsilon)$ under $KSA(k)$, with running time $O\big(n \cdot \big(\frac{k}{\varepsilon}\big)^{O_\mu(1)} \cdot \log(1/\delta)\big)$. Here $O_\mu(1) = \tilde{O}(1/\mu^4)$.*

2.1.2 A reverse Bonami-Beckner Corollary

As with past results on the population recovery problem, our result has interesting functional analytic consequences. The process we are observing generates observations that are obtained by taking a sample from $\{0,1\}^n$ according to the probability distribution f and applying noise independently to each coordinate. Thus, the observed samples come from a distribution that is obtained from f by applying a linear operator T_μ, where for each $x \in \{0,1\}^n$:

$$(T_\mu f)(x) = \mathbb{E}_{N \sim \eta_\mu}[f(x \oplus N)].$$

The operator T_μ is usually referred to in the literature as the Bonami-Beckner operator [6, 5, 25, 37]. Intuitively, T_μ "smooths" f by replacing the value of f at x by a weighted average of values of f near x. One way that this smoothing property is made precise is via *hypercontractive inequalities* [6, 5, 25], which have the following flavor: "(A higher order) norm of $T_\mu f$ can be upper bounded by (a lower order) norm of f", where the bounds are independent of the dimension (number of input variables) of the function.

Given such smoothing theorems, it is natural to try to establish reverse inequalities that assert that some norm of $T_\mu f$ is never much smaller than (the same or different) norm of f. No such dimension independent inequality can hold for all functions, as for parity function $f = (-1)^{\sum_i x_i}$, $T_\mu f$ is exponentially small everywhere. However, such reverse inequalities are possible for restricted classes of functions. For example, Borell [7] proved a reverse Bonami-Beckner inequality which roughly states that for positive valued functions $f : \{0,1\}^n \to \mathbb{R}^+$, the norm of $T_\mu f$ can't be too small if the norm of f is large.

Lovett and Zhang [31] observed that the existence of fast algorithms for population recovery problem for functions satisfying $BSA(k)$ is equivalent to a reverse Bonami-Beckner type inequality for sparse functions. In particular, they showed that for $f : \{0,1\}^n \to \mathbb{R}$, if $\text{supp}(f) = k$, then $\|T_\mu f\|_1 \geq k^{-O_\mu(\log\log k)}\|f\|_1$. The results of the present paper lead to the following improved reverse Bonami-Beckner inequality for sparse functions:

Corollary 2.1.3. *Assume $f : \{0,1\}^n \to \mathbb{R}$ with $|\text{supp}(f)| \leq k$. Then $\|T_\mu f\|_1 \geq k^{-O_\mu(1)}\|f\|_1$, where $O_\mu(1) = \tilde{O}(1/\mu^4)$.*

2.2 Preliminaries

2.2.1 Fourier analysis of Boolean Functions

We begin with some definitions. We write $\mathbf{0}$ for the point 0^n in $\{0,1\}^n$. For $x \in \{0,1\}^n$, $|x|$ is the Hamming weight of x, which is equal to the number of 1's. For binary strings $x, y \in \{0,1\}^n$, $x \oplus y$ denotes the bitwise sum mod 2, and $d_H(x,y) = |x \oplus y|$ is the Hamming distance between x and y, which is the number of positions where x and y differ.

For $Y \subseteq \{0,1\}^n$, $w(Y)$ denotes the maximum Hamming weight of any string in Y.

For a set $S \subseteq [n]$, 2^S denotes the set of subsets of S, $\binom{S}{r}$ denotes the set of subsets of size r and $\binom{S}{\leq r}$ denotes the set of subsets of size at most r. For sets S, T, $S \triangle T$ denotes their symmetric difference $(S - T) \cup (T - S)$.

We define the following sets of functions:

- $\mathcal{F} = \mathcal{F}_n$ is the space of real-valued functions on $\{0,1\}^n$
- $\mathcal{D} = \mathcal{D}_n$ is the set of nonnegative-valued $f \in \mathcal{F}$ satisfying $\sum_{x \in \{0,1\}^n} f(x) = 1$.
- $\mathcal{G}_\eta(X)$: for $X \subseteq \{0,1\}^n$ and $\eta \geq 0$, $\mathcal{G}_\eta(X)$ is the set of $f \in \mathcal{F}$ such that $f(\mathbf{0}) = 1$ and $|f(x)| \leq \eta$ for $x \in X - \{\mathbf{0}\}$.

We view $\mathcal{F}$ as an inner product space with inner product $\langle f, g\rangle = \sum_{x\in\{0,1\}^n} f(x)g(x)$.

For $x \in \{0,1\}^n$, the function $\mathbf{1}_x$ maps x to 1 and all other points to 0, and for $P \subseteq \{0,1\}^n$, $\mathbf{1}_P = \sum_{x\in P} \mathbf{1}_x$.

Functions in $\mathcal{D}$ can be viewed as probability measures on $\{0,1\}^n$. For $f \in \mathcal{D}$ we write $x \sim f$ to mean that x is a random string sampled according to f. The set $\mathcal{D}$ is a convex subset of $\mathcal{F}$ whose extreme points are the functions $\{\mathbf{1}_x : x \in \{0,1\}^n\}$.

For $S \subseteq [n]$, the character $\chi_S \in \mathcal{F}$ is defined by $\chi_S(x) = \prod_{i\in S}(-1)^{x_i}$. The functions $\{\chi_S : S \subseteq [n]\}$ form an orthonormal basis, and are all length $2^{n/2}$ for $\mathcal{F}$. Thus every $f \in \mathcal{F}$ can be written as a linear combination of characters: $f = 2^{-n}\sum_{S\subseteq[n]}\langle f, \chi_S\rangle\chi_S$. The *Fourier coefficient* of function f at $S \subseteq [n]$ is defined by[1] $\widehat{f}(S) = \langle f, \chi_S\rangle = \sum_{x\in\{0,1\}^n} f(x)\chi_S(x)$.

For $f \in \mathcal{D}$ we have:

$$\hat{f}(S) = \mathop{\mathbf{E}}_{x\sim f}[\chi_S(x)]. \tag{2.1}$$

The following equation, known as Plancherel's theorem expresses the inner product of f and g in terms of their Fourier coefficients.

$$\langle f, g\rangle = 2^{-n}\sum_{S\subseteq[n]} \hat{f}(S)\hat{g}(S). \tag{2.2}$$

We define:

- The *support of* f, $\mathrm{supp}(f) = \{x \in \{0,1\}^n : f(x) \neq 0\}$.
- The *Fourier support of* f, $\mathrm{supp}(\hat{f}) = \{S \subseteq [n] : \hat{f}(S) \neq 0\}$
- $\|f\|_1 = \sum_{x\in\{0,1\}^n} |f(x)|$
- $\|\hat{f}\|_{L_1} = 2^{-n}\sum_{S\subseteq[n]} |\hat{f}(S)|$

$\mathcal{F}$ has two natural products. For $f, g \in F$, the *pointwise product* fg is given by $fg(x) = f(x)g(x)$ for all x and the *convolution product* $f * g$ is given by $f * g(x) = \sum_y f(y)g(x \oplus y)$.

[1]In some other articles the Fourier coefficient is defined by $\widehat{f}(S) = 2^{-n}\sum_{x\in\{0,1\}^n} f(x)\chi_S(x)$. Our normalization is chosen so that (2.1) holds.

If $f \in \mathcal{D}$ then $f * g(x) = \mathbb{E}_{z \sim f}[g(x \oplus z)]$. If f and g are both in $\mathcal{D}$ then $f * g \in \mathcal{D}$ and a sample from $f * g$ can be obtained by taking $x \oplus z$ where z is sampled according to f and x is sampled according to g.

For $S \subseteq [n]$, we have:

$$\begin{aligned}\widehat{fg}(S) &= 2^{-n} \sum_{T \subseteq [n]} \hat{f}(T)\hat{g}(T \triangle S) \\ \widehat{f * g}(S) &= \hat{f}(S)\hat{g}(S).\end{aligned}$$

For a linear operator L on $\mathcal{F}$, and norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$, the $\alpha \to \beta$ norm of L, denoted by $\|L\|_{\alpha \to \beta}$ is defined to be the supremum of $\frac{\|Lv\|_\beta}{\|v\|_\alpha}$ over all $v \in V$.

For $S \subseteq [n]$, the operator $X_S : \mathcal{F} \to \mathcal{F}$ is defined as $X_S : f \mapsto \chi_S \cdot f$.

The Bonami-Beckner noise operator T_μ, defined for any real number μ, is most easily defined by its action on the character basis:

$$T_\mu \chi_S = \mu^{|S|} \chi_S.$$

More generally for $U \subseteq [n]$, the operator $T_{\mu,U}$ is defined by:

$$T_{\mu,U} \chi_S = \mu^{|S \cap U|} \chi_S.$$

Thus $T_\mu = T_{\mu,[n]}$. Using linearity, we can extend the action of T_μ to the space of all functions $\mathcal{F}$. For $i \in [n]$ we will adopt the shorthand $T_{\mu,i}$ for $T_{\mu,\{i\}}$.

It is easy to see that for any $\mu \neq 0$, $T_{\mu,U}$ is an invertible operator with its inverse being $T_{1/\mu,U}$. Likewise, for any U, U', the operators $T_{\mu,U}$ and $T_{\mu,U'}$ commute. In fact, if U and U' are disjoint, then $T_{\mu,U} \circ T_{\mu,U'} = T_{\mu,U \cup U'}$. Given the definition of $T_{\mu,U}$, it is straightforward to verify that for $x \in \{0,1\}^n$,

$$T_{\mu,U} f(x) = \sum_{z \in \{0,1\}^n : z_i = 0 \text{ for } i \notin U} f(x \oplus z) \prod_{i \in U} \frac{1}{2}(1 + (-1)^{z_i} \mu).$$

When $\mu \in [-1,1]$, $T_{\mu,U}$ has a nice probabilistic description. For $\mu \in [-1,1]$, define ν_μ to be the probability distribution on $\{0,1\}^n$ obtained by setting each bit to 1 independently with probability $(1-\mu)/2$. More generally, for $U \subseteq [n]$ $\nu_{\mu,U}$ denotes the probability distribution on $\{0,1\}^n$ obtained by setting each of the bits indexed by U independently to 1 with probability $(1-\mu)/2$ and setting all the bits indexed by $[n]\setminus U$ to 0. We then have for $\mu \in [-1,1]$, that

$$T_{\mu,U} f = \nu_{\mu,U} * f,$$

and thus for $x \in \{0,1\}^n$

$$(T_{\mu,U} f)(x) \quad = \quad \mathbb{E}_{z \sim \nu_{\mu,U}}[f(x \oplus z)].$$

One can view this operator as a process to smooth the function by spreading the value on one point to its neighbours. It is easy to verify that if $f \in \mathcal{D}$ and $\mu \in [-1,1]$ then $T_{\mu,U} f \in \mathcal{D}$. A sample from $T_{\mu,U} f$ is generated by taking $x \oplus z$ where $x \sim f$ and $z \sim \nu_{\mu,U}$. A *μ-noisy sample from f* is a sample from $T_\mu f$.

2.2.2 Parameter estimation

We consider the general problem of estimating a real-valued parameter $P = P(g)$ of an unknown probability distribution $g \in \mathcal{D}_n$. An estimator P_{est} is a random variable that is a function of a collection of independent samples.

- The *bias of* P_{est} (as an estimator of P) is $|\,\mathbf{E}[P_{\text{est}} - P]|$.
- The *range of* P_{est} is the maximum of $|P_{\text{est}}|$.
- P_{est} is an (ε,κ)*-estimator* of P provided that $\Pr[|P_{\text{est}} - P| > \varepsilon] < \kappa$.

It is well known that one can build (ε,κ)-estimators from independent copies of estimators wth fairly weak estimation properties. For an estimator P_{est} and positive integer k, let $A_k(P_{\text{est}})$ denote the average of k independent copies of P_{est}.

Proposition 2.2.1. *For any $\varepsilon,\delta \in (0,1)$, if the estimator P_{est} of P has bias at most $\frac{\varepsilon}{2}$, and range at most M, then the estimator $A_k(P_{\text{est}})$, with $k > 8\frac{M^2}{\varepsilon^2}\ln(\frac{1}{\kappa})$ is a (ε,δ)-estimator.*

Proof. Obviously $\mathbf{E}[A_k(P_{\text{est}})] = \mathbf{E}[P_{\text{est}}]$. By the Chernoff-Hoeffding bound [22],

$$\Pr[|A_k(P_{\text{est}}) - P| \geq \varepsilon] \leq \Pr[|A_k(P_{\text{est}}) - \mathbf{E}[P_{\text{est}}]|] \geq \varepsilon/2] \leq e^{-\varepsilon^2 k/8M^2} \leq \delta.$$

□

Möbius transforms

Let $(P, \preceq)$ be a poset. Define function $\zeta_P : P \times P \to \mathbb{R}$ as $\zeta(x, y) = 1$ if and only if $x \preceq y$ and 0 otherwise. Also define $\mu_P : P \times P \to \mathbb{R}$ recursively as follows:

$$\text{For } x \in P,\ \mu_P(x, x) = 1.$$

$$\text{For } x, y \in P,\ \mu_P(x, y) = \mathbf{1}_{x \preceq y} \cdot \Big(\sum_{x \preceq z \prec y} -\mu_P(x, z) \Big).$$

Let $\mathcal{F}_P$ be the space of real-valued functions on P. We define operators $\boldsymbol{\zeta}_P : \mathcal{F}_P \to \mathcal{F}_P$ and $\boldsymbol{\mu}_P : \mathcal{F}_P \to \mathcal{F}_P$ by:

$$(\boldsymbol{\zeta}_P f)(x) = \sum_{x \in P} \zeta(x, y) f(y) = \sum_{x \preceq y} f(y) \text{ and } (\boldsymbol{\mu}_P f)(x) = \sum_{x \preceq y} \mu_P(x, y) \cdot f(y).$$

It is well known (see [41]) that the transforms $\boldsymbol{\zeta}_P$ and $\boldsymbol{\mu}_P$ are inverses of each other. $\boldsymbol{\mu}_P$ is usually referred to as the Möbius transform of the poset P. The above notions can be extended to the more general setting of functions from P to a fixed vector space.

Proposition 2.2.2. *Let P be a poset and V be an arbitrary vector space over $\mathbb{R}$. Suppose $(f_x : x \in P)$ and $(g_x : x \in P)$ are indexed families of vectors in V satisfying $f_x = \sum_{x \preceq y} g_y$. Then*

$$g_x = \sum_{x \preceq y} \mu(x, y) \cdot f_y$$

Definition 1. *For $x \in P$, define $x^{\downarrow} = \{y : y \preceq x\}$. For $C \subseteq P$, define $C^{\downarrow} = \cup_{x \in C} x^{\downarrow}$. A subset D of P such that $D^{\downarrow} = D$ is a "downset". It is easy to see that $C^{\downarrow}$ is the unique minimal subset of P that is a downset and contains C, and is referred to as the "downset generated by C".*

If $C \subseteq P$, then we can view C as a poset, which has its own Möbius function μ_C. In general it is not true that for all $x, y \in C$, $\mu_C(x, y) = \mu_P(x, y)$ but it is true if C is a downset of P.

Proposition 2.2.3. *If $D \subseteq P$ is a downset, then for all $x, y \in D$, $\mu_D(x, y) = \mu_P(x, y)$.*

This is easily verified by induction using the above inductive definition of μ_D and μ_P.

We denote by $\mathcal{P}([n])$ the poset on $2^{[n]}$ ordered by set inclusion. It is well known that in this poset, for $x \preceq y$, $\mu_{\mathcal{P}([n])}(x, y) = (-1)^{|y \setminus x|}$. Combining with Proposition 2.2.3 we have:

Corollary 2.2.4. *If D is a downset of $\mathcal{P}([n])$ then for $x \preceq y \in D$ we have $\mu_D(x, y) = (-1)^{|y \setminus x|}$.*

2.2.3 Technical computational considerations

We now mention a few technical considerations concerning the cost of computation for our algorithm which will be used repeatedly throughout the paper. In some cases, we will have known functions $b, \ell \in \mathcal{F}_n$, given by an $n^{O(1)}$-time algorithm that on input $S \subseteq [n]$ evaluates $\hat{\ell}(S)$ and $\hat{b}(S)$, and we will want to evaluate a function of the form $\sum_{S \subseteq [n]} \hat{\ell}(S)\hat{b}(S)$. The cost of the trivial summation algorithm is $2^n n^{O(1)}$, but if $\text{supp}(\hat{\ell})$ is small compared to 2^n we can hope to speed this up by enumerating only over sets in $\text{supp}(\hat{\ell})$. However, even if we can evaluate $\hat{\ell}(S)$ for any given S, this does not mean that we can enumerate over sets in the support without looking at all sets. Technically what we want is a family of subsets $\mathcal{H}$ that contains $\text{supp}(\hat{\ell})$ together with an *efficient listing algorithm* for $\mathcal{H}$ which is an algorithm that lists all members of $\mathcal{H}$ in time $|\mathcal{H}|n^{O(1)}$. We will say that $\mathcal{H}$ is a *listable support* for $\hat{\ell}$.

Further, for the sake of clarity of exposition, throughout the paper, we will assume that we are able to do basic arithmetic operations on real numbers with infinite precision. In an actual implementation, we will only be working with finite precision approximations of these numbers. The next simple proposition (stated without a proof) asserts that basic arithmetic operations on real numbers can be done efficiently to any finite precision.

Proposition 2.2.5. *A sum of the form $\sum_{i=1}^{m} \prod_{j=1}^{l} B_{ij}$ that satisfies $|B_{ij}| \leq K$ for all i, j can be approximated to within additive error δ in time $\mathsf{poly}(m, l, \log(1/\delta), \log(K))$.*

Proof. In order to approximate the sum within additive error δ, it is enough to approximate each term $\prod_{j=1}^{l} B_{ij}$ within additive error δ/m. Since $|B_{ij}| \leq K$ for all i, j, for each B_{ij}, the additive error for B_{ij} will be scale up by at most K^l times in the product. Thus it is enough to approximate each B_{ij} within additive error $\delta/(m \cdot K^l)$. Thus the precision we need to represent each B_{ij} is $O(\log(K) + \log((m \cdot K^l)/\delta))$ bits. This implies the computation can be done in time $O(m \cdot l \cdot (\log(K) + \log((m \cdot K^l)/\delta))^2) = \mathsf{poly}(m, l, \log(1/\delta), \log(K))$.

□

2.3 Proof of Theorem 2.1.2

We have an unknown probability distribution f on $\{0,1\}^n$ together with a subset X that contains $\mathrm{supp}(f)$. We have access to samples from the distribution $T_\mu f$. Our goal is to give a good estimate for $f(0^n)$ in time $\mathsf{poly}(n, |X|, \frac{1}{\varepsilon}, \log(\frac{1}{\delta}))$. Our algorithm is based on the approach of [31] (which built on ideas from [42]). We present a framework that abstracts this approach, and identify a critical improvement. The key ingredient to our algorithm is a function u that satisfies the conclusions of the following lemma.

Lemma 2.3.1. *Given X and ε, there is a function $u \in \mathcal{F}_n$ such that for all f with $\mathrm{supp}(f) \subseteq X$:*

1. *$u(\mathbf{0}) \in [1/2, 1]$ and there is an algorithm that estimates $u(\mathbf{0})$ to within an additive $\varepsilon/10$ and runs in time $\mathsf{poly}\big(nk\frac{1}{\varepsilon}\log\frac{1}{\kappa}\big)$.*

2. *$|\langle u, f\rangle - u(\mathbf{0})f(\mathbf{0})| \leq \varepsilon/10$.*

3. *There is a real valued function $\alpha(x)$ defined on $\{0,1\}^n$ computable in time $(k/\varepsilon)^{\tilde{O}(1/\mu^4)} n^{O(1)}$ such that (a) For all $x \in \{0,1\}^n$, $|\alpha(x)| \leq (k/\varepsilon)^{\tilde{O}(1/\mu^4)}$, and (b) for $z \sim T_\mu f$, $\alpha(z)$ is an unbiased estimator for $\langle u, f\rangle$.*

Theorem 2.1.2 follows easily from this lemma.

Proof of Theorem 2.1.2. Let $R = (k/\varepsilon)^{\tilde{O}(1/\mu^4)}$ be the range of the estimator $\alpha(x)$. Applying Proposition 2.2.2, the average of $m = \mathsf{poly}(R/\varepsilon) = (k/\varepsilon)^{\tilde{O}(1/\mu^4)}$ independent copies of this estimator yields an estimate A that is within $\varepsilon/10$ of $\langle u, f\rangle$ with probability at least 7/8. Also, let B be the estimate of $u(\mathbf{0})$ given by the third part of the lemma that is within $\varepsilon/10$ with probability at least 7/8. Our algorithm outputs A/B (or, more precisely, a floating point approximation C to A/B that is within $\varepsilon/10$ of A/B) as the estimate of $f(\mathbf{0})$. The bound on the running time of the algorithm follows easily from the bounds on the running time of the estimator for $\langle u, f\rangle$ and computation of $u(\mathbf{0})$.

Next, we claim that with probability at least 3/4, the output C is within ε of $f(\mathbf{0})$. Note that with probability at least 3/4, $|A - \langle u, f\rangle| \le \varepsilon/10$ and $|B - u(\mathbf{0})| \le \varepsilon/10$. Assuming this is the case, we also have, $B \ge 1/3$ since $u(\mathbf{0}) \ge 1/2$ and we can assume that $\varepsilon < 1$. Also by hypothesis we have $|u(\mathbf{0})f(\mathbf{0}) - \langle u, f\rangle| \le \varepsilon/10$. So with probability at least 3/4, we have

$$\begin{aligned}
|f(\mathbf{0}) - C| &\le \frac{\varepsilon}{10} + \left|f(\mathbf{0}) - \frac{A}{B}\right| \\
&= \frac{\varepsilon}{10} + \frac{1}{B}\cdot|Bf(\mathbf{0}) - A| \\
&\le \frac{\varepsilon}{10} + 3|Bf(\mathbf{0}) - A| \\
&\le \frac{\varepsilon}{10} + 3(|Bf(\mathbf{0}) - u(\mathbf{0})f(\mathbf{0})| + |u(\mathbf{0})f(\mathbf{0}) - \langle u, f\rangle| + |\langle u, f\rangle - A|) \\
&\le \frac{\varepsilon}{10} + 3\left(\frac{\varepsilon}{10} + \frac{\varepsilon}{10} + \frac{\varepsilon}{10}\right) \le \varepsilon.
\end{aligned}$$

□

So the main part of the proof of the theorem is the construction of the function u and the proof of the associated Lemma 2.3.1. It turns out that u is best described as the pointwise product of two functions ℓ and q, and in the next section we motivate their construction and state the essential properties of the functions q and ℓ (see Lemmas 2.4.2 and 2.4.1). These properties immediately give Lemma 2.3.1. In Section 2.6 we construct ℓ and show that it satisfies Lemma 2.4.2 and in Section 2.5 we construct q and show that it satisfies Lemma 2.4.1.

2.4 Constructing the function u

2.4.1 Estimating $f(\mathbf{0})$ via estimates of Fourier coefficients

Recall we have an unknown function f with $\text{supp}(f) \subset X$. We have access to samples from $T_\mu f$ and we want to estimate $f(\mathbf{0})$. Suppose ℓ is any member of $\mathcal{F}$ that satisfies

$$f(\mathbf{0}) = \langle \ell, f \rangle. \tag{2.3}$$

By (2.2), this equals $\sum_{S\subseteq[n]}(\hat{\ell}(S)/2^n)\hat{f}(S)$. If we can estimate each of the fourier coefficients $\hat{f}(S)$, we can estimate $f(\mathbf{0})$ by replacing each fourier coefficient in this summation by its estimate.

Indeed, there is a natural unbiased estimator for $\hat{f}(S)$ using samples of $T_\mu(f)$. For any $d \in \mathcal{D}_n$, if z is a sample from $d \in \mathcal{D}_n$, then by (2.1), $\chi_S(z)$ is an unbiased estimator for $\hat{d}(S)$. In particular if $z \sim T_\mu f$ then $\chi_S(z)$ is an unbiased estimator of $\widehat{T_\mu f}(S) = \mu^{|S|}\hat{f}(S)$. Therefore $(\frac{1}{\mu})^{|S|}\chi_S(z)$ is an unbiased estimator for $\widehat{f}(S)$. Thus for ℓ satisfying (2.3),

$$W_\ell(z) = \sum_{S\subseteq[n]} (\hat{\ell}(S)/2^n)\left(\frac{1}{\mu}\right)^{|S|}\chi_S(z),$$

is an unbiased estimator of $f(\mathbf{0})$.

We are free to choose any ℓ satisfying (3). One natural choice for ℓ is $\mathbf{1_0}$. For this choice we have $\hat{\ell}(S) = 1$ for all S, and the resulting estimator of $f(\mathbf{0})$ is $\sum_{S\subseteq[n]}\frac{1}{2^n}\cdot(\frac{1}{\mu})^{|S|}\chi_S(z)$. Unfortunately, this estimator seems to require high sample complexity in order to guarantee small error. To see this, note that $W_\ell(z)$ simplifies to $(1-\frac{1}{\mu})^{|z|}(1+\frac{1}{\mu})^{n-|z|}\cdot 2^{-n}$. Thus, the range of this estimator (and in fact, the variance) is exponentially large in n when μ is small, and we would need exponential in n many samples to reduce the variance to O(1).

So we look for an alternative ℓ satisfying (2.3) for which both the cost of evaluating $W_\ell(z)$, and the range of $W_\ell(z)$ are "small". To guarantee (2.3), since $\text{supp}(f) \subseteq X$, it suffices that ℓ be chosen from $\mathcal{G}_0(X)$ (recall that $\mathcal{G}_0(X)$ is the set of functions that map $\mathbf{0}$ to 1 and all $x \in X \setminus \{\mathbf{0}\}$ to 0). To bound the cost of the induced (ε, δ)-estimator we need to bound both the cost of computing $W_\ell(z)$ and its range.

To compute $W_\ell(z)$ we need to sum $(\hat{\ell}(S)/2^n)(\frac{1}{\mu})^{|S|}\chi_S(z)$ over $S \in \text{supp}(\hat{\ell})$. As discussed in Section 2.2.3, to evaluate this sum quickly it is not enough to know that $|\text{supp}(\hat{\ell})|$ is small; we also need a listable support $\mathcal{H}$ for $\hat{\ell}$. With this, $W_\ell(z)$ can be evaluated in time $|\mathcal{H}|(T + n^{O(1)})$ where T is an upper bound on the time needed to evaluate $(\hat{\ell}(S)/2^n)$ on input $S \in \mathcal{H}$. To upper bound the range of $W_\ell(z)$, note that every term in the sum is bounded (in absolute value) by $(\hat{\ell}(S)/2^n)(\frac{1}{\mu})^{m(\mathcal{H})}$ where $m(\mathcal{H})$ is an upper bound on size of the largest set in $\mathcal{H}$. Thus, the range R of this estimator is bounded by $\|\hat{\ell}\|_{L_1}(\frac{1}{\mu})^{m(\mathcal{H})}$. Hence, the running time of the estimator is $\mathsf{poly}(|\mathcal{H}|, R, \frac{1}{\varepsilon}, \log(1/\delta))$.

The algorithm of Wigderson and Yehudayoff [42] can be formulated in this framework: They (implicitly) show how to (efficiently) construct a function $\ell_{WY} \in \mathcal{G}_0(X)$, and listable support $\mathcal{H}$ for $\hat{\ell}$ so that

- All sets in $\mathcal{H}$ have size at most $O(\log |X|)$.
- $|\mathcal{H}| \leq |X|^{\log |X|}$.
- $\|\widehat{\ell_{\mathsf{WY}}}\|_{L_1} = O(|X|^{\log |X|})$.

Thus the running time of the induced estimator for $f(\mathbf{0})$ is $\mathsf{poly}(|X|^{\log |X|}, n, \frac{1}{\varepsilon}, \log(1/\delta))$.

2.4.2 The Lovett-Zhang approach

The improved running time of Lovett and Zhang [31] involves two steps: (i) Constructing a function ℓ_{LZ} that gives a faster estimator in the case that all of the points in X have small Hamming weight, i.e., $O(\log |X|)$. (ii) A reduction that effectively reduces the case of general X to the small Hamming weight case.

Recall that for $Y \subseteq \{0,1\}^n$, $w(Y)$ is the maximum Hamming weight of any string in Y. Lovett and Zhang showed how to construct, for any set Y, a function $\ell_{\mathsf{LZ}}^Y \in \mathcal{G}_0(Y)$ and a listable support $\mathcal{H}$ for $\widehat{\ell_{\mathsf{LZ}}^Y}$ such that

- $m(\mathcal{H})$, the size of the largest set in $\mathcal{H}$, is at most $w(Y)$.
- $|\mathcal{H}| \leq |Y|2^{w(Y)}$.

- $\|\widehat{\ell^Y_{\mathsf{LZ}}}\|_{L_1} = |Y|^{O(\log w(Y))}$.

(This result is implied by Proposition 3.6 in their paper.) Applying this construction with $Y = X$ yields an estimator for $f(\mathbf{0})$. Unlike the WY estimator, the running time of this estimator deteriorates as $w(X)$ increases. For example, for $w(X) = O(\log |X|)$ the derived estimator has running time is $|X|^{O(\log\log|X|)}$.

Lovett and Zhang present a kind of a reduction of the general case ($w(X) \leq n$) to the case that $w(X) = O(\log |X| \cdot \log\log |X|)$. This reduction combined with the application of ℓ^Y_{LZ} yields their $O(|X|^{\log\log|X|})$ algorithm for the general case [2].

We now elaborate on the Lovett-Zhang reduction. For some threshold r (which we eventually set to $O_\varepsilon(\log|X|)$), let $\textsc{near} = \textsc{near}_r(X) = \{x \in X : |x| \leq r\}$ and $\textsc{far} = \textsc{far}_r(X) = X - \textsc{near}$. Consider the construction of the function ℓ^Y_{LZ} with $Y = \textsc{near}$ instead of $Y = X$, Then we have:

$$f(\mathbf{0}) = \langle \ell^Y_{\mathsf{LZ}}, f\rangle - \sum_{x\in\textsc{far}} \ell^Y_{\mathsf{LZ}}(x) f(x). \tag{2.4}$$

If the sum (error term) being subtracted off is small, then we can still estimate $f(\mathbf{0})$ by estimating $\langle \ell^Y_{\mathsf{LZ}}, f\rangle$. It turns out that $\ell^Y_{\mathsf{LZ}}(x) \in [0,1]$ for all x and so the error is bounded by $|X| \max_{x\in\textsc{far}} f(x)$. Unfortunately, this might be quite large.

To get around this, Lovett and Zhang effectively replaced f by another function g for which $\max_{x\in\textsc{far}} g(x)$ is very small. To do this, they constructed an explicit function q (depending on X but otherwise not on f) and set $g = q\cdot f$. We have $f(\mathbf{0}) = g(\mathbf{0})/q(\mathbf{0})$ so it suffices to approximate $g(\mathbf{0})$. Replacing f by g in (2.4) we have:

$$g(\mathbf{0}) = \langle \ell^Y_{\mathsf{LZ}}, g\rangle - \sum_{x\in\textsc{far}} \ell^Y_{\mathsf{LZ}}(x) g(x) = \sum_{S\subseteq[n]} (\widehat{\ell^Y_{\mathsf{LZ}}}(S)/2^n)\widehat{g}(S) - \sum_{x\in\textsc{far}} \ell^Y_{\mathsf{LZ}}(x) g(x) \tag{2.5}$$

[2] Actually, the framework we state here slightly oversimplifies their proof. In their paper, they did't give an efficient algorithm to compute ℓ^Y_{LZ}. Instead, their claimed that the existence of such a function implies the problem can be solved efficiently by the maximum likelihood estimator.

and so

$$\begin{aligned}
|g(\mathbf{0}) - \sum_{S\subseteq[n]} (\widehat{\ell^Y_{\mathsf{LZ}}}(S)/2^n)\cdot \widehat{g}(S)| &\le \sum_{x\in\text{FAR}} \ell^Y_{\mathsf{LZ}}(x)g(x)\\
&\le \sum_{x\in\text{FAR}} \ell^Y_{\mathsf{LZ}}(x)q(x)\\
&\le \max_{x\in\text{FAR}} q(x)\sum_{s\in\text{FAR}} \ell^Y_{\mathsf{LZ}}(x). \qquad (2.6)
\end{aligned}$$

The function q will be chosen so that $q(x)$ (and therefore $g(x)$) is very small for all $x \in$ FAR, and therefore so is the right hand side of (2.6). Therefore $g(\mathbf{0})$ is well approximated by $\sum_{S\subseteq[n]}(\widehat{\ell^Y_{\mathsf{LZ}}}(S)/2^n)\cdot\hat{g}(S)$. To estimate this sum we need to be able to estimate $\hat{g}(S)$ efficiently from samples from $T_\mu(f)$, and this imposes additional constraints on the function q. The precise properties of the function q are given by the following lemma.

Lemma 2.4.1. *For any X and $r \ge (1/\mu^2)\cdot\log|X|$, there is a function q_r having the following properties:*

- *For all $x \in$ FAR, $q_r(x) \le e^{-\frac{1}{2}\mu^2 r}$.*
- $q_r(\mathbf{0}) \in [1/2, 1]$
- *$q_r(\mathbf{0})$ can be (ε,κ) approximated in time $\mathsf{poly}(n|X|\frac{1}{\varepsilon}\log\frac{1}{\kappa})$.*
- *For every S, there is a function $\alpha_S(z)$ for $z \in \{0,1\}^n$ such that for $z \sim T_\mu f$, $\alpha_S(z)$ is an unbiased estimator $\widehat{q_r\cdot f}(S)$ with range at most $\left(\frac{1}{\mu}\right)^{|S|}$ and is computable in time $2^{|S|}n^{O(1)}$.*

Lemma 2.4.1 is implicit in [31]; we prove it in Section 2.5. Using this lemma with $r = O(\log|X|\cdot\log\log|X|)$, Lovett and Zhang estimate $g(\mathbf{0})$ by estimating $\sum_S(\widehat{\ell^Y_{\mathsf{LZ}}}(S)/2^n)\cdot\hat{g}(S)$ as outlined above and get an algorithm that runs in time $\mathsf{poly}(k^{\log\log k}, n, 1/\varepsilon)$.

2.4.3 Improving ℓ

Let's summarize the two main results in Lovett and Zhang's approach:

(1) For any $r \ge (1/\mu^2)\cdot\log|X|$, there exist a function q_r such that:

- For all $x \in \textsc{far}$, $q_r(x) \leq e^{-\frac{1}{2}\mu^2 r}$.

(2) Let $Y = X \bigcap B(0,r)$, there exist a function $\ell_{\mathsf{LZ}} \in \mathcal{G}_0(Y)$ such that:

- $|\mathcal{H}(\ell^Y_{\mathsf{LZ}})| \leq |Y|2^{w(Y)} \leq |Y|2^r$.
- $\|\widehat{\ell^Y_{\mathsf{LZ}}}\|_{L_1} = |Y|^{O(\log w(Y))} \leq |Y|^{O(\log r)}$.

Combining these two functions into $q_r \cdot \ell_{LZ}$ lead to an algorithm that runs in time $O(|\mathcal{H}(\ell^Y_{\mathsf{LZ}})| \cdot \|\widehat{\ell^Y_{\mathsf{LZ}}}\|_{L_1}) = O(2^r|X|^{O(\log r)})$ with error at most $\Theta(|X| \cdot \|\widehat{\ell^Y_{\mathsf{LZ}}}\|_{L_1} \cdot q_r(x)) = \Theta(|X|^{O(\log r)}e^{-1/2\mu^2 r})$. In order to bound the error, r has to be $O(\log|X| \cdot \log\log|X|)$, which gives a $|X|^{O(\log\log|X|)}$ running time.

We follow the approach outlined above, but replace ℓ^Y_{LZ} by a better function. Our first attempt uses the Möbius function (Section 2.2.2), to construct a function $\ell_0 = \ell_{0,Y}$ with listable support $\mathcal{H}_0$ such that:

- $|\mathcal{H}_0| \leq |Y|2^{w(Y)}$,
- $\|\widehat{\ell_0}\|_{L_1} = |Y|2^{w(Y)}$.

Using this choice with $Y = X$ in the basic approach outlined in Section 2.4.1 gives a polynomial time estimator in the case $w(X) = O(\log n)$ since both $|\mathcal{H}_0|$ and $\|\widehat{\ell_0}\|_{L_1}$ are polynomial in $|X|$.

Using ℓ_0 in place of ℓ_{LZ} in the Lovett-Zhang approach with r set to be $\Theta_\varepsilon(\log|X|)$, we can bound use the above bound on $\|\widehat{\ell_0}\|_{L_1}$ and the bound on $q(x)$ for $x \in \textsc{far}$ in Lemma 2.4.1 to bound the error term in (2.6) from above by $|X|2^r e^{-\mu^2 r/2}$.

Unfortunately, even when $\mu = 1$, 2^r overwhelms $e^{-\mu^2 r/2}$ and the term is large. In an earlier version of this paper, we showed how to modify q to get improved bounds on $q(x)$ for $x \in \textsc{far}$ of the form $2^{-\beta(\mu)r}$, where $\beta(\mu) > 1$ for $\mu > .555$. Thus, for such values of μ the error term can be made arbitrarily small, thereby getting a polynomial time estimation algorithm for this value of μ. While one might hope to prove this for even smaller values of μ by improving q further, this approach seems to be incapable of working for arbitrary $\mu > 0$ since the functions $\beta(\mu)$ that are obtained in this way tend to 0 as μ tends to 0.

So instead of changing q, we modify the function ℓ to reduce $\|\widehat{\ell}\|_{L_1}$ from $2^r\mathsf{poly}(|X|)$ to $(1+\delta)^r\mathsf{poly}(|X|)$ for an arbitrary $\delta > 0$. By choosing $r = O_\delta(\log|X|)$ appropriately, the error term in (2.6) can be made arbitrarily small. In order for us to accomplish this, we will relax the condition $\ell \in \mathcal{G}_0(\textsc{near})$ to the condition that $\ell \in \mathcal{G}_\eta(\textsc{near})$ for a suitably small η. (Recall that $\mathcal{G}_\eta(Y)$ is the set of functions ℓ such that $\ell(\mathbf{0}) = 1$ and $|\ell(x)| \leq \eta$ for all $x \in Y - \{\mathbf{0}\}$.) The next lemma states the several properties that is achieved by our construction of ℓ.

Lemma 2.4.2. *Let $C \subseteq \{0,1\}^n$, $\delta > 0$ and $\eta > 0$. Let r be an upper bound on $w(C)$. There is a function $\ell = \ell_{C,\delta,\eta} : \{0,1\}^n \to \mathbb{R}$*

- $\ell \in \mathcal{G}_\eta(C^{\downarrow})$,
- $\|\widehat{\ell}\|_{L_1} \leq |C|^2 \cdot (1+2\delta)^r \cdot (2/\eta)^{\delta^{-1}\cdot\log(2\delta^{-1})}$,
- $\mathrm{supp}(\widehat{\ell}) \subseteq C^{\downarrow}$,
- *For any $S \subseteq [n]$, the Fourier coefficient $\hat{\ell}(S)$ can be computed in time $\mathsf{poly}(|C^{\downarrow}|, n)$.*

Lemma 2.4.2 is proved in Section 2.6. With the help of Lemma 2.4.1 and Lemma 2.4.2, we are ready to prove 2.3.1.

2.4.4 Proof of Lemma 2.3.1

Recall in Section 2.3 we show that Lemma 2.3.1 is enough to obtain the main theorem. In this section, we will show how Lemma Lemma 2.3.1 follows from Lemmas Lemmas 2.4.1 and 2.4.2 (which will be proven later).

To do this, apply Lemma 2.4.1 with $r = (100/\mu^4) \cdot \log(1/\mu) \cdot \log(k/\varepsilon)$ to get function q. We then apply Lemma 2.4.2 with $C = X \cap B(0,r)$, $\eta = \frac{\varepsilon}{20}$ and $\delta = \frac{\mu^2}{16}$ to get the resulting function ℓ. Define $u = \ell \cdot q$. We will show that this u satisfies all the properties we need in Lemma 2.3.1. We begin by noting that the third item (i.e. $u(0)$ can be efficiently approximated and lies in $[1/2, 1]$) follows by combining that $\ell(0) = 1$

and Lemma 2.4.1. Next, we give an unbiased estimator for $\langle u, f\rangle$. We know that:

$$\begin{aligned}\langle u, f\rangle &= \langle \ell \cdot q, f\rangle = \langle \ell, q\cdot f\rangle \\ &= \sum_S (\widehat{\ell}(S)/2^n)\cdot \widehat{qf}(S) \\ &= \sum_{S\subseteq C^{\downarrow}} (\widehat{\ell}(S)/2^n)\cdot \widehat{qf}(S)\end{aligned}$$

Lemma 2.4.1 shows that for any S, there exist an unbiased estimator $\alpha_S(z)$ for $\widehat{qf}(S)$, with range at most $\left(\frac{1}{\mu}\right)^{|S|}$ that is computable in time $2^{|S|}n^{O(1)}$. It then follows that $\sum_{S\subseteq C^{\downarrow}}(\widehat{\ell}(S)/2^n)\cdot\alpha_S(z)$ is an unbiased estimator with range at most $\|\widehat{\ell}\|_{L_1}\cdot\left(\frac{1}{\mu}\right)^r \leq (k/\varepsilon)^{\tilde{O}(1/\mu^4)}$ and it can be computed in time $|C^{\downarrow}|\cdot 2^r n^{O(1)} = (k/\varepsilon)^{\tilde{O}(1/\mu^4)}n^{O(1)}$. All that remains is to bound $|\langle u, f\rangle - u(\mathbf{0})f(\mathbf{0})|$.

$$\begin{aligned}|\langle u, f\rangle - u(\mathbf{0})f(\mathbf{0})| &= |\sum_{x\in X\setminus\{\mathbf{0}\}} \ell(x)q(x)f(x)| \\ &\leq |\sum_{x\in \textsc{near}\setminus\{\mathbf{0}\}} \ell(x)q(x)f(x)| + |\sum_{x\in\textsc{far}} \ell(x)q(x)f(x)| \\ &\leq \eta\cdot|\sum_{x\in\textsc{near}\setminus\{\mathbf{0}\}} f(x)| + \|\ell\|_\infty\cdot e^{-\frac{1}{2}\mu^2 r}|\sum_{x\in\textsc{far}} f(x)| \\ &\leq \eta + k^2\cdot(1+2\delta)^r\cdot(2/\eta)^{\delta^{-1}\cdot\log(2\delta^{-1})}\cdot e^{-\frac{1}{2}\mu^2 r}\end{aligned}$$

By plugging the values of r, η and δ, we have $|\langle u, f\rangle - u(\mathbf{0})f(\mathbf{0})| \leq \varepsilon/10$.

2.5 Proof of Lemma 2.4.1

In this section we prove Lemma 2.4.1. Most of the proof is adapted from Lovett and Zhang's paper [31].

Recall that $\textsc{far} = \textsc{far}_r = \{x\in X : |x| > r\}$. Define the set $E = \{y\in\{0,1\}^n : d_H(\mathbf{0}, y)\leq d_H(x_i, y) \text{ for all } x_i\in\textsc{far}\}$ and $q = T_\mu \mathbf{1}_E$. Next, we show that q satisfies the requirements. First we obtain the following lemma, which is essentially identical to Lemma 3.2 in [31], that proves the first three properties we need.

Lemma 2.5.1. *For any X and $r\geq(1/\mu^2)\cdot\log|X|$, define set* FAR *and E as above, we have:*

- $(T_\mu\mathbf{1}_E)(\mathbf{0})\geq 1/2$.

- *For $x_i \in \mathsf{Far}$, $(T_\mu \mathbf{1}_E)(x_i) \le e^{-\frac{1}{2}\cdot\mu^2\cdot|x_i|}$.*

The function $\mathbf{1}_E(\cdot)$ can be computed in time $\mathsf{poly}(n, |X|)$. Further, $(T_\mu \mathbf{1}_E)(0)$ can be computed to additive error ε in time $\mathsf{poly}(n, |X|, 1/\varepsilon)\cdot\log(1/\kappa)$.

Proof. We first lower bound $(T_\mu \mathbf{1}_E)(0)$. Let $|X| = k$, define $s = \log(k)/\mu^2$. By definition,

$$\begin{aligned}(T_\mu \mathbf{1}_E)(\mathbf{0}) = \Pr_{e\sim\nu_\mu}[\mathbf{0}+e \in E] &= 1 - \Pr_{e\sim\nu_\mu}[\mathbf{0}+e \notin E] \\ &\ge 1 - \Bigg(\sum_{i:d_H(\mathbf{0},x_i)\ge s} \Pr_{y\sim\nu_\mu}[d_H(\mathbf{0},y) \ge d_H(x_i,y)]\Bigg)\end{aligned}$$

The last inequality follows by the definition of E and union bound. To lower bound the right hand side, let us define $S_i = \{j \in [n] : (x_i)_j = 1\}$. If $|x_i| \ge s$, then $|S_i| \ge s$. For such a point $x_i \in S$,

$$\Pr_{y\sim\nu_\mu}[d_H(\mathbf{0},y) \ge d_H(x_i,y)] = \Pr_{e\sim\nu_\mu}\left[\sum_{j\in S_i} e_j \ge |S_i|/2\right]$$

To bound the above sum, we recall the Chernoff bound.

Proposition. *Let $X_1, \ldots, X_n$ be n independent $\{0,1\}$ random variables such that $1 \le i \le n$, $\mathbf{E}[X_i] = p$. If $q > p$, then,*

$$\Pr\Big[X_1 + \ldots + X_n \ge n\cdot q\Big] \le \exp\bigg(-\frac{n}{2}\cdot\Big(\frac{q}{p}-1\Big)^2\bigg).$$

Applying the above proposition, we get that

$$\Pr_{y\sim\nu_\mu}[d_H(\mathbf{0},y) \ge d_H(x_i,y)] \le \exp\left(\frac{-|S_i|}{2}\cdot\mu^2\right) \le \frac{1}{2k}.$$

This implies that

$$(T_\mu \mathbf{1}_E)(\mathbf{0}) \ge 1 - \Bigg(\sum_{i:d_H(\mathbf{0},x_i)\ge s} \Pr_{y\sim\nu_\mu}[d_H(\mathbf{0},y) \ge d_H(x_i,y)]\Bigg) \ge \frac{1}{2}.$$

We now upper bound $(T_\mu \mathbf{1}_E)(x_i)$ for $x_i \in \mathsf{Far}$. Note that $(T_\mu \mathbf{1}_E)(x_i) = \Pr_{e\sim\nu_\mu}[x_i + e \in E]$. Note that if $x_i + e \in E$, then $d_H(x_i+e,\mathbf{0}) \le d_H(x_i+e, x_i)$. This implies that $\sum_{j\in S_i} e_j \ge |S_i|/2$. Applying the Chernoff bound, we have

$$(T_\mu \mathbf{1}_E)(x_i) = \Pr_{e\sim\nu_\mu}[x_i+e \in E] \le \Pr_{e\sim\nu_\mu}\left[\sum_{j\in S_i} e_j \ge \frac{|S_i|}{2}\right] \le e^{-\frac{1}{2}\cdot\mu^2\cdot|x_i|}.$$

The fact that $\mathbf{1}_E(\cdot)$ can be computed in time $\mathsf{poly}(n,k)$ follows from the definition of E. Further, since $\mathbf{1}_E(\cdot)$ is computable in time $\mathsf{poly}(n,k)$ and ν_μ is samplable in time $\mathsf{poly}(n)$, we immediately get that

$$(T_\mu \mathbf{1}_E)(\mathbf{0}) = \Pr_{e\sim\nu_\mu}[\mathbf{0}+e\in E],$$

can be approximated to ε in time $\mathsf{poly}(n,k,1/\varepsilon)\cdot\log(1/\kappa)$ with confidence $1-\kappa$. $\square$

Finally, we prove the final requirement of Lemma 2.4.1. For this we must construct, for each $S\subset[n]$, a suitable function α_S. We first show how to build an unbiased estimator for $\widehat{q\cdot f}(S)$ using a random sample $z\sim T_\mu f$. Since $(q\cdot f)(x) = f(x)\cdot(T_\mu\mathbf{1}_E)(x)$, we get that for any $S\subseteq\{0,1\}^n$.

$$\widehat{q\cdot f}(S) = \langle (X_S f),(T_\mu \mathbf{1}_E)\rangle = \langle (T_\mu X_S f),\mathbf{1}_E\rangle.$$

We now make two observations. The first is that for any $S\subseteq[n]$, $T_{\mu,S}$ is a self-adjoint operator. The second is that if $S,S'\subseteq[n]$ are disjoint sets, then the operators $X_{S'}$ and $T_{\mu,S}$ commute. Decomposing $T_\mu = T_{\mu,S}T_{\mu,\overline{S}}$, we have

$$T_\mu X_S f = T_{\mu,S}T_{\mu,\overline{S}}X_S f = T_{\mu,S}X_S T_{\mu,\overline{S}} f = T_{\mu,S}X_S T_{\mu,S}^{-1}T_\mu f.$$

Thus, we get

$$\widehat{q\cdot f}(S) = \langle T_{\mu,S}X_S T_{\mu,S}^{-1}T_\mu f,\mathbf{1}_E\rangle = \mathbf{E}_{z\sim T_\mu f}\langle T_{\mu,S}X_S T_{\mu,S}^{-1}\mathbf{1}_z,\mathbf{1}_E\rangle \tag{2.7}$$

Defining $\alpha_S(z) = \langle T_{\mu,S}X_S T_{\mu,S}^{-1}\mathbf{1}_z,\mathbf{1}_E\rangle$, we can see that $\alpha_S(z)$ is an unbiased estimator for $\widehat{q\cdot f}(S)$. We show that $\alpha_S(z)$ has the properties we need.

Lemma 2.5.2. *For any $S\subseteq\{0,1\}^n$, $\alpha_S(z)$ can be computed in time $2^{O(|S|)}n^{O(1)}$.*

Proof. We define $A=\{y : y_{\overline{S}} = z_{\overline{S}}\}$. Then $|A| = 2^{|S|}$. Define the linear subspace $\mathcal{F}_A = \{f\in\mathcal{F} : \mathit{supp}(f)\subset A\}$ of $\mathcal{F}$. Then we claim that $\mathcal{F}_A$ is an invariant subspace under the operator $T_{\mu,S}X_S T_{\mu,S}^{-1}$.

It is easy to see $\mathcal{F}_A$ is invariant under X_S. Also, consider $\{\mathbf{1}_y : y\in A\}$ being the basis of $\mathcal{F}_A$. For any $y\in A$, we have:

$$(T_{\mu,S}\mathbf{1}_y)(x) \quad = \sum_{t\in\{0,1\}^n : t_i=0 \text{ for } i\notin S} \mathbf{1}_y(x\oplus t)\prod_{i\in S}\frac{1}{2}(1+(-1)^{t_i}\mu)$$

Then we know:

$$\begin{aligned} T_{\mu,S}\mathbf{1}_y &= \sum_{t\in\{0,1\}^n : t_i = 0 \text{ for } i\notin S} \mathbf{1}_{y\bigoplus t} \prod_{i\in S} \frac{1}{2}(1+(-1)^{t_i}\mu) \\ &= \sum_{x\in A} \mathbf{1}_x \cdot \prod_{i\in S} \frac{1}{2}(1+(-1)^{y_i \bigoplus x_i}\mu) \end{aligned}$$

This directly implies $\mathcal{F}_A$ is invariant under $T_{\mu,S}$. Also, we can compute the matrix of $\mathcal{F}_A$ under this basis in time $O(|A|^2)$. This implies $T_{\mu,S}X_S T_{\mu,S}^{-1}\mathbf{1}_z$ can be computed in time $2^{O(|S|)}$. Using the fact that $\mathbf{1}_E(\cdot)$ can be efficiently evaluated at any point, we conclude that $\langle T_{\mu,S}X_S T_{\mu,S}^{-1}\mathbf{1}_z, \mathbf{1}_E\rangle$ can be evaluated in time $2^{O(|S|)}n^{O(1)}$.

□

Lemma 2.5.3. *For any $S \subseteq \{0,1\}^n$, $|\alpha_S(z)| \leq (1/\mu)^{|S|}$.*

Proof. First we recall the following facts from [31] (Claim 3.5 in [31]).

Claim. $\|T_{\mu,i}\|_{1\to 1} = 1$ *and* $\|T_{\mu,i}^{-1}\|_{1\to 1} = 1/\mu$.

Proof of the Claim. The bound $\|T_{\mu,i}f\|_1 \leq \|f\|_1$ is immediate, and is tight for $f = 1$. To derive the bound on $T_{\mu,i}^{-1}$, let x_0, x_1 be such that $(x_0)_i = 0$, $(x_1)_i = 1$ and $(x_0)_j = (x_1)_j$ for all $j \neq i$. If $(f(x_0), f(x_1)) = (a, b)$ then $T_{\mu,1}^{-1}f = (1/2\mu)\cdot((1+\mu)a - (1-\mu)b, -(1-\mu)a+(1+\mu)b)$. Then $|(T_{\mu,i}^{-1}f)(x_0)| + |(T_{\mu,i}^{-1}f)(x_1)| \leq (1/\mu)(|f(x_0)|+|f(x_1)|)$. The claim follows by summing over all choices for x_0, x_1, and noting that the bound is tight for $f(x) = (-1)^{x_i}$. □

The above immediately implies

$$\|T_{\mu,S}\|_{1\to 1} \leq 1, \qquad \|T_{\mu,S}^{-1}\|_{1\to 1} \leq (1/\mu)^{|S|}. \tag{2.8}$$

Using $\|X_S\|_{1\to 1} \leq 1$, we know $\|T_{\mu,S}X_S T_{\mu,S}^{-1}\|_{1\to 1} \leq (1/\mu)^{|S|}$. This implies:

$$|\alpha_S(z)| = |\langle T_{\mu,S}X_S T_{\mu,S}^{-1}\mathbf{1}_z, \mathbf{1}_E\rangle| \leq \|T_{\mu,S}X_S T_{\mu,S}^{-1}\mathbf{1}_z\|_1 \leq \|T_{\mu,S}X_S T_{\mu,S}^{-1}\|_{1\to 1} \leq (1/\mu)^{|S|}$$

□

Combining Lemma 2.5.1, Lemma 2.5.2 and Lemma 2.5.3, we get the result.

2.6 Proof of Lemma 2.4.2

In this section, we prove Lemma 2.4.2 which given $C \subseteq \{0,1\}^n$ and $\delta, \eta > 0$ constructs a suitable function ℓ. As a warmup, we construct the function ℓ_0 mentioned earlier. The function ℓ_0 is specified by the set $X \subseteq \{0,1\}^n$, which we change to C to match the notation of Lemma 2.4.2. We are given $C \subseteq \{0,1\}^n$ and want to construct a function $\ell_0 \in \mathcal{G}_0(C)$ with a listable support $\mathcal{H}_0$ for $\hat{\ell}_0$ such that:

- $|\mathcal{H}_0| \leq |C|2^{w(C)}$.
- $\|\widehat{\ell_0}\|_{L_1} = |C|2^{w(C)}$

The function we construct will satisfy the stronger condition that $\ell_0 \in \mathcal{G}_0(C^{\downarrow})$, which means that it is 1 at $\mathbf{0}$ and 0 on every other point of $C^{\downarrow}$.

We introduce some notation to represent the natural correspondence between strings in $\{0,1\}^n$ and subsets of $[n]$. For $z \in \{0,1\}^n$, define $\textsc{ones}(z) = \{i \in [n] : z_i = 1\}$. For $A \subseteq \{0,1\}^n$, let $\mathcal{H}(A)$ be the collection of subsets $\{\textsc{ones}(z) : z \in A\}$. We define $\mathcal{H}_0 = \mathcal{H}(C^{\downarrow})$. Observe that given C, we can efficiently list all the sets of $\mathcal{H}_0$ and $|\mathcal{H}_0| \leq |C|2^{w(C)}$.

Note that the requirement of $\widehat{\ell_0}$ being supported on $\mathcal{H}_0 = \mathcal{H}(C^{\downarrow})$ is the same as requiring the function ℓ_0 to be of the form $\ell_0 = \sum_{S \in C^{\downarrow}} \beta_S \cdot \chi_S$. In order to find the coefficients $\{\beta_S\}_{S \in C^{\downarrow}}$, we start by defining the family of functions $\{\mathbf{1}_{\succeq z}\}_{z \in \{0,1\}^n}$ as follows:

$$\mathbf{1}_{\succeq z}(x) = \mathbf{1}_{x \succeq z}$$

It is easy to verify:

$$\mathbf{1}_{\succeq z}(x) = \prod_{i : z_i = 1} x_i = \prod_{i : z_i = 1} \frac{1 - \chi_i(x)}{2} = \frac{1}{2^{|z|}} \sum_{S \subseteq \textsc{ones}(z)} (-1)^{|S|} \chi_S(x)$$

This implies that $\|\widehat{\mathbf{1}_{\succeq z}}\|_{L_1} = 1$ and $\text{supp}(\widehat{\mathbf{1}_{\succeq z}}) \subseteq \mathcal{H}(z^{\downarrow})$. Thus, a linear combination of functions $(\mathbf{1}_{\succeq z})_{z \in C^{\downarrow}}$ will have Fourier support in $\mathcal{H}(C^{\downarrow})$. We will construct ℓ_0 as a linear combination of $(\mathbf{1}_{\succeq z})_{z \in C^{\downarrow}}$. By considering the restriction of the function ℓ_0 to $C^{\downarrow}$ we can use the Möbius transform to find the linear combination.

For a function $f \in \mathcal{F}$, let f^R denote the function obtained by restricting the domain to $C^{\downarrow}$. The condition $\ell_0 \in \mathcal{G}_0(C^{\downarrow})$ is the same as $\ell_{\mathbf{0}}^R = \mathbf{1}_{\mathbf{0}}^R$. Observe that in the poset $C^{\downarrow}$ we have $\mathbf{1}_{\succeq z}^R = \sum_{y \succeq z} \mathbf{1}_y^R$ for all $z \in C^{\downarrow}$. By Proposition 2.2.2 and Corollary 2.2.4 we have:

$$\mathbf{1}_y^R = \sum_{y \preceq z \in C^{\downarrow}} (-1)^{|z \setminus y|} \mathbf{1}_{\succeq z}^R \quad \text{for all } z \in C^{\downarrow}$$

This result can also be verified directly without Proposition 2.2.2 and Corollary 2.2.4.

For $y \in C^{\downarrow}$, define the function $\ell_y = \sum_{y \preceq z \in C^{\downarrow}} (-1)^{|z \setminus y|} \mathbf{1}_{\succeq z}$. We claim that the function $\ell_0 = \ell_{\mathbf{0}}$ satisfies the requirements. To see this, note that $\ell_y^R = \mathbf{1}_y^R$ (but in general ℓ_y may disagree with $\mathbf{1}_y$ outside of $C^{\downarrow}$). Thus, $\ell_0 \in \mathcal{G}_0(C^{\downarrow})$. Further,

$$\|\widehat{\ell_0}\|_{L_1} \leq \sum_{z \in C^{\downarrow}} |(-1)^{|z|}| \cdot \|\mathbf{1}_{\succeq z}\|_{L_1} \leq \sum_{z \in C^{\downarrow}} 1 \leq |C^{\downarrow}| \leq |C| \cdot 2^{w(C)}.$$

We now turn to the proof of Lemma 2.4.2. We are given $C \subseteq \{0,1\}^n$ and $\delta, \eta > 0$, and an upper bound r on $w(C)$. We want to construct a function ℓ satisfying the conclusions of the lemma.

As mentioned in Section 2.4.3, the reason why ℓ_0 is not good enough for us is because the Fourier L_1 norm grows too fast. To circumvent this, we start with a modified family of functions $\ell_{\delta,y} = \sum_{y \preceq z \in C^{\downarrow}} (-1)^{|z \setminus y|} \cdot \delta^{|z|} \cdot \mathbf{1}_{\succeq z}$. Note that $\ell_{\delta,y}$ generalizes the function ℓ_y (which is obtained by setting $\delta = 1$). We will construct ℓ as a linear combination of $\{\ell_{\delta,y}\}_{y \in C^{\downarrow}}$. First we formally prove some properties of $(\ell_{\delta,y} : y \in C^{\downarrow})$.

Proposition 2.6.1. *For any $\delta > 0, y \in C^{\downarrow}$, the function $\ell_{\delta,y} = \sum_{y \preceq z \in C^{\downarrow}} (-1)^{|z \setminus y|} \cdot \delta^{|z|} \cdot \mathbf{1}_{\succeq z}$ satisfies the following properties:*

- *For $x \in C^{\downarrow}$, $\ell_{\delta,y}(x) = \mathbf{1}_{x \succeq y} \cdot (1-\delta)^{|x|-|y|} \cdot \delta^{|y|}$.*
- $\operatorname{supp}(\widehat{\ell_{\delta,y}}) \subseteq C^{\downarrow}$.
- $\|\widehat{\ell_{\delta,y}}\|_{L_1} \leq |C| \cdot (1+\delta)^{w(C)-|y|} \cdot \delta^{|y|}$

Proof. First we can rewrite $\ell_{\delta,y}$ as $\ell_{\delta,y} = \delta^{|y|} \sum_{y \preceq z \in C^{\downarrow}} (-\delta)^{|z \setminus y|} \cdot \mathbf{1}_{\succeq z}$. For any $x \in C^{\downarrow}$,

$$\ell_{\delta,y}(x) = \delta^{|y|} \sum_{y \preceq z \in C^{\downarrow}} (-\delta)^{|z \setminus y|} \cdot \mathbf{1}_{\succeq z}(x) = \delta^{|y|} \sum_{y \preceq z \preceq x} (-\delta)^{|z \setminus y|} = \mathbf{1}_{x \succeq y} \cdot (1-\delta)^{|x|-|y|} \cdot \delta^{|y|}$$

Since $\mathrm{supp}(\widehat{\mathbf{1}_{\succeq z}}) \subseteq C^{\downarrow}$, we deduce that $\mathrm{supp}(\widehat{\ell_{\delta,y}}) \subseteq C^{\downarrow}$. For the last requirement,

$$\begin{aligned}
\|\widehat{\ell_{\delta,y}}\|_{L_1} &\leq \delta^{|y|} \sum_{y \preceq z \in C^{\downarrow}} |(-\delta)^{|z \setminus y|}| \cdot \|\mathbf{1}_{\succeq z}\|_{L_1} \\
&\leq \delta^{|y|} \sum_{y \preceq z \in C^{\downarrow}} \delta^{|z \setminus y|} \\
&\leq \delta^{|y|} \sum_{t \in C} \sum_{y \preceq z \preceq t} \delta^{|z \setminus y|} \\
&\leq |C| \cdot \delta^{|y|} \cdot (1+\delta)^{w(C)-|y|}
\end{aligned}$$

□

Note that we have relaxed the requirement on ℓ, namely $\ell \in \mathcal{G}_\eta(C)$ for some appropriately small η as opposed to ℓ_0 which was in $\mathcal{G}_0(C)$. Recall that we will construct ℓ as a linear combination of form $\ell_{\delta,y}$ for $y \in C^{\downarrow}$. We now impose the additional requirement that the coefficient of $\ell_{\delta,y}$ depends only on $|y|$. This will help us in search of the said coefficients. With this, let $\ell = \sum_{y \in C^{\downarrow}} v_{|y|} \cdot \ell_{\delta,y}$, where $v = (v_0, ..., v_{w(C)})$ is the vector of coefficients. By Proposition 2.6.1 for any $x \in C^{\downarrow}$:

$$\ell(x) = \sum_{y \preceq x} v_{|y|} \cdot \delta^{|y|} \cdot (1-\delta)^{|x|-|y|} = \sum_{t=0}^{|x|} v_t \cdot \binom{|x|}{t} \cdot \delta^t \cdot (1-\delta)^{|x|-t}$$

Since the value of ℓ only depends on the weight of x, we can define a function $\tilde{\ell}$ on nonnegative integers so that $\ell(x) = \tilde{\ell}(|x|)$. Now we have $\tilde{\ell}(m) = \sum_{t=0}^{m} v_t \cdot \binom{m}{t} \cdot \delta^t \cdot (1-\delta)^{m-t}$ for $0 \leq m \leq w(C)$, and the condition $\ell \in \mathcal{G}_\eta(C^{\downarrow})$ is thus equivalent to $\tilde{\ell}(0) = 1$ and $|\tilde{\ell}(i)| \leq \eta$ for $i > 0$. Note that these are linear constraints on the entries of the vector v.

Also, applying Proposition 2.6.1, the Fourier L_1 norm can be bounded by:

$$\begin{aligned}
\|\widehat{\ell}\|_{L_1} &\leq \sum_{y \in C^{\downarrow}} |v_{|y|}| \cdot \|\widehat{\ell_{\delta,y}}\|_{L_1} \\
&\leq \|v\|_\infty \sum_{y \in C^{\downarrow}} |C| \cdot \delta^{|y|} \cdot (1+\delta)^{w(C)-|y|} \\
&\leq \|v\|_\infty \cdot |C| \cdot \sum_{j=0}^{w(C)} \delta^j (1+\delta)^{w(C)-j} \cdot |\{y \in C^{\downarrow} : |y| = j\}| \\
&\leq \|v\|_\infty \cdot |C| \cdot \sum_{j=0}^{w(C)} \delta^j (1+\delta)^{w(C)-j} \cdot |C| \cdot \binom{w(C)}{j} \\
&= \|v\|_\infty \cdot |C|^2 \cdot (1+2\delta)^{w(C)}.
\end{aligned}$$

Thus, we seek to find a vector $v = (v_0, \dots, v_{w(C)})$ such that $\|v\|_\infty$ is as small as possible while satisfying the linear constraints dictated by the requirement $\tilde{\ell}(0) = 1$ and $|\tilde{\ell}(i)| \leq \eta$ for $i > 0$. To do this, recall that $w(C) \leq r$ and define the matrix $A_{\delta,r} \in \mathbb{R}^{(r+1)\times(r+1)}$ as

$$A_{\delta,r}(i,j) = \binom{i}{j} \cdot \delta^j \cdot (1-\delta)^{i-j}.$$

Then we have $\tilde{\ell}(m) = (A_{\delta,r} \cdot v^T)_m$. Now the task of constructing ℓ is equivalent to finding a vector v with L_∞ norm as small as possible such that $(A_{\delta,r}(i,j) \cdot v^T)_0 = 1$ and $|(A_{\delta,r}(i,j) \cdot v^T)_m| \leq \eta$ for $m > 0$.

We note that this problem is equivalent to problem of finding a "robust local inverse" for the matrix $A_{\delta,r}$, which has been studied in [14, 32]. The following theorem is an easy corollary of the main result of [32]. We provide the reduction and a brief introduction to their result in Section 2.7.

Theorem 2.6.2. (Moitra-Saks [32]) *For any $\eta > 0$, there exists $v \in \mathbb{R}^{r+1}$ such that $\|A_{\delta,r} \cdot v - e_0\|_\infty \leq \eta$, $\|v\|_\infty \leq (2/\eta)^{(1/\delta)\cdot\log(2/\delta)}$ and the zeroth coordinate of $A_{\delta,r} \cdot v$ is 1. Here $e_0 \in \mathbb{R}^{r+1}$ denotes the unit vector with 1 at the zeroth coordinate. Further, v can be computed in time $\mathsf{poly}(r)$.*

Applying this theorem directly, we have $\ell \in \mathcal{G}_\eta(C^\downarrow)$ and $\|\widehat{\ell}\|_{L_1} \leq |C|^2 \cdot (1+2\delta)^r \cdot (2/\eta)^{\delta^{-1}\cdot\log(2\delta^{-1})}$. That finishes the proof of Lemma 2.4.2.

2.7 Robust local inverse from [32]

Here we briefly introduce the main result in Moitra and Saks [32]. Define matrix $A_{\mu,n} \in \mathbb{R}^{(n+1)\times(n+1)}$ by:

$$A_{\mu,n}(i,j) = \binom{i}{j} \cdot \mu^j \cdot (1-\mu)^{i-j},$$

where $\binom{i}{j} = 0$ if $j > i$. Following their paper, we now define an ε-local inverse.

Definition 2. *Let $v \in \mathbb{R}^{n+1}$ such that $\|A_{\mu,n} \cdot v - e_0\|_\infty \leq \varepsilon$. Such a vector v is said to be an ε-local inverse of $A_{\mu,n}$.*

Further, $\|v\|_\infty$ is defined to be the sensitivity of such a vector. Definition 2.1 from [32] defines $\sigma_n(\mu, \varepsilon)$ to be

$$\sigma_n(\mu, \varepsilon) = \min_{\|A_{\mu,n} \cdot v - e_0\|_\infty \leq \varepsilon} \|v\|_\infty.$$

The next observation states that the v achieving the optimum in the above definition can be found using linear programming.

Observation 2.7.1. *Using linear programming, it is possible to find $v \in \mathbb{R}^{n+1}$ in time $\mathsf{poly}(n)$ such that $\|A_{\mu,n} \cdot v - e_0\|_\infty \leq \varepsilon$, such that $\|v\|_\infty = \sigma_n(\mu, \varepsilon)$.*

We now restate Theorem 2.2 from [32] which gives an upper bound on $\sigma_n(\mu, \varepsilon)$.

Theorem. *For all positive integers n and $\mu, \varepsilon > 0$, $\sigma_n(\mu, \varepsilon) = (1/\varepsilon)^{f(\mu)}$ where $f(\mu) = (1/\mu) \cdot \log(2/\mu)$.*

Theorem 2.6.2 is just an easy corollary of the result.

Proof of Theorem 2.6.2. Apply the above theorem by choosing n to be r, μ to be δ and ε to be $\frac{\eta}{1+\eta}$. Assume w is the local inverse we get by Observation 2.7.1. Let α_0 be the zeroth coordinate of $A_{\delta,r} \cdot v$. Note that $1 + \frac{\eta}{1+\eta} \geq \alpha_0 \geq 1 - \frac{\eta}{1+\eta}$.Define $v = w/\alpha_0$. Then the zeroth coordinate of $A_{\delta,r} \cdot v$ is 1. For the other coordinate $i \neq 0$, we have:

$$|(A_{\delta,r} \cdot v)_i| = |(A_{\delta,r} \cdot w)_i|/\alpha_0 \leq \frac{\eta}{1+\eta} \cdot \left(1 - \frac{\eta}{1+\eta}\right)^{-1} \leq \eta$$

Also we have: $\|v\|_\infty = (1/\alpha_0) \cdot \|w\|_\infty \leq (1 - \frac{\eta}{1+\eta})^{-1}((1+\eta)/\eta)^{(2/\delta)\cdot\log(1/\delta)} \leq (2/\eta)^{(2/\delta)\cdot\log(1/\delta)}$. This proves Theorem 2.6.2. ☐

2.8 Reduction from LBA to KSA

In this section we prove Lemma 2.1.1. First we restate the lemma:

Lemma. *Assume that we have access to a population recovery algorithm $A(k)$ that operates under the assumption $KSA(k)$. Then there is a population recovery algorithm B that operates under the assumption $LBA(k)$ which on size parameter n, makes n calls to algorithm $A(2k)$ one for each size parameter $i \in [n]$ with error parameter $\min(1/3k, \varepsilon)$, such that B is correct provided that all calls to A are correct.*

Proof. Let f be the distribution we want to recover; by the assumptions we know $|\mathrm{supp}(f)| \le k$ and $f(x) \ge 1/k$ for all $x \in \mathrm{supp}(f)$. For all $i \in [n]$, define $\pi_i : \{0,1\}^n \to \{0,1\}^i$ to be the projection operator that projects a vector to its first i coordinates. Let $\pi_i f$ denote the probability distribution on $\{0,1\}^i$ where $\pi_i f(z) = \sum_{x:\pi_i(x)=z} f(x)$. Let $\tau = \min(1/3k, \varepsilon)$. We claim that, after the $i-$th oracle call, we can recover $\pi_i f$ correctly with error less than τ.

We prove this claim by induction. First it is easy to see we can recover $\pi_1 f$. Then, assume we already have an estimate of $\pi_i f$(denoted by $\tilde{f}_i$) with error less than τ. Since $f(x) \ge 1/k$ for all $x \in \mathrm{supp}(f)$, we also have: $\pi_i f(x) \ge 1/k$ for all $x \in \mathrm{supp}(\pi_i f)$. Recall that $\tilde{f}_i$ is close to $\pi_i f$ within error $\tau \le 1/3k$. That means $\tilde{f}_i(x) < 1/2k$ implies $\pi_i f(x) = 0$. We can simply set the value of $\tilde{f}_i$ on all those points to 0 so we have: $\mathrm{supp}(\tilde{f}_i) = \mathrm{supp}(\pi_i f)$. Let $S = \mathrm{supp}(\tilde{f}_i) \times \{0,1\}$ be a subset in $\{0,1\}^{i+1}$. It is easy to see $|S| \le 2k$ and $\mathrm{supp}(\pi_{i+1}) \subset S$. So we can make our oracle call using this set S with parameter $i+1, 2k, \tau$ to recover $\pi_{i+1} f$. That means, after n oracles, we can recover f within error $\tau \le \varepsilon$.

□

2.9 Proof of Corollary 2.1.3

Without loss of generality, assume $\|f\|_1 = 1$. We may further assume $f(\mathbf{0}) > 0$ and it maximizes $|f(x)|$, thus $f(0) > 1/k$. Define $f^+ = f \cdot \mathbf{1}_{>0}$ and $f^- = -f \cdot \mathbf{1}_{<0}$, thus $f = f^+ - f^-$. Normalizing these two terms we have,

$$f = \|f^+\|_1 \cdot \frac{f^+}{\|f^+\|_1} - \|f^-\|_1 \cdot \frac{f^-}{\|f^-\|_1}.$$

If $\|f^-\|_1 = 0$ then we just omit the second term.

Here $g^+ = \frac{f^+}{\|f^+\|_1}$ and $g^- = \frac{f^-}{\|f^-\|_1}$ can be viewed as distributions supported on $\mathrm{supp}(f)$. Applying Lemma 2.3.1 with parameter $\varepsilon = 1/k$ and $X = \mathrm{supp}(f)$, we get functions u and $\alpha : \{0,1\}^n \to \mathbb{R}$ satisfying $u(0) \in [1/2, 1]$ and $|\alpha(z)| \le k^{\tilde{O}(1/\mu^4)}$, such that

$$\begin{aligned} |\langle u, g^+\rangle - u(0)g^+(0)| \le \frac{1}{10k}, \qquad \langle u, g^+\rangle = \mathbf{E}_{z\sim T_\mu g^+}[\alpha(z)], \\ |\langle u, g^-\rangle - u(0)g^-(0)| \le \frac{1}{10k}, \qquad \langle u, g^-\rangle = \mathbf{E}_{z\sim T_\mu g^-}[\alpha(z)]. \end{aligned} \tag{2.9}$$

We will show that

$$1/2k \leq \langle u, f\rangle \leq k^{\tilde{O}(1/\mu^4)} \cdot \|T_\mu f\|_1. \tag{2.10}$$

For the first part of equation (2.10), since $f = \|f^+\|_1 \cdot g^+ - \|f^-\|_1 \cdot g^-$, the left two inequalities of (2.9) directly imply

$$|\langle u, f\rangle - u(0)f(0)| \leq \frac{1}{10k}(\|f^+\|_1 + \|f^-\|_1) = \frac{1}{10k}$$

Thus $\langle u, f\rangle \geq u(0)f(0) - \frac{1}{10k} \geq \frac{1}{2k}$. For the second part of equation (2.10), the right two equations of (2.9) imply

$$\begin{aligned}
\langle u, f\rangle &= \|f^+\|_1 \cdot \mathbf{E}_{z\sim T_\mu g^+}\alpha(z) - \|f^-\|_1 \cdot \mathbf{E}_{z\sim T_\mu g^-}\alpha(z) \\
&= \sum_{z\in\{0,1\}^n} \alpha(z) \cdot \left(\|f^+\|_1 \cdot T_\mu\left(\frac{f^+}{\|f^+\|_1}\right)(z) - \|f^-\|_1 \cdot T_\mu\left(\frac{f^-}{\|f^-\|_1}\right)(z)\right) \\
&= \sum_{z\in\{0,1\}^n} \alpha(z) \cdot T_\mu f(z) \\
&\leq \|T_\mu f\|_1 \cdot \max_{z\in\{0,1\}^n} \alpha(z) \\
&\leq k^{\tilde{O}(1/\mu^4)} \cdot \|T_\mu f\|_1
\end{aligned}$$

These two results imply $\|T_\mu f\|_1 \geq k^{-\tilde{O}(1/\mu^4)}$, which finishes the proof.

Chapter 3

Lower Bound for Non-adaptive Generalized Noisy Decision Trees and Noisy Broadcasts

3.1 Introduction

3.1.1 Problem overview

It has long been an interesting problem to find reliable ways to do communication or computation against noise. A general goal of this problem, in both theory and practice, is to minimize the additional resources needed to get reliable results. This problem was studied in a variety of different models, such as decision trees [19, 39, 17, 18], formulas and circuits [38, 16], quantum computation [3, 28] and different kind of communication models [15, 40, 29, 36, 24, 13].

The *noisy broadcast model* was proposed by El Gamal [15] in 1984, and later popularized by Yao [44], as a model to study the effect of noise in highly distributed systems. The noisy broadcast model considers n processors $P_1, ...P_n$ and one receiver P_0. Each processor P_i has a private input bit $x_i \in \{0, 1\}$, and the goal is for P_0 to evaluate a particular function $f(x_1, ..., x_n)$. The communication between processors is done as follows: in each time step, a prespecified processor broadcasts a single bit to all other processors. For some fixed noise parameter $\varepsilon < 1/2$, each processor receives the broadcast bit with probability $1 - \varepsilon$ and receives the complement of the bit with probability ε. At the end of the algorithm, P_0 should output the final answer based on all the bits it has heard. The formal definition of noisy broadcast model will be given in Section 3.7.1.

Let's consider the identity function $ID(x_1, ..., x_n) = (x_1, ..., x_n)$, which means the receiver P_0 needs to learn the entire input correctly. Note that P_0 can evaluate any function by itself if it gets the correct input $(x_1, ..., x_n)$, thus the function ID is complete for the model. It is easy to see ID can be computed using $O(n \log n)$ broadcasts: simply let each processor broadcasts its bit $O(\log n)$ times and let P_0 output the majority value for each bit. In 1988, Gallager [23] gave a protocol for ID using only $O(n \log \log n)$ broadcasts. Later in 2008, Goyal, et al. [24] showed that this bound is tight up to a constant factor. The lower bound proof was obtained by introducing a new model called *generalized noisy decision tree* (*gnd-tree*) model. They showed that noisy broadcast protocols can be simulated efficiently by gnd-trees and then they proved a lower bound for ID in the gnd-tree model.

The gnd-tree model is a centralized computational model. Again the goal is to compute some function of the input $x \in \{0,1\}^n$. However, instead of getting the true input x, the algorithm only has access to an indexed collection $(X_\gamma^\varepsilon : \gamma \in \Gamma)$ of noisy copies of x, where each noisy copy X_γ^ε is obtained by flipping each coordinate independently with some fixed probability ε. In each time step, the algorithm can query an arbitrary boolean function on some noisy copy of x. One noisy copy can be queried multiple times. This process can be interpreted as a binary tree where each node is labeled by an arbitrary boolean function and an index $\gamma \in \Gamma$ specifying one of the noisy copies. The two outgoing edges of the node indicate the value of the boolean function on that noisy copy. At the leaf of the tree, the algorithm outputs the answer based on the answers of all the queries.

Goyal, et al. [24] showed that any gnd-tree with noise parameter ε that computes the identity function must have depth $\Omega(\varepsilon^2 \cdot n \log n)$. They also proved a simulation theorem that reduces the noisy broadcast model to the gnd-tree model. This theorem will be discussed in details later.

It remains to be an interesting question that whether we can prove any superlinear lower bounds for functions with boolean output in the gnd-tree model. If so, we may also prove superlinear lower bounds in the noisy broadcast model using the simulation theorem in [24]. It turns out that some interesting functions can be computed by linear depth gnd-tree.

For example, consider MAJ, the majority function. If we query MAJ on an noisy copy of x, we get the correct answer with probability $1/2 + \Omega(1/\sqrt{n})$. By Chebyshev's Inequality, the majority of n majority queries on different noisy inputs will output the correct answer with probability 2/3. (In fact, [24] gives a gnd-tree that can compute the hamming weight with linear depth using a similar idea) This algorithm can also be adapted to give an algorithm for the noisy broadcast model that computes MAJ with $O(n)$ broadcasts.

However, one may criticise this algorithm for the following reason: the correctness of this algorithm heavily relies on the assumption that the error rate on each coordinate is known and never changes in the computation. The algorithm may fail if the error rate on some coordinates are decreased. For example, suppose for each coordinate, when the true input is 0, we decrease the error to 0; when the true input is 1, the error rate remains to be ε. Then the algorithm may give the wrong answer. In practice, an algorithm whose correctness depends on the presence of a predictable noise rate is unsatisfactory. This restriction on algorithms is formalized by the notion of *noise cancellation adversary.* The noise cancellation adversary can be viewed as an evil player that has full knowledge and has the power to eliminate any noise but can not introduce additional noise. This type of adversary was introduced by Feige and Kilian [20] for the noisy broadcast model and they gave a linear time algorithm for OR that is robust under this kind of adversary.

Gallager's $O(n \log \log n)$ algorithm for ID [23] in the broadcast model is still valid under noise cancellation adversary. However, the algorithm we just described for MAJ doesn't work under noise cancellation adversary for either the gnd-tree or the noisy broadcast and we want to rule out such algorithms. It is an interesting question to ask how powerful the gnd-tree model is under the noise cancellation adversary.

We propose that the MAJ is hard under the noise cancellation adversary, i.e. it has an $\Omega(n \log n)$ lower bound in the gnd-tree model and $\Omega(n \log \log n)$ lower bound under the noisy broadcast model. We currently don't know how to prove this, or any superlinear lower bounds for boolean functions with single bit output under the noise cancellation adversary. We notice that $\Omega(n \log n)$ is also the lower bound of MAJ in noisy decision tree model [19]. In [39], this lower bound was proved by first proving an

$\Omega(nlogn)$ lower bound on the OR function for non-adaptive noisy decision trees, and then extending the ideas to prove the same bound for MAJ in the adaptive model. Following this approach, here we will show that the $\Omega(nlogn)$ lower bound for OR for non-adaptive noisy decision trees extends to non-adaptive generalized noisy decision trees. This result implies an $\Omega(n \log \log n)$ lower bound result for a special kind of noisy broadcast model which we call the *2-Phase noisy broadcast* model.

3.1.2 Our Result

We prove the following theorem:

Theorem 3.1.1. *Suppose T is a non-adaptive gnd-tree that can compute OR correctly with probability > 0.9 under any noise cancellation adversary with noise parameter ε. Then the depth of T is $\Omega(\varepsilon^4 \cdot n \log n)$.*

When ε is a constant, this lower bound meets the lower bound for OR under non-adaptive noisy decision tree model. [19]

This result implies a lower bound result for a special kind of noisy broadcast model which we call the 2-Phase noisy broadcast model. In this model, all the broadcasts are made by two phases: in the Phase 1, each processor P_i broadcasts its bit x_i some pre-specified number which may depend on i. Then in the Phase 2, instead of broadcasting messages to everyone, each processor can only send noisy messages to the receiver P_0. In other words, the broadcasts made by each processors in the 2nd Phase may only depend on its input and all the broadcasts it received in the 1st Phase. While this model is weaker than the original noisy broadcast model, some interesting protocols including the $O(n \log \log n)$ protocol for ID fit this model.

We will prove the following corollary:

Corollary 3.1.2. *Let $\mathcal{P}$ be a protocol in 2-Phase noisy broadcast model. Then if $\mathcal{P}$ can compute OR correctly with probability > 0.9 under any noise cancellation adversary with noise parameter ε, the number of broadcasts in $\mathcal{P}$ is at least $\Omega(\log(1/\varepsilon)^{-1} \cdot n \log \log n)$.*

For now, we will focus on the gnd-tree model and prove Theorem 3.1.1. In Section 3.7.3, we will provide the formal definition of the noisy broadcast model together with a simulation theorem similar to one given by Goyal, et al. [24], and use these to prove Corollary 3.1.2.

3.2 Preliminaries for Gnd-tree and Adversary

In this section we introduce the definition of the noisy decision tree model and generalized noisy decision tree model. Also, we give the formal definition of noise cancellation adversary.

3.2.1 Noisy Bits

Let $\varepsilon \in (0, 1/2)$. An ε-noisy bit is a random variable N that is 1 with probability ε and 0 otherwise. If $b \in \{0, 1\}$, an ε-noisy copy of b is $b \bigoplus N$, where N is an ε-noisy bit and $\bigoplus$ denotes the sum modulo 2.

For $n \in \mathbb{N}$, $N(n, \varepsilon)$ denotes a vector of n independent ε-noisy bits. For vector $x \in \{0, 1\}^n$, an ε-noisy copy of x is the random variable $x \bigoplus N(n, \varepsilon)$. We will omit the ε when the error parameter is fixed in the context.

3.2.2 Noisy Decision Tree

First let's recall the definition of decision tree:

Definition 3. *A decision tree T for input $x = (x_1, ..., x_n) \in \{0, 1\}^n$ is a binary tree, in which each internal node is labelled by an index i, together with an output function that takes all the leaves of the binary tree as input. The 2 out going edges of a non-leaf node v are labeled by value* 0 *and* 1*. The computation starts at the root. At a non-leaf node with label x_i the value of this variable is queried and according to the answer received the corresponding edge is chosen to the next node. This defines a unique path to a leaf, the value of the output function on that leaf is the result of the computation.*

In an *ε-noisy decision tree (nd-tree)* , when the tree queries x_i, instead of getting the true value of x_i, the tree only gets an ε-noisy copy of x_i. If x_i is queried multiple times, each time the tree will get an independent fresh ε-noisy copy of x_i. Then instead of reaching a particular leaf, the computation procedure will end up with a random variable supported on all the leaves that depends on the noise. The final output will be a random variable as well. For $x \in \{0,1\}^n$, we use $T(x)$ to represent the distribution of the output when the input is x. Similarly, for any distribution σ supported on $\{0,1\}^n$, we define $T(\sigma)$ to be the output distribution when the input is picked with respect to σ.

We say a noisy decision tree computes a function f with success probability $1-\delta$ if for all $x \in \{0,1\}^n$

$$\mathbb{P}_{y\sim T(x)}(y = f(x)) \geq 1-\delta$$

For two trees T_1 and T_2, we define the composition of T_1 and T_2 to be a new tree that links a copy of T_2 to each leaf of T_1. i.e. the tree runs T_1 and then runs T_2. We use $T = T_1 \circ T_2$, or simply $T = T_1, T_2$, to represent the composition.

We allow the decision tree to be randomized, which means we can have a family of trees and pick one to run under an arbitrary distribution independent to all the noise. An alternative way to define randomized decision tree is to allow the tree to have random nodes, which generate independent random bits to decide to go left or right. Random nodes are not counted in the depth of the computation. It is well known and easy to see both two definitions are equivalent. For convenience, we also allow the output function to be randomized.

The complexity of an nd-tree can be measured by the expected number of queries it made. We define $D_E(T,x)$ to be the expected number of queries made by T when the input is x. Note that random nodes don't count as queries here. Also we define $D_E(T) = D_E(x,\mathbf{0})$, where $\mathbf{0}$ denotes the all zero vector.

A decision tree is called non-adaptive if the sequence of coordinates it queries does not depend on the results of previous queries (but may depend on random bits). Intuitively, the adaptive decision tree should have more power than the non-adaptive decision tree. This is indeed the case for some problems. For example the function OR has a $O(n\log n)$ lower bound in non-adaptive nd-tree model [39], however, it can be computed by an adaptive nd-tree with linear depth [19].

In the proof of the main result we will use the following special nd-tree model which is called the *semi-adaptive nd-tree* model. Intuitively, a semi-adaptive nd-tree is a composition of several adaptive nd-trees which query each coordinate at most once. Formally speaking:

Definition 4. *A semi-adaptive noisy decision tree T is the composition of a family of adaptive noisy decision trees $(T_i)_{i\in[l]}$, where each tree T_i queries each variable at most once. Thus $T = T_1, T_2, ..., T_l$.*

We will later show that semi-adaptive nd-tree is essentially as powerful as non-adaptive nd-tree in computing function OR.

3.2.3 Generalized Noisy Decision Tree

The notation of generalized noisy decision tree (gnd-tree) is introduced by [24]. In a generalized noisy decision tree T, each non-leaf node v is associated with a boolean function q_v. For each input $x \in \{0,1\}^n$, the algorithm gets a collection $\{x \bigoplus N^i(n,\varepsilon)\}_{1\leq i\leq l}$ of independent ε-noisy copies of x. Usually we view collections of noisy vectors as an l by n matrix N_T; similarly we use $x \bigoplus N_T$ to refer to the matrix, called the noisy input matrix, whose rows are the distinct noisy copies of x During the computation, at each non-leaf node v, the tree is allowed to chose a noisy copy $x \bigoplus N^i(n,\varepsilon)$ and compute $q_v(x \bigoplus N^i(n,\varepsilon))$, then it chooses an edge to go based on the answer of this query. One noisy copy can be used multiple times in a single path.

As with nd-trees, we allow gnd-trees to be randomized. Similarly we can define adaptivity, depth and composition for gnd-trees. When we do composition of two gnd-trees, we assume the noisy bits they use are independent of each other.

The notation of the noisy input matrix gives us an alternative way to evaluation process of an nd-tree, which will be used later in our proof. Recall the noisy input matrix of a gnd-tree is a matrix such that each row is an independent noisy copy of the true input x. Thus the i-th column of the matrix is a sequence of noisy copies of x_i. Then those bits can be used as answers of the queries to x_i in the nd-tree.

Formally speaking, we say an nd-tree T runs on the noisy input matrix $M = x \bigoplus N$

if: for each non-leaf node v of T, assume v queries x_i, then the answer of the query is given by $M_{j,i}$ for some index j that depends on v; in each root-to-leaf path of T, no position in M is used more than once. It is easy to see this evaluation process will get the exact same result as the one in the original definition of nd-tree

3.2.4 Noise Cancellation Adversary

Intuitively speaking, a noise cancellation adversary is an evil player who wants to fool the algorithm. It has full knowledge of the true input and all the noise and the randomness. However, it is allowed to cancel noise but not introduce new noise. This means, while executing the algorithm, whenever there is an error caused by noise of the input or noise in the communication, the noise cancellation adversary can correct this noise. Formally speaking:

Definition 5. *Suppose $\mathcal{P}$ is a computational model that has noise in its computation. Let $N \in \{0,1\}^k$ be the noisy bits it gets during the computation, $\sigma \in \{0,1\}^l$ be the internal random bits it used. Then a noise cancellation adversary is defined by a function $\mathcal{A}(x, N, \sigma) = N'$. The image N' is a random variable that takes values in $\{0,1\}^k$ that satisfies $N' \preceq N$. To run $\mathcal{P}$ under noise cancellation adversary $\mathcal{A}$ means run $\mathcal{P}$ using random bits σ and with noise vector N'. The output distribution of T with input x under adversary $\mathcal{A}$ is represented by $T(x, \mathcal{A})$.*

We will simply use the word adversary to refer to the noise cancellation adversary.

In this definition, the adversary knows all the internal randomness of the protocol and its decision may depend on that. However, it may not change those random bits. We use the word ”noisy bits” to be those random bits that can be corrected by the adversary and the word ”random bits” to be those internal random bits that cannot be change by the adversary.

Under the above notation, we say a gnd tree T can compute function f under adversary with success probability $1 - \delta$ if $\mathbb{P}_{y \sim T(x,\mathcal{A})}(y = f(x)) \geq 1 - \delta$ holds for any input x and adversary $\mathcal{A}$.

3.3 Proof Overview of the Main Theorem

Assume T is a non-adaptive gnd-tree that computes OR. In order to prove a lower bound for the depth of T, we will analyze the behaviours of T under the following two types of input: μ_0 is the distribution concentrated on $\mathbf{0}$ and μ_1 is the uniform distribution on $\{e_i : i \in [n]\}$, where e_i is the unit vector on the i-th direction. We will construct two adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ together with an non-adaptive nd-tree T' with depth $O(1/\varepsilon^4 \cdot D(T))$ such that, the distributions $T(\mu_0, \mathcal{A}_0)$ and $T(\mu_1, \mathcal{A}_1)$ are close to $T'(\mu_0)$ and $T'(\mu_1)$. Thus T' can compute OR as well. The result in [39] shows that any non-adaptive nd-tree that separates this two distributions must have depth $\Omega(\log(1/\varepsilon)^{-1} \cdot n \log n)$, which proves our result.

More precisely, we will prove the following lemma:

Lemma 3.3.1. *Let μ_0 and μ_1 be distributions defined above. For any non-adaptive gnd-tree T, there exist adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ and a non-adaptive nd-tree T', such that:*

- $D(T') = O(1/\varepsilon^3 \cdot D(T))$
- $\|T(\mu_0, \mathcal{A}_0) - T'(\mu_0)\|_1 = 0$
- $\|T(\mu_1, \mathcal{A}_1) - T'(\mu_1)\|_1 \leq 0.01$

Where $\|\cdot\|_1$ is the L_1 distance between two distributions.

This lemma directly implies the main theorem:

Proof of Theorem 3.1.1. If T can compute OR correctly with probability at least 0.9 under any adversary, then Lemma 3.3.1 implies there exist an nd-tree T' such that T' can compute OR correctly with probability at least 2/3 when the input distribution is μ_0 or μ_1.

Theorem 3.6.1, which is adapted from [39] and proved in Section 3.6, shows that any non-adaptive nd-tree that computes OR must have depth at least $\Omega(\log(1/\varepsilon)^{-1} \cdot n \log n)$. This implies $D(T) = \Omega(\varepsilon^4 \cdot n \log n)$. □

We say an nd-tree T' simulates a gnd-tree T for some particular distribution σ if there exist an adversary $\mathcal{A}$ for T such that $T(\sigma, \mathcal{A}) = T(\sigma)$. Then one possible way to prove Lemma 3.3.1 is to construct an nd-tree T' that simulates T for both μ_0 and μ_1. Before we go further, we will state this property in an alternative way which is easier for us to work on.

First we make the following definition:

Definition 6. *Let T be a gnd-tree. We say T is confusable for distributions σ_0 and σ_1 if there exist adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ such that $T(\sigma_0, \mathcal{A}_0) = T(\sigma_1, \mathcal{A}_1)$*

If a gnd-tree T is confusable for σ_0 and σ_1, then we can use an empty nd-tree to simulate it for both σ_0 and σ_1. The empty nd-tree consists of a single node, which is labeled by a random variable with distribution $T(\sigma_0, \mathcal{A}_0) = T(\sigma_1, \mathcal{A}_1)$.

It is natural to consider the property of confusability conditioned on some information we know about the input matrix.

Definition 7. *Let T be a gnd-tree with input matrix $M = X \bigoplus N_T \in \{0,1\}^{l\times n}$, σ_0 and σ_1 be two input distributions. Let $K \subset l \times n$ be a subset of positions in M, we define $M_K \in \{0,1\}^K$ to be the boolean vector representing the values of M on those positions in K.*

For any partial assignment $\rho \in \{0,1\}^K$, we say T is confusable for μ_0 and μ_1 conditioned on ρ if there exist adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ such that:

$$T(\sigma_0, \mathcal{A}_0)|(M_K = \rho) = T(\sigma_1, \mathcal{A}_1)|(M_K = \rho)$$

Suppose a partial assignment fixes all bits, i.e. $K = l \times n$. Then for any gnd-tree T that runs on M, the output of T is determined by ρ and the adversary, regardless of what the true input is. Thus T is confusable conditioned on ρ for any two distributions. More generally, if the partial assignment ρ satisfies $\sigma_0|(M_K = \rho) = \sigma_1|(M_K = \rho)$, then T is confusable conditioned on ρ for σ_0 and σ_1.

As we mentioned earlier, the ordinary nd-tree model can be viewed as a special case of the generalized nd-tree model, whose queries are all single bit queries to the noisy input matrix. This viewpoint allows us to make the following definition:

Definition 8. *Suppose we have a gnd-tree T and an nd-tree T' that run on the same noisy input matrix M. We say T is confusable conditioned on T' for distributions σ_0*

and σ_1 if: for any leaf v in T', let ρ_v be the partial assignment determined by the bits in M that get queried by the path to v, then T is confusable for σ_0 and σ_1 conditioned on ρ_v.

Proposition 3.3.2. *Suppose T is a gnd-tree and T' an nd-tree defined on the same noisy input matrix. Suppose that σ_0 and σ_1 are two input distributions such that T conditioned on T' is confusable for σ_0, σ_1. Then there is a labeling of the leaves of T' by distributions over output values such that T' with this leaf labeling simulates T for both input distributions σ_0 and σ_1.*

Proof. We need to (1) assign to each leaf v of T' a probability distribution $\psi(v)$ over output values, and (2) define adversaries B_0 and B_1 such that on input distribution σ_0, $T'(\sigma_0) = T(\sigma_0, B_0)$ and $T'(\sigma_1) = T(\sigma_1, B_1)$. Under the hypothesis, for every leaf v of T, there are adversaries $A_0(v)$ and $A_1(v)$ such that the two conditional output distributions $T(\sigma_0, A_0)|(T'(x) = v)$ and $T(\sigma_1, A_1)|(T'(x) = v)$ are the same. We define $\psi(v)$ to be this output distribution. We define the adversary B_0 (for T) as follows: given input x, simulate T' on x (without an adversary) to determine a leaf v of T'. Then apply adversary $A_0(v)$ to the noisy input matrix. We define B_1 similarly. Now it follows that $T'(\sigma_0) = T(\sigma_0, B_0)$ since both can be written as the average of $\psi(v)$ over leaves v of T' selected by applying T' to input chosen according to σ_0. Similarly we have $T'(\sigma_1) = T(\sigma_1, B_1)$. This finishes the proof. $\square$

Recall μ_0 is the distribution concentrated on $\mathbf{0}$ and μ_1 is the uniform distribution on $\{e_i : i \in [n]\}$. The above analysis shows that if we can find a non-adaptive nd-tree T' such that $D(T') = O(1/\varepsilon^3 \cdot D(T))$ and T is confusable conditioned on T', we can use this T' to prove Lemma 3.3.1. However, we don't know how to achieve that. Instead, we can construct an nd-tree T'' with small expected depth such that T is confusable conditioned on T''. Though T'' is adaptive, it turns out this adaptive nd-tree satisfies the semi-adaptive property defined in Section 3.2.2. We will further show that semi-adaptive decision trees is not much more powerful than non-adaptive decision trees. In conclusion, we will prove Lemma 3.3.1 by two steps:

Lemma 3.3.3. *For any gnd-tree T, there exist a semi-adaptive nd-tree T' with expected depth $D_E(T') = O(1/\varepsilon^3 \cdot D(T))$ such that T is confusable conditioned on T' for μ_0 and μ_1.*

Lemma 3.3.4. *Let T be a semi-adaptive nd-tree, then there exist a non-adaptive nd-tree T' with depth $D(T') = O(D_E(T))$ such that:*

- $\|T(\mu_0) - T'(\mu_0)\|_1 = 0$
- $\|T(\mu_1) - T'(\mu_1)\|_1 \leq 0.01$

We will prove Lemma 3.3.3 in Section 3.4 and Lemma 3.3.4 in Section 3.5. With the help of these two lemma, the proof of 3.3.1 is straightforward:

Proof of Lemma 3.3.1. Suppose T is a non-adaptive gnd-tree that computes OR correctly under adversary with probably at least 2/3. Let T_1 be the semi-adaptive nd-tree we get by applying Lemma 3.3.3. By Proposition 3.3.2 we know that there exist a way to label the leaves of T_1 such that T_1 can simulate T for distributions μ_0 and μ_1, i.e.$\|T(\mu_0) - T_1(\mu_0)\|_1 = 0$ and $\|T(\mu_1) - T_1(\mu_1)\|_1 = 0$.

Apply Lemma 3.3.4 to T_1 we get an non-adaptive nd-tree T_2 with $\|T_1(\mu_0) - T_2(\mu_0)\|_1 = 0$ and $\|T_1(\mu_1) - T_2(\mu_1)\|_1 = 0$.

Combining these together we have:

- $\|T(\mu_0) - T_2(\mu_0)\|_1 = 0$
- $\|T(\mu_1) - T_2(\mu_1)\|_1 \leq 0.01$

Also we know $D_E(T_1) = O(1/\varepsilon^3 \cdot D(T))$ from Lemma 3.3.3. From Lemma 3.3.4 we know $D(T_2) = O(D_E(T_1))$. That gives $D(T_2) = O(1/\varepsilon^3 \cdot D(T))$, which finishes the proof. □

3.4 From non-adaptive gnd-tree to semi-adaptive nd-tree

In this section we prove Lemma 3.3.3.

3.4.1 Proof Outline

Let T be the non-adaptive nd-tree we are working on. Since T is non-adaptive, it can be viewed as a sequence of queries. Thus, we may change the order of the queries to put the queries on the same noisy copy together. In other words, we can always assume T has the following property: it makes a few queries to the first noisy copy, then goes to the second copy and never comes back. Under this setting, assuming T made d queries to a particular noisy copy, we can represent all these queries as a function $Q : \{0,1\}^n \rightarrow \{0,1\}^d$, where each coordinate of the output represents the output of the corresponding query. We call such a query Q *composite query* and call d the depth of Q.

Under this definition we can view T as a sequence of composite queries. Assume $T = Q_1, Q_2, ..., Q_l$, the first natural idea to try is to construct nd-tree T'_j for each Q_j such that Q_j is confusable conditioned on T'_j. Then we can compose all T'_j together to construct T'. We claim that Q conditioned on T is confusable for μ_0 and μ_1. This is not immediate, and requires some care. Let Q^j be the composition of $Q_1....Q_j$ and let T^j be the tree obtained by composing $T_1, ..., T_j$. We will prove by induction on j, that Q^j is confusable conditioned on T^j for μ_0 and μ_1. We would like to do this using that each Q_j is confusable conditioned on T_j for μ_0 and μ_1. When we try to show that confusability for μ_0 and μ_1 is preserved under composition we see that it does not work as one might hope. What does work is that confusability for two distribution α and β composes nicely provided that both α and β are concentrated on a single input. In our case μ_0 is concentrated on a single input but μ_1 is not. But μ_1 is the average of n distributions σ_i, where σ_i is concentrated on e_i. So the property that μ_0 is confusable with σ_i does behave well under composition. So we will prove by induction on j that for each i, that Q^j conditioned on T^j is confusable for μ_0 and σ_i. This motivates the following definition:

Definition 9. *For $i \in [n]$, let σ_i be the distribution concentrated on e_i. Suppose we have a gnd-tree T and an nd-tree T' that run on the same noisy input matrix M. We say T is strongly confusable conditioned on T' if for any $i \in [n]$, T is confusable conditioned* *on T' for μ_0 and σ_i.*

It is easy to see if T is strongly confusable conditioned on T', then T is confusable conditioned on T' for μ_0 and μ_1. Thus it is enough to construct a semi-adaptive nd-tree T' such that T is strongly confusable conditioned on T.

In the Section 3.4.2, we will show for any composite query Q, how we can construct an nd-tree T with small depth such that Q is strongly confusable conditioned on T. Then we will apply this result and prove Lemma 3.3.3 in Section 3.4.3.

3.4.2 Simulate a Single Composite Query

In this section we will show how to simulate a single composite query by a (possibly adaptive) nd-tree whose depth is at most a constant factor larger. Precisely speaking, we will prove the following lemma:

Lemma 3.4.1. *Let $Q : \{0,1\}^n \to \{0,1\}^d$ be a composite query. Then there exist an nd-tree T such that: $D_E(T) = O(1/\varepsilon^3 \cdot d)$ and Q is strongly confusable conditioned on T.*

Proof. For input $x \in \{0,1\}^n$, let $y = x \bigoplus N$ be the noisy input. Recall σ_i is the distribution that is concentrated on e_i. We will construct T in the following way:

- Let r be the root of the tree and mark it as an undetermined node.
- If there exist an undetermined node v, let $K \subset [n]$ be the set of coordinates queried from root to v and $\rho \in \{0,1\}^K$ be the answers. Define $Q|_v$ to be the function generated by restricting Q on ρ.
  - If $Q|_v$ is strongly confusable, mark v as a leaf node.
  - Else, pick $i \in [n]$ such that $Q|_v$ is not confusable for μ_0 and σ_i. Let v query the i-th coordinate. Then mark two children of v as undetermined nodes.
- Keep running until there is no undetermined node.

This procedure will stop because if T' queries all the coordinates, the restricted function will be a constant, thus $D(T) \leq n$. Also, we know that if T queries the i-th bit, then consider input distributions μ_0 and σ_i, the corresponding distributions of the noisy input matrix will be the same conditioned on the result of the query. Thus Q is confusable conditioned on T for μ_0 and σ_i. Together with the stopping condition it is easy to see Q is strongly confusable conditioned on T. Now we will try to give an upper bound for the depth of T.

First of all, we introduce some preliminaries about information theory.

For a discrete random variable X with probability mass function $p(x)$, the entropy of X is define by: $H(X) = -\sum_x p(x)\log p(x)$. It is easy to verify that $H(X) \le \log n$. Also, we use $H(p) = -p\log p - (1-p)\log(1-p), p \in [0,1]$ to denote the binary entropy function, which is the entropy of a biased coin that is 0 with probability p.

It is easy to verify the following properties of the binary entropy function. $H'(p) = \log(\frac{1-p}{p})$. $H''(p) = -\frac{1}{p} - \frac{1}{1-p} \le -4$. $H'''(p) = \frac{1}{p^2} - \frac{1}{(1-p)^2}$. $H'''(p) \ge 0, \forall p \in (0, 1/2]$; $H'''(p) \le 0, \forall p \in [1/2, 1)$

For two random variables X, Y with probability mass function $p(x, y)$, the conditional entropy of X given Y is defined by: $H(X|Y) = -\sum_{x,y} p(x, y)\log p(x|y) = \mathbf{E}_{y\sim Y} H(X|Y = y)$. The mutual information of X and Y is defined by: $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. Intuitively, mutual information measures the information that X and Y share. We will need the follow result about mutual information known as data processing inequality(Theorem 2.8.1 in [9]):

Theorem. *Suppose we have random variables X, Y, Z such that X and Z are independent conditioned on Y. Then we have:*

$$I(Y; X) \ge I(Z; X)$$

Now we are ready to prove the upper bound of $D_E(T)$. We introduce the following lemma:

Lemma 3.4.2. *Let $Q : \{0, 1\}^n \to \Sigma$ be any composite query. Suppose Q is not confusable for μ_0 and σ_i. Let X be the random variable that represents an ε-noisy copy of $\mathbf{0}$ and $Y = Q(X)$, then:*

$$I(Y; X_i) \ge C \cdot \varepsilon^3$$

Where C is a constant that doesn't depends on n.

Before we prove this lemma, let first see how this lemma implies the desired upper bound on the expected depth of T. Let T_k be the tree T restrict on depth k, i.e. T_k is obtained by deleting all the nodes in T with depth more than k. Let X be the random variable that represents an ε-noisy copy of $\mathbf{0}$. Let $T(X)$ ($T_k(X)$) be the random variable that represents the output leaf of T (T_k) when the input is X. Define the potential function Φ by:

$$\Phi(k) = C \cdot \varepsilon^3 \cdot D_E(T_k) + H\left(Q(X)|T_k(X)\right) - H(Q).$$

We claim that $\Phi(k) \le 0$ for all $k \in [n]$. First it is easy to see $\Phi(0) = 0$. Now let's consider the change of Φ when we increase k by 1. Let L_k be the set of non-leaf nodes in T at level k. For a non-leaf node v in T, let $\mathbb{P}(v|\mathbf{0})$ be the probability that the evaluation process passes through v when the input is $\mathbf{0}$. Let v^i be the index of the coordinate queried by v. By the definition of T we know $Q|_v$ is not confusable for μ_0 and σ_{v^i}. Then we have: $\forall k \in [n]$,

$$\begin{aligned}
\Phi(k-1) - \Phi(k) &= C \cdot \varepsilon^3 \cdot (D_E(T_{k-1}) - D_E(T_k)) \\
&\quad + H[Q(X)|T_{k-1}(X)] - H[Q(X)|T_k(X)] \\
&= C \cdot \varepsilon^3 \cdot (D_E(T_{k-1}) - D_E(T_k)) \\
&\quad + H[Q(X)|T_{k-1}(X)] - H[Q(X)|T_{k-1}(X), X_{v^i}] \\
&= \sum_{v \in L_k} \mathbb{P}(v|\mathbf{0}) \cdot \left(-C \cdot \varepsilon^3 + H[Q|_v(X)] - H[Q|_v(X)|X_{v^i}]\right) \\
&= \sum_{v \in L_k} \mathbb{P}(v|\mathbf{0}) \cdot \left(-C \cdot \varepsilon^3 + I[Q|_v(X); X_{v^i}]\right) \\
&\ge 0
\end{aligned}$$

The last inequality follows from Lemma 3.4.2. So we have $0 \ge \Phi(n) = C \cdot \varepsilon^3 \cdot D_E(T) + 0 - H(\tilde{Q})$, that implies $D_E(T) \le H(\tilde{Q})/(C \cdot \varepsilon^3) = O(1/\varepsilon^3 \cdot d)$. This finishes the proof. □

Proof of Lemma 3.4.2. Without loss of generality assume $\Sigma = [k]$ and $i = 1$. Define $Q^{-1}(i) = \{x \in \{0,1\}^n : Q(x) = i\}$.

Let τ_0 and τ_1 be the noisy distribution on $\{0,1\}^n$ centered at $\mathbf{0}$ and e_1. By hypothesis, Q is not confusable for μ_0 and σ_1. This means that for any choice of adversaries A_0 for μ_0 and A_1 for σ_1, the output distribution of $Q(\mu_0, A_0)$ And $Q(\mu_1, A_1)$ are different. To understand the implications of this, we first need to carefully describe the possible behaviors of adversaries A_0 and A_1. For τ_0, for $x \in \{0,1\}^n$, the probability of getting x as the noisy copy of $\mathbf{0}$ before adversary is $\tau_0(x)$. The adversary may cancel noise on some coordinates and output $y \preceq x$ as the noisy copy. Let w_{xy} be the probability that the original noisy copy is x and the adversary changes it to y. Note that the adversary A_0 is completely specified by the numbers w_{xy}. Then we know $\sum_{y:y\preceq x} w_{xy} = \tau_0(x)$. Also, the probability of getting y as the noisy copy after adversary is given by: $\sum_{x:x\succeq y} w_{xy}$. Similarly, we define t_{xy} to be the probability that the original noisy copy is x and the adversary change it to y when the true input is e_1. Note that in this case the adversary can change x to y only when $x \bigoplus e_1 \succeq y \bigoplus e_1$. Thus we define $\succeq_1$ such that $x \succeq_1 y$ if $x \bigoplus e_1 \succeq y \bigoplus e_1$.

Now we have represent the behaviours of adversaries for inputs $\mathbf{0}$ and e_1, thus the confusability of Q for μ_0 and σ_1 can be viewed as linear programming problem. Formally speaking, Q is not confusable for μ_0 and σ_1 if and only if the following linear programming doesn't have a feasible solution:

$$\begin{aligned}
max \quad & 0 \\
s.t. \quad & \sum_{y\in Q^{-1}(i)} \sum_{x:x\succeq y} w_{xy} - \sum_{y\in Q^{-1}(i)} \sum_{x:x\succeq_1 y} t_{xy} = 0, \quad \forall i \in [k] \\
& \sum_{y:y\preceq x} w_{xy} = \tau_0(x), \quad \forall x \in \{0,1\}^n \\
& \sum_{y:y\preceq_1 x} t_{xy} = \tau_1(x), \quad \forall x \in \{0,1\}^n \\
& w_{xy} \geq 0, \quad \forall y \preceq x \\
& t_{xy} \geq 0, \quad \forall y \preceq_1 x
\end{aligned}$$

The first group of k equations represents the requirement that for each i, $Q^{-1}(i)$ has the same probability under the two distributions as modified by the adversaries. Consider the dual of this linear programming, let α_i be the dual variable for each equation with respect to $Q^{-1}(i)$, β_x be the dual variable for equation with respect to

$\tau_0(x)$, γ_x be the dual variable for equation with respect to $\tau_1(x)$. By linear programming duality we know the original linear programming has no feasible solution is equivalent to the following dual programming having a negative minimum:

$$\begin{aligned} min \quad & \sum_x \beta_x \tau_0(x) + \sum_x \gamma_x \tau_1(x) \\ s.t. \quad & \alpha_{Q(y)} + \beta_x \geq 0, \quad \forall y \preceq x \\ & -\alpha_{Q(y)} + \gamma_x \geq 0, \quad \forall y \preceq_1 x \end{aligned}$$

For convenience we impose the additional condition that $|\alpha_i| \leq 1$ for each α_i, which does not change whether the minimum is negative.

$$\begin{aligned} min \quad & \sum_x \beta_x \tau_0(x) + \sum_x \gamma_x \tau_1(x) \\ s.t. \quad & \alpha_{Q(y)} + \beta_x \geq 0, \quad \forall y \preceq x \\ & -\alpha_{Q(y)} + \gamma_x \geq 0, \quad \forall y \preceq_1 x \\ & |\alpha_i| \leq 1, \quad \forall i \in [k] \end{aligned}$$

It is easy too see when the objective function reaches minimum, $\beta_x = -\min_{y:y \preceq x} \alpha_{Q(y)}$ and $\gamma_x = \max_{y:y \preceq_1 x} \alpha_{Q(y)}$. Then the objective function becomes a function of α_i's. We claim that the minimum is achieved when every α_i is at the boundary, i.e. $\alpha_i = \pm 1$, $\forall i \in [k]$. We establish the following lemma:

Lemma 3.4.3. *We say a function $f : \mathbb{R}^k \to \mathbb{R}$ is almost linear if there exist $A, B \subset 2^{[k]}$ which are two collections of subsets of $[k]$, such that f can be written as:*

$$f(x_1, ..., x_k) = \sum_{a \in A} c_a \max_{i \in a} x_i + \sum_{b \in B} d_b \min_{i \in b} x_i$$

Where $(c_a)_{a \in A}$'s and $(d_b)_{b \in B}$'s are constants. If f is almost linear, then for any domain with form $[a,b]^k \subset \mathbb{R}^k$, the minimum of f can be achieved when every x_i is at the boundary, i.e.

$$\min_{(x_1,...,x_k) \in [a,b]^k} f(x_1, .., x_k) = \min_{(x_1,...,x_k) \in \{a,b\}^k} f(x_1, .., x_k)$$

Proof. Assume the minimum of f on $[a,b]^k$ is achieved at $(y_1, ..., y_k)$, we will show that we can "move" all variables to the boundary without increasing the value of f.

Suppose the order of those variables is $a \leq y_{\sigma(1)} \leq y_{\sigma(2)} \leq ... \leq y_{\sigma(k)} \leq b$. If every variable equals to either a or b then we are done, else assume $a < y_{\sigma(i)} < b$. Further more we may assume $y_{\sigma(i-1)} < y_{\sigma(i)} = y_{\sigma(i+1)} = ... = y_{\sigma(j)} < y_{\sigma(j+1)}$. (Here we let $a = y_{\sigma(0)}$ and $b = y_{\sigma(k+1)}$ for convenience.)

Consider the following function $\tilde{f}(y) = f(z_1, ..., z_k)$ where $z_l = y$ if $i \leq l \leq j$ and $z_l = y_l$ otherwise. Then it is easy to see $\tilde{f}$ is linear on $[y_{\sigma(i-1)}, y_{\sigma(j+1)}]$ and $\tilde{f}(y_{\sigma(i)}) = f(y_1, ..., y_k)$. That means we can move y to one of the end points without increasing the value of $\tilde{f}$. This is equivalent to saying we change the value of $y_{\sigma(i)}, y_{\sigma(i+1)}, ..., y_{\sigma(j)}$ together to $y_{\sigma(i-1)}$ or $y_{\sigma(j+1)}$ without increasing the value of f.

By repeating this procedure we can move every y_i to the boundary, i.e. $y_i = a$ or b for every $i \in [k]$. $\square$

Applying this lemma directly we have $\alpha_i = \pm 1$, $\forall i \in [k]$ when the objective function achieves the minimum. Then we know:

$$\beta_x = \begin{cases} -1 & \text{if } \alpha_{Q(y)} = 1 \;\; \forall y \preceq x, \\ 1 & \text{otherwise.} \end{cases}$$

$$\gamma_x = \begin{cases} -1 & \text{if } \alpha_{Q(y)} = -1 \;\; \forall y \preceq_1 x, \\ 1 & \text{otherwise.} \end{cases}$$

Let $E_0 = \{x \in \{0,1\}^n : \beta_x = -1\}$ and $E_1 = \{x \in \{0,1\}^n : \gamma_x = -1\}$. Then the minimum of the objective function is:

$$\begin{aligned} \sum_x \beta_x \tau_0(x) + \sum_x \gamma_x \tau_1(x) &= \sum_{x \in E_0} -\tau_0(x) + \sum_{x \notin E_0} \tau_0(x) + \sum_{x \in E_1} -\tau_1(x) + \sum_{x \notin E_1} \tau_1(x) \\ &= -\tau_0(E_0) + \tau_0(\overline{E_0}) - \tau_1(E_1) + \tau_1(\overline{E_1}) \\ &= 1 - 2\tau_0(E_0) + 1 - 2\tau_1(E_1) \\ &= 2 \cdot (1 - (\tau_0(E_0) + \tau_1(E_1))) \end{aligned}$$

Recall that Q is not confusable for μ_0 and σ_1 if and only if this minimum is negative, which is equivalent to $\tau_0(E_0) + \tau_1(E_1) > 1$.

Now we will deduce some properties of E_0 and E_1. Then we use those properties to prove Lemma 3.4.2.

First, for any $x \in E_0$, we know $\alpha_{Q(y)} = 1 \ \ \forall y \preceq x$. This implies for any $z \preceq x$, $\beta_z = -1$, thus $z \in E_0$. In an other word, E_0 is a downset of $\preceq$. Similarly, we can conclude that E_1 is a downset of $\preceq_1$. For any $x_0 \in E_0$, $x_1 \in E_1$, we have: $\alpha_{Q(x_0)} = 1$ and $\alpha_{Q(x_1)} = -1$. This means $Q(x_0) \neq Q(x_1)$. That means for any $i \in \Sigma$, $Q^{-1}(i)$ cannot intersect both E_0 and E_1. This also implies $E_0 \cap E_1 = \emptyset$.

Also, we know both E_0 and E_1 are non-empty since $\tau_0(E_0) + \tau_1(E_1) > 1$. This implies E_0 and E_1 are subsets of the half spaces, i.e. $E_0 \subset H_0 = \{x \in \{0,1\}^n : x_1 = 0\}$, $E_1 \subset H_1 = \{x \in \{0,1\}^n : x_1 = 1\}$.

Now we are ready to prove the lemma. We define function $g : \Sigma \to \{0,1\}$ such that $g(i) = 0$ if $Q^{-1}(i) \cap E_0 \neq \emptyset$. Then we know $\forall x \in E_0$, $g(Q(x)) = 0$, $\forall x \in E_1$, $g(Q(x)) = 1$.

Recall X represents the random variable 0 plus noise, X_1 is the first coordinate of X, $Y = Q(X)$. Apply the data processing inequality on $X_1, Y, g(Y)$ we have: $I(Y; X_1) \geq I(g(Y); X_1)$. Thus it is enough to prove $I(g(Y); X_1) \geq C \cdot \varepsilon^3$ for some constant $C > 0$.

Now we try to compute $I(g(Y); X_1)$, we have:

$$\begin{aligned}
\mathbf{Pr}(g(Y) = 0 | X_1 = 0) &\geq \frac{\tau_0(E_0)}{1-\varepsilon} \\
\mathbf{Pr}(g(Y) = 0 | X_1 = 1) &\leq 1 - \frac{\tau_1(E_1)}{1-\varepsilon}
\end{aligned}$$

Let $a = \mathbf{Pr}(g(Y) = 0 | X_1 = 0)$ and $b = \mathbf{Pr}(g(Y) = 0 | X_1 = 1)$, we have: $a - b = \frac{\tau_0(E_0)+\tau_1(E_1)}{1-\varepsilon} - 1 \geq \varepsilon$. Now we can compute the mutual information of $g(Y)$ and X_i as a function of ϵ, a, b which we denote by $h(\varepsilon, a, b)$.

$$\begin{aligned}
h(\varepsilon, a, b) &= I(g(Y); X_1) \\
&= H(g(Y)) - H(g(Y)|X_1) \\
&= H((1-\varepsilon)a + \varepsilon b) - [(1-\varepsilon)H(a) + \varepsilon H(b)]
\end{aligned}$$

Now it is enough to prove there exist a constant $C > 0$ such that for all $\varepsilon > 0$:

$$\min_{a,b \in [0,1], a-b \geq \varepsilon} h(\varepsilon, a, b) \geq C \cdot \varepsilon^3 \tag{3.1}$$

Recall $H(p) = -p \log p - (1-p)\log(1-p)$ is the binary entropy function, this is a pure calculus problem. First we make the following observation:

Claim 3.4.4. *If there exists a constant $\varepsilon_0 > 0$ such that (3.1) holds for all $\epsilon < \varepsilon_0$ with constant $C > 0$. Then there exists a constant $C' > 0$ such that (3.1) holds for all $\epsilon \in (0, 1/2]$ with constant C'*

Proof. Since $h(\varepsilon, a, b) = H((1-\varepsilon)a+\varepsilon b) - [(1-\varepsilon)H(a) + \varepsilon H(b)]$ and H is strictly concave down, we know $h(a, b, \varepsilon) > 0$. Define $w(\varepsilon_0) = \min_{\{a,b\in[0,1],\varepsilon\in[\varepsilon_0,1/2]:a-b\geq\varepsilon\}} h(\varepsilon, a, b)$ we have $w(\varepsilon_0) > 0$. Then we have: $h(\varepsilon, a, b) \geq w(\varepsilon_0), \forall\varepsilon \in [\varepsilon_0, 1/2]$. Let $C' = \min(C, w(\varepsilon_0))$ we have: $h(\varepsilon, a, b) \geq C' \cdot \varepsilon^3, \forall\varepsilon \in (0, 1/2]$. □

Fix $\varepsilon > 0$, we further show $h(\varepsilon, a, b)$ achieves minimum when $a = b + \epsilon$ and $b \in [1/3, 1/2]$. In other words:

$$\min_{a,b\in[0,1],a-b\geq\varepsilon} h(\varepsilon, a, b) = \min_{b\in[1/3,1/2]} h(\varepsilon, b+\varepsilon, b)$$

To see this, first take partial derivative in term of a we get:

$$\frac{\partial}{\partial a} h(\varepsilon, a, b) = (1-\varepsilon)\big(H'((1-\varepsilon)a + \varepsilon b) - H'(a)\big)$$

Since $H'(\cdot)$ is strictly decreasing and $a - b \geq \epsilon$, we have $\frac{\partial}{\partial a}h(\varepsilon, a, b) > 0$. Thus we know $a = b + \varepsilon$ when $h(\varepsilon, a, b)$ achieves minimum. Plug it in we have:

$$h(\varepsilon, b+\varepsilon, b) \quad = \quad H(b + \varepsilon - \varepsilon^2) - [(1-\varepsilon)H(b+\varepsilon) + \varepsilon H(b)]$$

Take partial derivative in term of b we get:

$$\frac{\partial}{\partial b} h(\varepsilon, b+\varepsilon, b) \quad = \quad H'(b+\varepsilon-\varepsilon^2) - [(1-\varepsilon)H'(b+\varepsilon) + \varepsilon H'(b)]$$

Recall $H'''(p) = \frac{1}{p^2} - \frac{1}{(1-p)^2}$. We know H' is concave up on $(0, 1/2)$ and concave down on $(1/2, 1)$. Thus $\frac{\partial}{\partial b}h(\varepsilon, b+\varepsilon, b) \leq 0$ when $b \leq 1/2 - \varepsilon$ and $\frac{\partial}{\partial b}h(\varepsilon, b+\varepsilon, b) \geq 0$ when $b \geq 1/2$. Thus when $\varepsilon < 1/6$ we may assume $b \in [1/3, 1/2]$ when the function $h(\varepsilon, b+\varepsilon, b)$ achieves minimum.

In conclusion, we have: $\min_{a,b\in[0,1],a-b\geq\varepsilon} h(\varepsilon, a, b) = \min_{b\in[1/3,1/2]} h(\varepsilon, b+\varepsilon, b)$. Now it is enough to prove there exist $C > 0$ such that $h(\varepsilon, b+\varepsilon, b) \geq C \cdot \varepsilon^3$ for all $\varepsilon > 0$ and $b \in [1/3, 1/2]$.

Let's first consider the Taylor expansion of H at b up to the third term, define $f_b(\eta) = H(b+\eta) - \left[H(b) + H'(b)\cdot\eta + \frac{1}{2}H''(b)\cdot\eta^2 + \frac{1}{6}H'''(b)\cdot\eta^3\right]$. Then it is easy to see $f_b'(0) = 0$, $f_b''(0) = 0$ and $f_b'''(0) = 0$. Also we have $f_b^{(4)}(\eta) = H^{(4)}(b+\eta) = -2(\frac{1}{(b+\eta)^3} + \frac{1}{(1-b-\eta)^3})$. When $b \in [1/3, 1/2]$ and $\eta \in [0, 1/6]$, we have: $|f_b^{(4)}(\eta)| \le 2(3^3 + 3^3) = 108$. That implies $|f_b(\eta)| \le \frac{108}{4!}\cdot\eta^4 \le 5\cdot\eta^4$ for all $b \in [1/3, 1/2]$ and $\eta \in [0, 1/6]$.

Now we have:

$$\begin{aligned}
& h(\varepsilon, b+\varepsilon, b) \\
= \ & H(b+\varepsilon-\varepsilon^2) - [(1-\varepsilon)H(b+\varepsilon) + \varepsilon H(b)] \\
= \ & H(b) + H'(b)\cdot(\varepsilon-\varepsilon^2) + \frac{1}{2}H''(b)\cdot(\varepsilon-\varepsilon^2)^2 + \frac{1}{6}H'''(b)\cdot(\varepsilon-\varepsilon^2)^3 + f_b(\epsilon-\epsilon^2) \\
& -(1-\varepsilon)\cdot\left[H(b) + H'(b)\cdot\varepsilon + \frac{1}{2}H''(b)\cdot\varepsilon^2 + \frac{1}{6}H'''(b)\cdot\varepsilon^3 + f_b(\epsilon)\right] \\
& -\epsilon H(b) \\
= \ & H(b) + H'(b)\cdot(\varepsilon-\varepsilon^2) + \frac{1}{2}H''(b)\cdot(\varepsilon-\varepsilon^2)^2 + \frac{1}{6}H'''(b)\cdot(\varepsilon-\varepsilon^2)^3 + O(\varepsilon^4) \\
& -(1-\varepsilon)\cdot\left[H(b) + H'(b)\cdot\varepsilon + \frac{1}{2}H''(b)\cdot\varepsilon^2 + \frac{1}{6}H'''(b)\cdot\varepsilon^3 + O(\varepsilon^4)\right] \\
& -\epsilon H(b) \\
= \ & -\frac{1}{2}H''(b)\varepsilon^3 + O(\varepsilon^4)
\end{aligned}$$

The last inequality follows from the fact that $H'(b)$, $H''(b)$ and $H'''(b)$ are bounded by a constant when $b \in [1/3, 1/2]$. Since $H''(b) < -4$, we know there exists a constant C such that: $\forall b \in [1/3, 1/2]$, $\eta \in [0, 1/6]$.

$$\begin{aligned}
h(\varepsilon, b+\varepsilon, b) &\ge -\frac{1}{2}H''(b)\varepsilon^3 - C\cdot\varepsilon^4 \\
&\ge 2\varepsilon^3 - C\cdot\varepsilon^4
\end{aligned}$$

Thus when $\varepsilon < 1/C$, $h(\varepsilon, b+\varepsilon, b) > \varepsilon^3$.

Since $1/C$ is a constant that does not depend on b, by Claim 3.4.4 there exist $C' > 0$ such that for all $\epsilon > 0$, $a, b \in [0,1]$ that satisfy $a - b \ge \varepsilon$, $h(\varepsilon, a, b) \ge C'\cdot\varepsilon^3$. Thus $I(g(Y); X_1) \ge C'\cdot\varepsilon^3$, which finishes the proof.

$\square$

3.4.3 Proof of Lemma 3.3.3

Recall we may assume $T = Q_1, Q_2, ..., Q_l$ where each Q_j is a composite query that queries the j-th noisy copy. Applying Lemma 3.4.1 to every Q_j we get a sequence of nd-trees $\{T_j\}_j$ such that $D_E(T_j) = O(1/\varepsilon^3 \cdot d_j), \forall j \in [l]$, where d_j is the depth of Q_j. Also we known Q_j is strongly confusable conditioned on T_j for all j.

Define $T' = T_1, T_2, ..., T_l$, then it is easy to see T' is a semi-adaptive decision tree with $D_E(T') = O(1/\varepsilon^3 \cdot D(T))$. It is enough to show the strongly confusable property is preserved under composition. We state it as the following lemma:

Lemma 3.4.5. *Let T_1, T_2 be gnd-trees, T_1', T_2' be nd-trees such that T_1 (T_2) is strongly confusable conditioned on T_1' (T_2'). Then $T_1 \circ T_2$ is strongly confusable conditioned on $T_1' \circ T_2'$.*

Proof. For any $i \in [n]$, since T_1 is confusable conditioned on T_1' for μ_0 and σ_i, let $\mathcal{A}_1^0$ and $\mathcal{A}_1^i$ be the adversaries for μ_0 and σ_i that achieves confusability. Similarly for T_2 and T_2', let the two adversaries be $\mathcal{A}_2^0$ and $\mathcal{A}_2^i$. Then we know $T_1(\mu_0, \mathcal{A}_1^0) = T_1(\sigma_i, \mathcal{A}_1^i)$ and $T_2(\mu_0, \mathcal{A}_2^0) = T_2(\sigma_i, \mathcal{A}_2^i)$.

We define $\mathcal{A}_0$ ($\mathcal{A}_i$) for tree $T_1 \circ T_2$ in the follow way: apply $\mathcal{A}_1^0$ ($\mathcal{A}_1^i$) to the first noisy copy of the input and apply $\mathcal{A}_2^0$ ($\mathcal{A}_2^i$) to the second noisy copy.

Since the noise on the two noisy copies are independent, we have: $T_1 \circ T_2(\mu_0, \mathcal{A}_0) = T_1(\mu_0, \mathcal{A}_1^0) \times T_2(\mu_0, \mathcal{A}_2^0) = T_1(\sigma_i, \mathcal{A}_1^i) \times T_2(\sigma_i, \mathcal{A}_2^i) = T_1 \circ T_2(\sigma_i, \mathcal{A}_i)$ This implies $T_1 \circ T_2$ is strongly confusable conditioned on $T_1' \circ T_2'$. □

The above lemma directly implies T is strongly confusable conditioned on T', and thus confusable for μ_0 and μ_1. That finishes the proof.

3.5 From semi-adaptive nd-tree to non-adaptive nd-tree

In this section we will prove Lemma 3.3.4. First we restate the lemma here:

Lemma. *Let T be a semi-adaptive nd-tree, then there exists a non-adaptive nd-tree T' with depth $D(T') = O(D_E(T))$ such that:*

- $\|T(\mu_0) - T'(\mu_0)\|_1 = 0$
- $\|T(\mu_1) - T'(\mu_1)\|_1 \leq 0.01$

Recall μ_0 is the distribution concentrated on $\mathbf{0}$ and μ_1 is the uniform distribution on $\{e_i : i \in [n]\}$.

Proof. Assume $D_E(T) = d$, recall $D_E(T) = D_E(T, \mathbf{0})$ is the expected number of queries made by T when the input is $\mathbf{0}$. Define d_i to be the expected number of queries on the i-th coordinate when the input is $\mathbf{0}$. Thus we have $\sum_i d_i = d$. Define nd-tree T' to be the following: query the i-th coordinate $100d_i$ times. The depth of T' will be $D(T') = \sum_i 100d_i = 100d = O(D_E(T))$.

The output of T' is determined by the following: At each leaf of T', let $v_i \in \{0,1\}^{100d_i}$ be the result we get from T' by querying x_i. We will use this vector to simulate T. When T queries x_i, we use bits in v_i to answer those queries. If we run out of bits, i.e. T queries a coordinate x_i which has been queried more than $100d_i$ times, then we just generate a new random bit y which equals 1 with probability ε and 0 other wise, and use y as the answer. Finally we will reach some leaf of T, we will use the output with respect to that leaf as the output for T'.

When the input is $\mathbf{0}$, this simulation works perfectly since y is always equivalent to the noisy copy of x_i for all i. Thus $T(\mu_0) = T'(\mu_0)$. When the input is picked from μ_1, let X_1 be the random variable representing the input, the simulation only deviates when $X_1 = e_i$ and T queries x_i more than $100d_i$ times for some $i \in [n]$. We can define the bad event B to be:

$$B = \bigcup_{i\in[n]} B_i = \bigcup_{i\in[n]} \{X_1 = e_i \ \wedge \ T \text{ queries } x_i \text{ more than } 100d_i \text{ times}\}$$

Then we have: $\|T(\mu_1) - T'(\mu_1)\|_1 \leq \mathbb{P}(B)$. So it is enough to show B happens with small probability.

For any $i \in [n]$, recall $D_i(T, x)$ is the expectation of the number of times that coordinate x_i gets queried by tree T when the input is x. Thus $D_i(T, \mathbf{0}) = d_i$. We have the following important observation:

Lemma 3.5.1. *For any semi-adaptive decision tree T,*

$$D_i(T, e_i) = D_i(T, \mathbf{0}) \quad \forall i \in [n]$$

Proof. Recall the definition of semi-adaptive decision tree we may assume $T = T_1, ..., T_l$, where each T_j queries every coordinate at most once. So it is enough to prove $D_i(T_j, e_i) = D_i(T_j, \mathbf{0})$ for all i, j.

For any T_j, since $\mathbf{0}$ and e_i only differs on the i-th coordinate, they will behave exactly the same until the i-th coordinate gets queried. However, since T_j only queries the i-th coordinate at most once, the rest of the tree does not affect D_i at all, so $D_i(T_j, e_i) = D_i(T_j, \mathbf{0})$. □

With the help of Lemma 3.5.1 we know $D_i(T, e_i) = d_i \; \forall i \in [n]$. By Markov inequality,

$$\begin{aligned}
\mathbb{P}(B) &= \sum_{i \in [n]} \mathbb{P}(B_i) \\
&\leq \sum_{i \in [n]} \frac{1}{n} \cdot \frac{D_i(T, e_i)}{100 d_i} \\
&\leq \sum_{i \in [n]} \frac{1}{n} \cdot \frac{d_i}{100 d_i} \\
&= 0.01
\end{aligned}$$

That implies $\|T(\mu_1) - T'(\mu_1)\|_1 \leq 0.01$, which finishes the proof.

□

3.6 Lower bound for non-adaptive nd-tree

In this section we will provide the proof of the lower bound result given in [39]. The statement of the theorem will be modified for convenience.

Theorem 3.6.1. *Recall μ_0 is the distribution concentrated on $\mathbf{0}$ and μ_1 is the uniform distribution on $\{e_i : i \in [n]\}$. If T is a non-adaptive nd-tree that gives correct answer for μ_0 and μ_1 with probability at least $2/3$, then $D(T) = \Omega(\log(1/\varepsilon)^{-1} \cdot n \log n)$.*

Proof. Define set $S = \{i \in [n] :$ T queries coordinate i more than $20D(T)/n$ times $\}$. Then since T is non-adaptive it is easy to see $|S| \leq 0.05 \cdot n$.

Let T' be the decision tree that queries each coordinate $k = D(20D(T)/n)$ times. Then we can use T' to simulate T in the following way:

- If T queries x_i s.t. $i \notin S$, use the answer in T' as answer.
- If T queries x_i s.t. $i \in S$. use an independent ε noise as answer.

This simulation works perfectly for input $\mathbf{0}$ and e_j when $j \notin S$. That's because for any $i \in S$, an independent ε noise is indeed an ε-noisy copy for the i-th coordinate. That result implies: $\|T(\mu_0) - T'(\mu_0)\|_1 = 0$ and $\|T(\mu_1) - T'(\mu_1)\|_1 \leq |S|/n \leq 0.05$. Since T can give correct answer for μ_0 and μ_1 with probability at least $2/3$, We know:

$$\|T'(\mu_0) - T'(\mu_1)\|_1 \geq 2/3 - 2 \cdot 0.05 \geq 1/2$$

Now we will bound this difference from above in term of n and k.

Lemma 3.6.2. *Let $\gamma = (4\varepsilon(1-\varepsilon))^{-1}$ and suppose $k \leq \log n/2\log\gamma$. Then:*

$$\|T'(\mu_0) - T'(\mu_1)\|_1 \leq O(\sqrt{\frac{\log n}{n}}k^{2/3}\gamma^{4k})$$

Proof. First, we claim that it is enough to prove this inequality for deterministic nd-trees, since a randomized nd-tree is just a distribution over deterministic nd-trees. We represent each leaf u by the $k \times n$ matrix where u_{ij} is the value of x_i in copy j along the path to the leaf. Let $k_i(u) = \sum_{j\in[k]} u_{ij}$ to be the number of "1" answer for x_i. and let $z_i(u) = |\{i : k_i(u) = l\}|$ be the number of variables with exactly l "1"'s in the answer. Let K_i^0 (K_i^1) and Z_i^0 (Z_i^1) be the random variable for the value of k_i and z_i when the input is picked from μ_0 (μ_1). Furthermore, let

$$p_l = \mathbb{P}(\{K_i^0 = l\}) = \binom{k}{l}\varepsilon^l(1-\varepsilon)^{k-l}.$$

Then we have:

$$\begin{aligned}
& \|T'(\mu_0) - T'(\mu_1)\|_1 \\
= & \sum_{u\in\{0,1\}^{kn}} |\prod_{i\in[n]} \varepsilon^{k_i}(1-\varepsilon)^{k-k_i} - \frac{1}{n}\sum_{j\in[n]}(1-\varepsilon)^{k_j}\varepsilon^{k-k_j}\prod_{i\neq j}\varepsilon^{k_i}(1-\varepsilon)^{k-k_i}| \\
= & \sum_{u\in\{0,1\}^{kn}} \prod_{i\in[n]} \varepsilon^{k_i}(1-\varepsilon)^{k-k_i}|1 - \frac{1}{n}\sum_{j\in[n]}\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2k_j-k}| \\
= & \sum_z \mathbb{P}(Z^0 = z)|1 - \frac{1}{n}\sum_{l=0}^{k} z_l\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k}| \\
= & E\left(\left|1 - \frac{1}{n}\sum_{l=0}^{k} Z_l^0\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k}\right|\right) \\
= & E\left(\left|\sum_{l=0}^{k}(p_l - \frac{1}{n}Z_l^0)\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k}\right|\right) \\
\leq & \sum_{l=0}^{k} E\left(\left|(p_l - \frac{1}{n}Z_l^0)\right|\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k}\right) \\
\leq & \left(\sum_{l=0}^{k}\mathbb{P}\left(\left|Z_l^0 - p_l n\right| \geq \sqrt{3p_l n\log\frac{n}{p_l}}\right)\right. \\
& \left. + \sum_{l=0}^{k}\mathbb{P}\left(\left|Z_l^0 - p_l n\right| \leq \sqrt{3p_l n\log\frac{n}{p_l}}\right)\sqrt{\frac{3p_l\ \log(n/p_l)}{n}}\right)\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k}.
\end{aligned}$$

Note that $Z_l^0 \leq n$ and

$$\sum_{l=0}^{k} p_l\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k} = \sum_{l=0}^{k}\binom{k}{l}\varepsilon^{k-l}(1-\varepsilon)^l = 1.$$

Since Z_l^0 is binomially distributed with parameters n and p_l, using Chernoff bound we have for $\beta \leq \sqrt{p_l n/3}$

$$\mathbb{P}\left(|Z_l^0 - p_l m| \geq \beta\sqrt{3p_l n}\right) \leq 2e^{-\beta^2}$$

Pick $\beta = \sqrt{\log(n/p_l)}$ we have:

$$\begin{aligned}
& \|T'(\mu_0) - T'(\mu_1)\|_1 \\
\leq\ & \sum_{l=0}^{k} \frac{2p_l}{n}\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k} + \sum_{l=0}^{k}\sqrt{\frac{3p_l\log(n/p_l)}{n}}\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k} \\
\leq\ & \frac{2}{n} + \sqrt{\frac{3\log n}{n}}\sum_{l=0}^{k}\sqrt{p_l}\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k} + \sqrt{\frac{3}{n}}\sum_{l=0}^{k}\sqrt{p_l}\log\frac{1}{p_l}\left(\frac{1-\varepsilon}{\varepsilon}\right)^{2l-k} \\
\leq\ & \frac{2}{m} + \sqrt{\frac{3k\log n}{n}}\gamma^{2k} + \frac{6}{e}\sqrt{\frac{3}{n}}k^{2/3}\gamma^{4k} \\
\leq\ & O(\sqrt{\frac{\log n}{n}}k^{2/3}\gamma^{4k})
\end{aligned}$$

□

Applying this result directly we have:

$$1/2 \leq O(\sqrt{\frac{\log n}{n}}k^{2/3}\gamma^{4k})$$

Solving this inequality we have: $k = O(\log(1/\varepsilon)^{-1} \cdot \log n)$. Thus $D(T) = k \cdot n = O(\log(1/\varepsilon)^{-1} \cdot n\log n)$, which finishes the proof.

□

3.7 Noisy Broadcast Model and It's Relation to the Gnd-Tree Model

Now we have finished proving Theorem 3.1.1. In this section we provide the formal definition of the noisy broadcast model. Goyal, et al. [24] Showed how to reduce the noisy broadcast model to the gnd-tree model. We show how to adapt their reduction so that it applies when we introduce the noise cancellation into both models. At the end of this section, we will provide the proof of Corollary 3.1.2.

3.7.1 The Noisy Broadcast Model

Here we provide the definition given in [24]:

The noisy broadcast model considers one receiver P_0 and n processors $P_1, ..., P_n$. For input $x \in \{0,1\}^n$, the i-th coordinate x_i is given to processor P_i. The goal is for P_0 to evaluate a function f at x. The receiver and the processors are allowed to communicate under a noisy broadcast protocol.

The specification of a noisy broadcast protocol consists of:

- The total number s of broadcast used in the protocol.
- A sequence of indices $i^1, ..., i^s$ that indicate the processor that make the broadcast in each round.
- A sequence of boolean functions $g^1, ..., g^s$ that are used to compute the message of each broadcast. Here $g^j : \{0,1\}^j \to \{0,1\}$.
- An output function h for P_0 to compute the final output.

Running the protocol. For a fixed noisy parameter ε, let N be a s by $n+1$ matrix of independent ε-noisy bits and let N^j denotes the j-th row of N. In the j-th round of the execution, the processor P_{i^j} broadcast b^j and each other processor P_h receives a noisy copy of b^j given by $b^j_h = b^j \bigoplus N^j_h$. The bit b^j is computed by the function g^j using P_{i^j}'s input bit and all the previous noisy bits received by P_{i^j}, i.e. $b^j = g^j(x_{i^j}, b^1_{i^j}, ..., b^{j-1}_{i^j})$. Finally, the receiver P_0 will evaluate the output of the protocol $h(b^0_1, ..., b^0_s)$.

We allow the protocol to be randomized. This means that each processor has access to a source that generates independent random bits. The function g^j may depend on the random bits generated by P_{i^j}.

The receiver P_0 is not essential for this model. In some other definitions, there is no P_0 and the goal is for all the processors to learn the correct value of $f(x)$. It is easy to see these definitions are essentially the same.

This model enforces certain properties typically required of communication protocols in noisy environments. First, protocols must be oblivious in the sense that the sequence of processors who broadcast is fixed in advance and does not depend on the execution. Without this requirement, noise could lead to several processors speaking at the same time. Second, it rules out communication by silence: when it is the turn of a processor to speak, it must speak.

3.7.2 Reduction from the Noisy Broadcast Model to the Gnd-tree Model

In this section we give the formal statement of the simulation theorem with adversaries in both models. Also, we will discuss the behaviours of the result when the original protocol is in the 2-Phase noisy broadcast model. For convenience, we will restate their result in a slightly different but more general way.

For the gnd-tree model, recall the noisy copies are obtained by adding noise to every coordinate of the true input x with fixed probability ε. It will be convenient to consider the possibility of different noise rates for different input bits. For any vector $(\varepsilon_1, ..., \varepsilon_n)$ satisfying $\varepsilon_i \in (0, 1/2), \forall i$, we can generate noisy copies using ε_i as the noise rate on the i-th coordinate. We call this vector the noise vector of the noisy copy.

We define the following special kind of gnd-tree called *flexible noise gnd-tree* . A flexible noise gnd-tree T is a gnd-tree with a flexible noise parameter $\varepsilon \in (0, 1/2)$. We will use the word *uniform noise parameter* to refer to the fixed noise parameter used in the normal gnd-tree model. The algorithm starts by choosing any vector $(\varepsilon_1, ..., \varepsilon_n) \in (0, 1/2)^n$ such that the geometric mean of ε_i's is at least ε, which means $(\prod_i \varepsilon_i)^{1/n} \geq \varepsilon$. Then this vector is used as the noise vector to generate all the noisy copies. The rest of the evaluation is the same as the normal gnd-tree. Clearly, the algorithm can always choose $\varepsilon_i = \varepsilon, \forall i$, so the flexible noise gnd-tree model is at least as powerful as the original model.

Under this definition, we can state the reduction theorem as follow:

Theorem 3.7.1. *Suppose $\mathcal{P}$ is a noisy broadcast protocol that uses k broadcasts. If $\mathcal{P}$ can compute function f with error at most δ under any adversary, then there exists a flexible noise gnd-tree T of depth $2k + n$ with flexible noise parameter $\varepsilon^{k/n}$ such that, T can compute f with error at most δ under any adversary. What's more, if $\mathcal{P}$ is in the 2-Phase noisy broadcast model, then T is non-adaptive.*

The proof of this theorem is given in Section 3.7.4.

Remark: It is natural to ask whether we can replace the flexible noise gnd-tree by a normal gnd-tree in the above theorem. Unfortunately, this is not achievable even for the uniform noise parameter being $(\varepsilon^{k/n})^{O(1)}$. For example, consider the identity function on a subset of size $n/\log\log n$, using Gallager's algorithm [23] it can be computed in $O(n)$ broadcasts in the noisy broadcast model. However, it has a $\Omega(n \cdot \log n/\log\log n)$ lower bound by the result of [24] in the gnd-tree model when the error rate is constant.

The remark above implies that it is not possible to reduce a flexible noise gnd-tree with flexible noise parameter ε to a normal gnd-tree with uniform noise parameter $\varepsilon^{O(1)}$. The main reason is the noise parameter on some coordinate might be much smaller than $\varepsilon^{O(1)}$. What we can still hope for is once we fix the bits on those positions we can get a reduction result for the restricted function.

Before we state the reduction result we first introduce some definitions. For $K \subset [n]$, we refer to a point in $\{0,1\}^K$ as a partial assignment to K. Let ρ be a partial assignment to K and let x be a partial assignment to $[n]-K$ we write $x\rho$ as the point in $\{0,1\}^n$ that agrees with ρ on K and with x on $[n]-K$. For any function f that takes input on $[n]$, we define $f|_\rho$ to be the restriction of f on ρ, i.e. $f|_\rho(x) = f(x\rho)$ for $x \in \{0,1\}^{[n]-K}$.

Theorem 3.7.2. *Let $C > 1$ be any constant. Suppose T is a flexible noise gnd-tree of depth k and flexible noise parameter ε. Suppose T can compute function f with error at most δ under any adversary. Then there is a subset $K \subset [n]$ of size at most n/C such that for any partial assignment ρ to K, there is a gnd-tree T_ρ of depth k that takes input in $[n]-K$ with uniform noise parameter ε^C, such that T_ρ can compute $f|_\rho$ with error at most δ under any adversary.*

The idea of the proof is pretty simple: we choose K to be the set of coordinate that has noise parameter smaller than ε^C, then once we fixed a partially assignment on K, T becomes a flexible noise gnd-tree such that the noise parameters on every coordinates is greater than ε^C. The details of the proof is given in Section 3.7.5.

We now explain how Theorem 3.7.1 and Theorem 3.7.2 can be used to obtain lower bound results for the noisy broadcast model. If f is a function such that for any $K \subset [n]$ with $|K| \leq n/C$, there exist a partial assignment ρ to K such that $f|_\rho$ is hard in the gnd-tree model, then combine these two theorems we can obtain a hardness result for the noisy broadcast model.

For example, let f be the identity function ID_n, $\mathcal{P}$ be a protocol in the noisy broadcast model that can compute ID_n with k broadcasts. Applying theorems above it is easy to see for any set $K \subset [n]$ with $|K| \leq n/C$, we can pick any partial assignment ρ and the restricted function will become: $ID_{(1-1/C)n}$, which is almost as hard as the original function. As we mentioned before, [24] shows that any gnd-tree that computes ID_n with noise parameter ε must have depth $\Omega(\varepsilon^2 \cdot n \log n)$. Applying this result together with the two reduction theorems above we have:

$$2k + n = \Omega((\varepsilon^{C \cdot k/n})^2 \cdot (1 - 1/C)n \cdot \log((1 - 1/C)n))$$

Pick $C = 4$, we get $k = \Omega(\log(1/\varepsilon)^{-1} \cdot n \cdot \log \log n)$.

3.7.3 Preliminaries of the Proof

In this section we introduce some notation we are going to use in the proofs of Theorem 3.7.1 and Theorem 3.7.2.

For two randomized computational models $\mathcal{P}$ and $\mathcal{Q}$, we say $\mathcal{P}$ can be reduced to $\mathcal{Q}$ (or $\mathcal{Q}$ can simulate $\mathcal{P}$) if for any input x, the output distributions for the two models are the same. Then any function that can be computed by $\mathcal{P}$ can also be computed by $\mathcal{Q}$. We need to extend this definition to models with adversary.

Definition 10. *Assume $\mathcal{P}$ and $\mathcal{Q}$ are noisy computation models under noise cancellation adversary. We say $\mathcal{P}$ can be reduced to $\mathcal{Q}$ under noise cancellation adversary, or $\mathcal{Q}$ can simulate $\mathcal{P}$ under noise cancellation adversary if the following property holds:*

For any adversary $\mathcal{A}_1$ for $\mathcal{Q}$, there exists an adversary $\mathcal{A}_0$ for $\mathcal{P}$ such that for any input x, the output distributions $\mathcal{P}(x, \mathcal{A}_0)$ and $\mathcal{Q}(x, \mathcal{A}_1)$ are exactly the same.

This definition implies that if $\mathcal{P}$ can be reduced to $\mathcal{Q}$ and $\mathcal{Q}$ can compute some function f with error at most δ under any adversary, then $\mathcal{P}$ can compute f with error at most δ under any adversary as well.

Now let's set up a few computational models we are going to use in the proofs.

The semi-noisy broadcast model $SNB(\varepsilon)$ is similar to the noisy broadcast model. In this model there are n input processors $Q_1, ..., Q_n$, a receiver P_0 and a collection of auxiliary processors $\{P_i : i \in E \subset \{0,1\}^n\}$. Each Q_i initially has input x_i and it can only broadcast x_i with noise. An auxiliary processor P_i can broadcast any boolean function of the bits it heard previously, and this broadcast is noise-free. We say a protocol in SNB model is non-adaptive if for every auxiliary processor P_i, $i \in [n]$, the broadcasts it makes does not depend on broadcasts received from other auxiliary processors. In other words, they only depend on broadcasts received from input processors $Q_1, ..., Q_n$.

The flexible noisy-copy broadcast model $NCB(\varepsilon)$ has one receiver P_0 and a collection of auxiliary processors $\{P_i : i \in E \subset \{0,1\}^n\}$. Before the evaluation starts, the algorithm picks a vector $(\varepsilon_1, ..., \varepsilon_n) \in (0, 1/2)^n$ satisfying the condition $(\prod_i \varepsilon_i)^{1/n} \geq \varepsilon$. Then each of the P_i gets an independent noisy copy of the entire input. The noise parameters for the i-th coordinate is ε_i. Here ε is called the *flexible noise parameter* of the protocol. All broadcasts are noise-free. Similarly, for a protocol in NCB model, we say it is non-adaptive if for every auxiliary processor P_i, $i \in [n]$, the broadcasts it makes does not depend on previous broadcasts. Note that the final output broadcast made by the receiver P_0 will depend on both the noisy copy it gets and all the previous broadcasts.

We consider a sequence of protocols:

- $\mathcal{P}_1 := \mathcal{P}$ is the original broadcast model.
- $\mathcal{P}_2$ in the model $SNB(\varepsilon)$.
- $\mathcal{P}_3$ in the model $NCB(\varepsilon^{k/n})$.
- $\mathcal{P}_4 := T$ is a flexible noise gnd-tree with flexible noise parameter $\varepsilon^{k/n}$.

We will prove Theorem 3.7.1 by sequence of reductions from $\mathcal{P}_1$ to $\mathcal{P}_4$.

3.7.4 Proof of Theorem 3.7.1

Recall the notation we defined in Section 3.7.1, for the noisy broadcast protocol $\mathcal{P}$, we make the following definitions: for any $j \in [k]$.

- i^j is the index of the processor broadcasting at step j.

- b^j is the bit broadcast at step j.
- g^j is the boolean function that computes b^j.
- For $h \in [n]$, $b^j_h = b^j \bigoplus N^j_h$ is the noisy copy received by P_h.

From $\mathcal{P}_1$ to $\mathcal{P}_2$. We will show that $\mathcal{P}_1$ can be simulated by $\mathcal{P}_2$ under adversary. That means in addition to the normal simulation, we need to show that any adversary behaviour in $\mathcal{P}_2$ can be simulated by some adversary in $\mathcal{P}_1$.

We will assume that both models are using the same noisy bits N. Assume $\mathcal{P}_2$ already finished the simulation for the first $j-1$ rounds of $\mathcal{P}_1$. i.e. Each P_h in $\mathcal{P}_2$ knows the values $b^1_h, ...b^{j-1}_h$.

Then to simulate the j-th round of $\mathcal{P}_1$, Q_{i^j} broadcasts x_{i^j}, and P_{i^j} evaluates both $g^j(0, b^{i^j}_1, ..., b^{i^j}_{j-1})$ and $g^j(1, b^{i^j}_1, ..., b^{i^j}_{j-1})$ and broadcasts both of them with no noise.

Now each P_i must generate b^j_i by the three bits (x, a_0, a_1) received in this round, where x is the noisy broadcast and a_0 and a_1 are the bits sent by P_{i^j}. If $a_0 = a_1$, then P_i uses its private randomness to generate an ε biased bit and add it to b^j_i to simulate the noise. If $a_0 \neq a_1$, P_i simply sets b^i_j to a_x. In this case, if the adversary for $\mathcal{P}_2$ decides to cancel the noise, i.e. correct x back to the true value, x_{i^j}, then the adversary for $\mathcal{P}_1$ will also correct b^j_h back to b^j, which guarantees both model will end up with same b^j_h in the end.

The total number of broadcast in $\mathcal{P}_2$ is $3k$, of which k are by input processors and $2k$ are by auxiliary processors.

It remains to show if $\mathcal{P}_1$ is also a 2-Phase noisy broadcast model, then $\mathcal{P}_2$ is non-adaptive. We known when $\mathcal{P}_2$ is simulating Phase 1, when some processor P_i in $\mathcal{P}_1$ broadcasts x_i, $\mathcal{P}_2$ will simulate it by 3 broadcasts, 2 by P_i and 1 by Q_i. It is easy to see the 2 broadcasts made by P_i are fixed be 0 and 1, and can be ignored. This implies that when $\mathcal{P}_2$ is simulating Phase 2, each broadcast made by $\{P_i\}_{i\in[n]}$ only depends on broadcasts made by $\{Q_i\}_{i\in[n]}$. Thus $\mathcal{P}_2$ is non-adaptive in SNB model.

From $\mathcal{P}_2$ to $\mathcal{P}_3$. Here we will build a protocol $\mathcal{P}_3$ in the $NCB(\varepsilon^{k/n})$ model that simulates $\mathcal{P}_2$.

In $\mathcal{P}_2$, for any $i \in [n]$, let k_i be the number of times that Q_i broadcasts x_i. Then each processor will receive k_i bits of ε-noisy copies of x_i. In $\mathcal{P}_3$, set $\varepsilon_i = \varepsilon^{k_i}$ to be the noise parameter for coordinate i, this is allowed because $(\prod_i \varepsilon_i)^{1/n} = (\prod_i \varepsilon^{k_i})^{1/n} = \varepsilon^{\sum_i k_i/n} = \varepsilon^{k/n}$. Then at the start of $\mathcal{P}_3$, each processor P_j will get an ε_i-noisy copy of x_i, it is enough to show that we can use this noisy bit to construct a sequence of k_i ε-noisy bits of x_i. Then we can use this sequence to simulate all the broadcasts made by Q_i and thus simulate protocol $\mathcal{P}_3$.

First, we will review the proof in [24], where there is no adversary in both protocols. Let $t = k_i$ and $\gamma = \varepsilon^t$. Then our goal is to generate t independent ε noisy copies of x_i using one γ-noisy copy of x_i. Let p_0 (resp. p_1) be the distribution on $\{0,1\}^t$ be of t independent ε-noisy copies of 0 (reps. 1). Those are the distributions we want to get when x_i is 0 and 1 respectively. Let b be the γ-noisy copy of x_i given to the algorithm. We may describe the algorithm by: If $b = 0$, output $s \in \{0,1\}^t$ with respect to distribution q_0; if $b = 1$, output $s \in \{0,1\}^t$ with respect to distribution q_1. q_0 and q_1 need to satisfies the following requirement: when $x_i = 0$, $(1-\gamma)q_0 + \gamma q_1 = p_0$; similarly when $x_i = 1$, $(1-\gamma)q_1 + \gamma q_0 = p_1$. Solve these equations we have:

$$\begin{aligned} q_0(s) &= \frac{(1-\gamma)p_0(s) - \gamma p_1(s)}{1-2\gamma}, \\ q_1(s) &= \frac{(1-\gamma)p_1(s) - \gamma p_0(s)}{1-2\gamma}. \end{aligned}$$

It is easy to show both of them are indeed distributions.

Now consider the reduction under adversary. Again we focus on simulating the ε-noisy broadcasts of length t using one γ-noisy copy of x_i. Assume $x_i = 0$, let N_γ be the noise used in $\mathcal{P}_3$, then the above algorithm will sample a string according to q_0 if $N_\gamma = 0$ and according to q_1 if $N_\gamma = 1$. One thing the adversary can do is to cancel the noise with some fixed probability, then the final output distribution will be $(1-\gamma')q_0 + \gamma' q_1$ for some $\gamma' < \gamma$. However, the adversary can do more than this. Recall the adversary can see all the random bits, even random bits in the future! The adversary can decide to cancel the noise only when the algorithm is about to get some specific string from the sampling.

In order to control what the adversary can do and better analyze the output, we will carefully design a sampling process. First we have the following observation:

Lemma 3.7.3. *Let q_0 and q_1 be the distributions defined above. Then there exist two joint random variables X, Y in $\{0,1\}^t$ with the following properties:*

1. *The distribution of X is q_0.*
2. *The distribution of Y is q_1.*
3. $P(X \preceq Y) = 1$.

Here $\preceq$ is the partial order in $\{0,1\}^t$, $x \preceq y$ if and only if $x_i \leq y_i$ for all $i \in [t]$.

Proof. We claim that we can sample (X, Y) that satisfies the properties we need by the following process:

1. Sample X according to q_0
2. If $|X| \geq t/2$, set $Y = X$.
3. If $|X| < t/2$,
   (a) with probability $q_1(X)/q_0(X)$, set $Y = X$.
   (b) with probability $1 - q_1(X)/q_0(X)$, sample Y from set $\{Y \in \{0,1\}^t : |Y| = t - |X|, X \preceq Y\}$ uniformly at random.

Property 1 and 3 are obvious from the definition of this algorithm. Then it is enough to prove property 2.

Recall the definition of p_0 and p_1. We know that $p_0(x) = \varepsilon^{|x|}(1-\varepsilon)^{t-|x|}$ and $p_1(x) = \varepsilon^{t-|x|}(1-\varepsilon)^{|x|}$. This implies they are symmetric, the value only depend on the hamming weight, we denote $p_b(|x|) = p_b(x)$, $b \in \{0,1\}$. Thus the following properties are straightforward: $p_1(l) = p_0(t-l)$; p_0 is monotone decreasing in term of the hamming weight.

From the definition of q_0 and q_1 it is easy to see they also follow the same properties, i.e. $q_1(l) = q_0(t-l)$; q_0 is monotone decreasing in term of the hamming weight.

The above properties imply $q_1(X) \leq q_0(X)$ when $|X| \leq t/2$, so the algorithm we just described is well defined. Now we prove the distribution of Y is q_1.

For any $y \in \{0,1\}^t$, $|y| < t/2$, $\mathbb{P}(Y = y) = \mathbb{P}(X = y) \cdot q_1(y)/q_0(y) = q_1(y)$. For any $y \in \{0,1\}^t$, $|y| \geq t/2$, let $l = |y|$. Consider the size of the set $\{x \in \{0,1\}^t : |x| = t-l,\ x \preceq y\}$, it is easy to see this number only depends on the hamming weight of y. We use $C(t-l,l)$ to refer this number. Then we have: $\mathbb{P}(Y = y) = \mathbb{P}(X = y) + \sum_{x \preceq y, |x|=t-|y|} \mathbb{P}(X = x) \cdot (1 - q_1(x)/q_0(x)) \cdot (1/C(n-l,l)) = q_0(l) + q_0(t-l) \cdot (1 - q_1(t-l)/q_0(t-l)) = q_1(l) = q_1(y)$. This finishes the proof.

□

Then in protocol $\mathcal{P}_3$, for each P_j, let $N(\varepsilon^{k_i})$ be the noisy bit P_j gets for x_i. Then it sample (x,y) according to the joint distribution in Theorem 3.7.3. If $x_i \bigoplus N(\varepsilon^{k_i}) = 0$, it outputs x, if $x_i \bigoplus N(\varepsilon^{k_i}) = 1$, it outputs y. From the argument above we know the output distribution of this string is an ε-noisy copy of x_i with length t. We will choose the coupling such that this string is the noisy copy used in protocol $\mathcal{P}_2$.

Now consider what the adversary in $\mathcal{P}_3$ can do. Here we will only show the case when $x_i = 0$. (The discussion for $x_i = 1$ is very similar.) If $N(\varepsilon^{k_i}) = 1$, let (x,y) be the string we get from the sampling. Then the adversary may cancel this noise, which means output x instead of y. Then in protocol $\mathcal{P}_2$, it is easy to the adversary can do the same thing. When the noisy vector is y, the adversary changes it to x, that follows from the fact that $x \preceq y$.

Applying the above result for all $i \in [n]$ we get a protocol $\mathcal{P}_3$ in $NCB(\varepsilon)$. The total number of broadcasts in $\mathcal{P}_3$ is $2k$.

It remains to show if $\mathcal{P}_2$ is non-adaptive, then $\mathcal{P}_3$ is non-adaptive. Recall for each auxiliary processor P_i, $i \in [n]$, all the broadcasts it received from $\{Q_i\}_{i\in[n]}$ in $\mathcal{P}_2$ are simulated in $\mathcal{P}_3$ using the noisy copy of x given to P_i at the beginning of the algorithm. Thus if $\mathcal{P}_2$ is non-adaptive, all broadcasts made by P_i only depend on its own noisy copy. This implies $\mathcal{P}_3$ is non-adaptive.

From $\mathcal{P}_3$ to $\mathcal{P}_4$.

Recall $\mathcal{P}_4 = T$ is in the gnd-tree model. Here T will first simulate all the random choices of all the processors in $\mathcal{P}_3$. Then we can view $\mathcal{P}_4$ as a deterministic protocol. Also since all the broadcasts in $\mathcal{P}_3$ are noise free, it suffices for T to evaluate all the broadcasts $b^1, ..., b^{2k}$. We will simulate all the broadcasts except the last one made by the receiver P_0 in the following way: In the j-th round, given the previous broadcasts $b^1, ..., b^{j-1}$, we know b^j should be broadcast by processor P_{i^j}. Once $b^1, ..., b^{j-1}$ are fixed, it can be computed by a boolean function that only depends on the noisy copy of x given to P_{i^j} at the beginning of the protocol. So T will just query this boolean function on this copy and get the right b^j. For the last broadcast, T makes n queries, one for each coordinate of the noisy copy given to P_0. Then T can use this vector together with all previous broadcasts to output the final broadcast.

It is easy to see any adversary in T can be used in $\mathcal{P}_3$ that gives the same output.

The depth of T will be $2k + n$.

Now we show if $\mathcal{P}_3$ is non-adaptive, then T is non-adaptive. We know that in $\mathcal{P}_3$, every broadcast except the last one only depends on the processor's own noisy copy. Thus all these gnd-queries in T are non-adaptive. Also we know the last broadcast made by P_0 is simulated by n non-adaptive queries. This implies T is non-adaptive. (This is why we simulate the last broadcast differently.)

Combine all those reductions we complete the proof of Theorem 3.7.1.

3.7.5 Proof of Theorem 3.7.2

Let $(\varepsilon_1, ...\varepsilon_n)$ be the noise parameters used by T to compute f, then we have $(\prod_i \varepsilon_i)^{1/n} \geq \varepsilon$. Define $K = \{i \in [n] : \varepsilon_i \leq \varepsilon^C\}$, we have: $|K| \leq n/C$. For any partial assignment ρ to K, by using this partial assignment in T we get a gnd-tree $T|_\rho$ with noise parameters $(\varepsilon_i)_{i \in [n] \setminus K}$ such that $T|_\rho$ can compute $f|_\rho$ under any adversary. By the definition of K we know the error rate on every coordinates of $T|_\rho$ is greater than ε^C.

Let T_ρ be the same tree as $T|_\rho$ but run with uniform noise parameter ε^C. Then it is enough to show T_ρ can simulate $T|_\rho$ under noise cancellation adversary. For any adversary $\mathcal{A}$ for T_ρ, we can define adversary $\mathcal{A}'$ for $T|_\rho$ such that $\mathcal{A}'$ will first decrease the noise rate to ε^C for every coordinate independently and then apply the same behaviour as $\mathcal{A}$. Then it is easy to see T_ρ and $T|_\rho$ are equivalent under these adversaries, which finishes the proof.

3.7.6 Proof of Corollary 3.1.2

Recall $\mathcal{P}$ is a protocol in 2-Phase broadcast model. Apply Theorem 3.7.1 we get a non-adaptive flexible noise gnd-tree T with depth $2k+n$ and flexible noise parameter $\varepsilon^{k/n}$. Then apply Theorem 3.7.2 with $C = 4$ and the ρ being the all zero vector, we conclude that there exist a subset $K \subset [n]$ of size at most $n/4$ such that there exist a gnd-tree T' of depth $2k+n$ and fixed noise parameter $\varepsilon^{4 \cdot k/n}$ such that, T' can compute OR on $[n] - K$ under any adversary. Also it is easy to see from the prove that if T is non-adaptive, so does T'. Then apply Theorem 3.1.1 we get:

$$2k + n = \Omega((\varepsilon^{4 \cdot k/n})^4 \cdot 3/4 \cdot n \log(3/4 \cdot n))$$

Solve this equality we get $k = \Omega(\log(1/\varepsilon)^{-1} \cdot n \cdot \log\log n)$.

3.8 Towards Lower Bounds for General gnd-trees

We have proved an $\Omega(n \log n)$ lower bound of function OR for non-adaptive gnd-trees and an $\Omega(n \log\log n)$ lower bound for 2-Phase noisy broadcast protocols. It remains open whether the framework can be extended from to prove lower bounds for unrestricted gnd-trees, thus lower bounds for noisy broadcast protcols.

First we notice that in the general gnd-tree model, OR can be solved using an adaptive gnd-tree of depth $O(\log(1/\varepsilon)^{-1} \cdot n)$ [19]. It is easy to see that the algorithm also works under adversary. The majority function MAJ is a natural candidate of hard function for general gnd-trees under adversary. i.e. we believe any gnd-tree that computes MAJ under any adversary must have depth at least $\Omega(\mathsf{poly}(\varepsilon) \cdot n \log n)$.

Without loss of generality we assume n is odd and $n = 2k+1$. We also define $\tilde{\mu}_0$, $\tilde{\mu}_1$ to be the uniform distributions over all points in $\{0,1\}^n$ with hamming weight k and $k+1$. In analogy with the proof we have in the non-adaptive case, we propose the following conjecture:

Conjecture 3.8.1. *For any gnd-tree T, there exist an nd-tree T' with depth at most $O(\mathsf{poly}(1/\varepsilon) \cdot D(T))$ such that: T is confusable conditioned on T' for $\tilde{\mu}_0$ and $\tilde{\mu}_1$.*

If this conjecture is true, then it directly gives a $\Omega(\mathsf{poly}(\varepsilon) \cdot n \log n)$ lower bound for the gnd-tree model under adversary, which by Theorem 3.7.1 and Theorem 3.7.2 would imply a $\Omega(\log(1/\varepsilon)^{-1} \cdot n \log\log n)$ lower bound for the noisy broadcast model under adversary.

There are two main reasons why the proof framework for non-adaptive gnd-tree does not work directly here.

The first barrier is that since the tree can be adaptive, we can no longer reorganize the tree into a tree of composite queries. In other words, the gnd-tree may make some queries on the first noisy copy, then make some queries on other noisy copies to decide which query it want to ask again on the first noisy copy.

Assume we only consider gnd-trees that have the following property: the tree first makes queries to the first noisy copy, then it will never query the first noisy copy again Instead, depends on the answers it gets so far, it makes queries to the second noisy copy. And then it follows the same rule in the rest of the noisy copies. Under this assumption, we can reform such a gnd-tree into a tree made by composite queries. Then we may able to construct the nd-tree we want using the same framework: simulate each composite query by a small depth nd-tree separately and then compose them together.

However, since we are working on the majority function with distributions $\tilde{\mu}_0$ and $\tilde{\mu}_1$, the definition of strongly confusable does not apply here directly. Recall in the proof we have for OR, a gnd-tree T is strongly confusable if for any $i \in [n]$, it is confusable for μ_0 and σ_i, where σ_i is the distribution that concentrated on e_i. This definition guarantees that the composition of two strongly confusable queries is strongly confusable. Thus, one naive generalization of strongly confusability for a query q in this case would be: for all $x_0 \in \text{supp}(\tilde{\mu}_0)$, $x_1 \in \text{supp}(\tilde{\mu}_1)$, q is confusable for x_0 and x_1. This definition also guarantees that the composition of two strongly confusable queries is strongly confusable. However, on the other hand, it is impossible to make a single gnd query confusable using a small depth nd-tree. The distance between x_0 and x_1 can be n. That means the nd-tree needs to query all the coordinates in the worst case.

That means we need to make a weaker definition of strongly confusable here. Recall the reason why we need strongly confusable is to guarantees that any two distributions that have same support as $\tilde{\mu}_0$ and $\tilde{\mu}_1$ are confusable. We may replace this requirement by a weaker one: Let $\mathcal{D} \subset \{(\sigma_0, \sigma_1) : \text{supp}(\sigma_0) \subset \text{supp}(\tilde{\mu}_0), \text{supp}(\sigma_1) \subset \text{supp}(\tilde{\mu}_1)\}$ be a family of pairs of distributions such that $(\tilde{\mu}_0, \tilde{\mu}_1) \in \mathcal{D}$. We say a gnd-tree T is strongly confusable for $\mathcal{D}$ if for any $(\sigma_0, \sigma_1) \in \mathcal{D}$:

- T is confusable for σ_0 and σ_1.
- For any leaf v in T, $(\sigma_0|(T(\sigma_0) = v), \sigma_1|(T(\sigma_1) = v)) \in \mathcal{D}$.

The second condition guarantees that the composition of two strongly confusable trees is also strongly confusable. Furthermore, we can conclude that if a tree is generated by strongly confusable composite queries, then this tree is strongly confusable. In order to preserve this property we define the conditional confusability by: T is confusable for $\mathcal{D}$ conditioned on T' if for any $(\sigma_0, \sigma_1) \in \mathcal{D}$:

- T is confusable for σ_0 and σ_1 conditioned on T'.
- For any leaf v in T, $(\sigma_0|(T(\sigma_0) = v), \sigma_1|(T(\sigma_1) = v)) \in \mathcal{D}$.

Then the proof of the conjecture might be done by constructing a family $\mathcal{D}$ such that: For any composite query $q : \{0,1\}^n \to \{0,1\}^d$, there exist an nd-tree T' of depth $O(\mathsf{poly}(1/\varepsilon) \cdot d)$ such that q is strongly confusable for $\mathcal{D}$ conditioned on T'. Then apply this theorem to every composite query in T we can get the nd-tree we want.

Note that the proof we have for non-adaptive gnd-tree is also in this framework with $\mathcal{D} = \mathcal{D}_0 \times \mathcal{D}_1$ where $\mathcal{D}_0$ is the set that only contains μ_0 and $\mathcal{D}_1$ is the set of all distributions that support on $\{e_i : i \in [n]\}$.

Here we propose one natural candidate for $\mathcal{D}$. Consider all edges between points with hamming weight k and $k+1$, i.e. $E = \{(x, y) \in \{0,1\}^n \times \{0,1\}^n : |x| = k, |y| = k+1, |x \bigoplus y| = 1\}$. Since all the pairs in E have distance 1, they are more likely to be able to confuse by adversary. It is natural to ask whether for any composite query q, there exist a small depth nd-tree T' such that q is conditionally confusable for every pair in E. Assume this statement is true, then for two distributions σ_0, σ_1, we claim q is confusable conditioned on T' if there exist a weighted perfect matching between σ_0

and σ_1, which means: there exists $(p_{xy})_{\{(x,y)\in E\}}$, $p_{xy} \geq 0, \forall (x,y)$ such that $\sum_y p_x y = \sigma_0(x), \forall x$ and $\sum_x p_x y = \sigma_1(y), \forall y$. Note that we can use $(p_{xy})_{\{(x,y)\in E\}}$ as coefficients to combine adversaries linearly to achieve confusable. The above property defines a set $\mathcal{D} = \{(\sigma_0, \sigma_1) :$ there exists a weighted perfect matching between σ_0 and $\sigma_1\}$. Then it is sufficient to prove: for any composite query $q : \{0,1\}^n \to \{0,1\}^d$, there exist an nd-tree T' of depth $O(\mathsf{poly}(1/\varepsilon) \cdot d)$ such that: q is strongly confusable for $\mathcal{D}$ conditioned on T'.

- For any pair $(x,y) \in E$, q is confusable for x and y conditioned on T'.
- For any output $w \in \{0,1\}^d$, $(\sigma_0|(q(\sigma_0) = w), \sigma_1|(q(\sigma_1) = w)) \in \mathcal{D}$.

We leave the question of whether these are true to a future investigation.

References

[1] Achlioptas, Dimitris, and Frank McSherry. ”On spectral learning of mixtures of distributions.” International Conference on Computational Learning Theory. Springer, Berlin, Heidelberg, 2005.

[2] Agrawal, Rakesh, and Ramakrishnan Srikant. ”Privacy-preserving data mining.” ACM Sigmod Record. Vol. 29. No. 2. ACM, 2000.

[3] Aharonov, Dorit, and Michael Ben-Or. ”Fault-tolerant quantum computation with constant error.” Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. ACM, 1997.

[4] Batman, Lucia, et al. ”Finding heavy hitters from lossy or noisy data.” Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Springer, Berlin, Heidelberg, 2013. 347-362.

[5] Beckner, William. ”Inequalities in Fourier analysis.” Annals of Mathematics (1975): 159-182.

[6] Bonami, Aline. ”tude des coefficients de Fourier des fonctions de Lp (G).” Ann. Inst. Fourier (Grenoble) 20.2 (1970): 335-402.

[7] Borell, Christer. ”Positivity improving operators and hypercontractivity.” Mathematische Zeitschrift 180.3 (1982): 225-234.

[8] Chan, Siu-On, et al. ”Learning mixtures of structured distributions over discrete domains.” Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2013.

[9] Cover, Thomas M., and Joy A. Thomas. ”Elements of information theory 2nd edition.” (2006).

[10] De, Anindya, Michael Saks, and Sijian Tang. ”Noisy population recovery in polynomial time.” In Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on, pp. 675-684. IEEE, 2016.

[11] De, Anindya, Michael Saks, and Sijian Tang. ”Noisy population recovery in polynomial time (if the noise is not too high)”, 2015.

[12] De, Anindya, Ryan O’Donnell, and Rocco Servedio. ”Sharp bounds for population recovery.” arXiv preprint arXiv:1703.01474 (2017).

[13] Dutta, Chinmoy, et al. ”A tight lower bound for parity in noisy communication networks.” Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2008.

[14] Dvir, Zeev, et al. "Restriction access." Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ACM, 2012.

[15] El Gamal, Abbas. "Open problems presented at the 1984 workshop on specific problems in communication and computation sponsored by bell communication research." Open Problems in Communication and Computation (1987).

[16] Evans, William S., and Leonard J. Schulman. "Signal propagation and noisy circuits." IEEE Transactions on Information Theory 45.7 (1999): 2367-2373.

[17] Evans, William, and Nicholas Pippenger. "Average-case lower bounds for noisy Boolean decision trees." SIAM Journal on Computing 28.2 (1998): 433-446.

[18] Feige, Uriel. "On the complexity of finite random functions." Information processing letters 44.6 (1992): 295-296.

[19] Feige, Uriel, et al. "Computing with unreliable information." Proceedings of the twenty-second annual ACM symposium on Theory of computing. ACM, 1990.

[20] Feige, Uriel, and Joe Kilian. "Finding OR in a noisy broadcast network." Information Processing Letters 73.1-2 (2000): 69-75.

[21] Feldman, Jon, Ryan O'Donnell, and Rocco A. Servedio. "Learning mixtures of product distributions over discrete domains." SIAM Journal on Computing 37.5 (2008): 1536-1564.

[22] Feller, Willliam. An introduction to probability theory and its applications. Vol. 2. John Wiley & Sons, 2008.

[23] Gallager, Robert G. "Finding parity in a simple broadcast network." IEEE Transactions on Information Theory 34.2 (1988): 176-180.

[24] Goyal, Navin, Guy Kindler, and Michael Saks. "Lower bounds for the noisy broadcast problem." SIAM Journal on Computing 37.6 (2008): 1806-1841.

[25] Gross, Leonard. "Logarithmic sobolev inequalities." American Journal of Mathematics 97.4 (1975): 1061-1083.

[26] Kalai, Adam Tauman, Ankur Moitra, and Gregory Valiant. "Efficiently learning mixtures of two Gaussians." Proceedings of the forty-second ACM symposium on Theory of computing. ACM, 2010.

[27] Kearns, Michael, et al. "On the learnability of discrete distributions." Proceedings of the twenty-sixth annual ACM symposium on Theory of computing. ACM, 1994.

[28] Kitaev, A. Yu. "Fault-tolerant quantum computation by anyons." Annals of Physics 303.1 (2003): 2-30.

[29] Kushilevitz, Eyal, and Yishay Mansour. "Computation in Noisy Radio Networks." SODA. Vol. 98. 1998.

[30] Lovett, Shachar, and Jiapeng Zhang. "Noisy population recovery from unknown noise." Conference on Learning Theory. 2017.

[31] Lovett, Shachar, and Jiapeng Zhang. "Improved noisy population recovery, and reverse Bonami-Beckner inequality for sparse functions." Proceedings of the forty-seventh annual ACM symposium on Theory of computing. ACM, 2015.

[32] Moitra, Ankur, and Michael Saks. "A polynomial time algorithm for lossy population recovery." Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on. IEEE, 2013.

[33] Mossel, Elchanan, Krzysztof Oleszkiewicz, and Arnab Sen. "On reverse hypercontractivity." Geometric and Functional Analysis 23.3 (2013): 1062-1097.

[34] Mossel, Elchanan, et al. "Non-interactive correlation distillation, inhomogeneous Markov chains, and the reverse Bonami-Beckner inequality." Israel Journal of Mathematics 154.1 (2006): 299-336.

[35] Mossel, Elchanan, Ryan O'Donnell, and Krzysztof Oleszkiewicz. "Noise stability of functions with low influences: invariance and optimality." Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on. IEEE, 2005.

[36] Newman, Ilan. "Computing in fault tolerance broadcast networks." Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on. IEEE, 2004.

[37] O'Donnell, Ryan. Analysis of boolean functions. Cambridge University Press, 2014.

[38] Pippenger, Nicholas. "On networks of noisy gates." Foundations of Computer Science, 1985., 26th Annual Symposium on. IEEE, 1985.

[39] Reischuk, Rdiger, and Bernd Schmeltz. "Reliable computation with noisy circuits and decision trees-a general n log n lower bound." Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on. IEEE, 1991.

[40] Schulman, Leonard J. "Coding for interactive communication." IEEE Transactions on Information Theory 42.6 (1996): 1745-1756.

[41] Stanley, Richard P. "Enumerative combinatorics, Wadsworth Publ." Co., Belmont, CA (1986).

[42] Wigderson, Avi, and Amir Yehudayoff. "Population recovery and partial identification." Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on. IEEE, 2012.

[43] Wilf, Herbert S. "Hadamard determinants Mbius functions, and the chromatic number of a graph." Bulletin of the American Mathematical Society 74.5 (1968): 960-964.batman2013finding

[44] Yao, A. "On the complexity of communication under noise." Invited talk in the 5th ISTCS conference. 1997.