Electronic Colloquium on Computational Complexity, Report No. 115 (2009)

Local list-decoding and testing of random linear codes from high-error

Swastik Kopparty*and Shubhangi Saraf†

November 13, 2009

Abstract

In this paper, we give surprisingly efficient algorithms for list-decoding and testing *random* linear codes. Our main result is that random sparse linear codes are locally testable and locally list-decodable in the *high-error* regime with only a *constant* number of queries. More precisely, we show that for all constants $c > 0$ and $\gamma > 0$, and for every linear code $\mathcal{C} \subseteq \{0,1\}^N$ which is:

- *sparse*: $|\mathcal{C}| \leq N^c$, and
- *unbiased*: each nonzero codeword in $\mathcal{C}$ has weight $\in (\frac{1}{2} - N^{-\gamma}, \frac{1}{2} + N^{-\gamma})$,

$\mathcal{C}$ is locally testable and locally list-decodable from $(\frac{1}{2} - \epsilon)$-fraction worst-case errors using only $\text{poly}(\frac{1}{\epsilon})$ queries to a received word. We also give *sub-exponential time* algorithms for list-decoding arbitrary unbiased (but not necessarily sparse) linear codes in the high-error regime. In particular, this yields the first sub-exponential time algorithm even for the problem of (unique) decoding random linear codes of inverse-polynomial rate from a fixed positive fraction of errors.

Earlier, Kaufman and Sudan had shown that sparse, unbiased codes can be locally (unique) decoded and locally tested from a constant-fraction of errors, where this constant-fraction tends to 0 as the number of codewords grows. Our results significantly strengthen their results, while also having significantly simpler proofs.

At the heart of our algorithms is a natural "self-correcting" operation defined on codes and received words. This self-correcting operation transforms a code $\mathcal{C}$ with a received word w into a simpler code $\mathcal{C}'$ and a related received word w', such that w is close to $\mathcal{C}$ if and only if w' is close to $\mathcal{C}'$. Starting with a sparse, unbiased code $\mathcal{C}$ and an arbitrary received word w, a constant number of applications of the self-correcting operation reduces us to the case of local list-decoding and testing for the *Hadamard code*, for which the well known algorithms of Goldreich-Levin and Blum-Luby-Rubinfeld are available. This yields the constant-query local algorithms for the original code $\mathcal{C}$.

Our algorithm for decoding unbiased linear codes in sub-exponential time proceeds similarly. Applying the self-correcting operation to an unbiased code $\mathcal{C}$ and an arbitrary received word a super-constant number of times, we get reduced to the problem of *learning noisy parities*, for which non-trivial sub-exponential time algorithms were recently given by Blum-Kalai-Wasserman and Feldman-Gopalan-Khot-Ponnuswami. Our result generalizes a result of Lyubashevsky, which gave sub-exponential time algorithms for decoding random linear codes of inverse-polynomial rate, from *random errors*.

*Computer Science and Artificial Intelligence Laboratory, MIT, swastik@mit.edu. This work was done while the author was a summer intern at Microsoft Research New England.

†Computer Science and Artificial Intelligence Laboratory, MIT, shubhangi@csail.mit.edu. This work was done while the author was a summer intern at Microsoft Research New England.

ISSN 1433-8092

1 Introduction

In this paper, we study the basic tasks of *error-correction* and *error-detection* for random linear codes in the presence of high error-rates. Our main result is that with high probability for a sparse random linear code $\mathcal{C}$, by inspecting just a *constant* number of coordinates of an arbitrary received word w, one can (i) decide whether the distance of w from $\mathcal{C}$ is at most 49% or not, and (ii) recover implicit descriptions of all the codewords of $\mathcal{C}$ which are 49%-close to w. We also give *sub-exponential time* algorithms for decoding random linear codes of inverse-polynomial rate from 49% errors.

We begin by setting up some notation. A linear code $\mathcal{C}$ in $\mathbb{F}_2^N$ is simply a linear subspace of $\mathbb{F}_2^N$. The elements of $\mathcal{C}$ are often referred to as "codewords". We say that $\mathcal{C}$ is sparse if $|\mathcal{C}| \leq N^c$ for some constant c. For a string $x \in \mathbb{F}_2^N$, we define its (normalized Hamming) weight $\mathrm{wt}(x)$ to equal $\frac{1}{n}\times$ (the number of nonzero coordinates of x). We define the *bias* of the code $\mathcal{C}$ as $\max_{y \in \mathcal{C}\setminus\{0\}} \left|\frac{1-\mathrm{wt}(y)}{2}\right|$. Thus each nonzero codeword y of a code of bias β has $\mathrm{wt}(y) \in [(1-\beta)/2, (1+\beta)/2]$. For two strings $x, y \in \mathbb{F}_2^N$, we define the (normalized Hamming) distance between x and y, $\Delta(x,y)$, to be $\frac{1}{n}\times$ (the number of coordinates $i \in [N]$ where x_i and y_i differ). We then define the distance of a string x from the code $\mathcal{C}$, $\Delta(x,\mathcal{C})$ to be the minimum distance of x to some codeword of $\mathcal{C}$:

$$\Delta(x,\mathcal{C}) = \min_{y \in \mathcal{C}} \Delta(x,y).$$

The basic algorithmic tasks associated with codes are error-detection and error-correction. Here we are given an arbitrary received word $w \in \mathbb{F}_2^N$, and we want to (1) determine if $\Delta(w,\mathcal{C}) > \delta$, and (2) find all codewords $y \in \mathcal{C}$ such that $\Delta(w,\mathcal{C}) \leq \delta$. In recent years, there has been much interest in developing sublinear time algorithms (and in particular, highly query-efficient algorithms) for these tasks. In what follows, we will describe our results on various aspects of these questions.

1.1 Local list-decoding

Informally, a local list-decoder for $\mathcal{C}$ from δ-fraction errors is a randomized algorithm A that, when given oracle access to a received word $w \in \mathbb{F}_2^N$, recovers the list of all codewords c such that $\Delta(c,w) < \delta$, while querying w in very few coordinates. The codewords thus recovered are "implicitly represented" by randomized algorithms $A_1, \ldots, A_l$ with oracle access to w. Given a coordinate $j \in [N]$, A_i makes very few queries to w and is supposed to output the j^{th} coordinate of the codeword that it implicitly represents.

A particular case of local list-decoding which is of significant interest is local list-decoding in the "high-error" regime. Specifically, for every constant $\epsilon > 0$, one wants to query-efficiently locally list-decode a code from $(\frac{1}{2}-\epsilon)$-fraction errors (the significance of $\frac{1}{2}-\epsilon$ is that it is just barely enough to to distinguish the received word from a random string in $\mathbb{F}_2^N$; for codes over large alphabets, one considers the problem of decoding from $(1-\epsilon)$-fraction errors). Local list-decoding in the high-error regime plays a particularly important role in the complexity-theoretic applications of coding theory (see [STV99], for example).

The first known local list-decoder (for any code) came from the seminal work of Goldreich and Levin [GL89], which gave time-efficient, low-query, local list-decoders for the Hadamard code in

the high-error regime. In the following years, many highly non-trivial local list-decoders were developed for various codes, including multivariate polynomial based codes (in the works of Goldreich-Rubinfeld-Sudan [GRS00], Arora-Sudan [AS03], Sudan-Trevisan-Vadhan [STV99], and Gopalan-Klivans-Zuckerman [GKZ08]) and combinatorial codes such as direct-product codes and XOR codes (in the works of Impagliazzo-Wigderson and Impagliazzo-Jaiswal-Kabanets-Wigderson [IW97, IJKW08]). Many of these local list-decoders, especially the ones in the high-error regime, play a prominent role in celebrated results in complexity theory on hardness amplification and pseudorandomness [IW97, STV99, IJKW08].

To summarize, all known local list-decoding algorithms were for highly structured algebraic or combinatorial codes. Recently, Kaufman and Sudan [KS07] showed that *random* sparse linear codes can be locally (unique-)decoded from a small constant fraction of errors. This was the first result to show that query-efficient decoding algorithms could also be associated with random, unstructured codes. This result was proved by studying the weight distribution of these codes and their duals, and in particular was based on the MacWilliams identities and non-trivial information about the roots of Krawtchouk polynomials. In an earlier paper [KS09], we gave an alternate (and arguably simpler) proof of this result, as part of a more general attempt to characterize sparse codes that are locally decodable and testable in the low-error regime. Popular belief [Sud09] suggested that these low-error local decoders for random codes could *not* be extended to the high-error regime.

Our first main result is that even random codes *can* have query-efficient local list-decoders in the high-error regime. Specifically, we show that linear codes which are *sparse* and *unbiased* (both properties are possessed by sparse random linear codes with high probability) admit high-error local list-decoders with constant query complexity.

Informal Theorem A For every constant $c, \gamma > 0$, every linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$ with N^c codewords and bias $N^{-\gamma}$ can be locally list-decoded from $(1/2 - \epsilon)$-fraction errors using only $\mathrm{poly}(\frac{1}{\epsilon})$ queries to a received word.

The formal statement appears in Theorem 4.

1.2 Local testing

Informally, a local tester for $\mathcal{C}$ is a randomized algorithm A, which when given oracle access to a received word $w \in \mathbb{F}_2^N$, makes very few queries to the received word w, and distinguishes between the case where $w \in \mathcal{C}$ and the case where w is "far" from all codewords of $\mathcal{C}$.

The first local tester (for any code) came from the seminal work of Blum, Luby and Rubinfeld [BLR93], which gave an efficient, 3-query local tester for the Hadamard code. This result was subsequently generalized to the problem of local testing of low-degree multivariate polynomials [RS96, AS98, ALM+98, AKK+03]. This body of work played a significant role in the proof of the PCP theorem and were the harbingers of the field of property testing.

A particular variant of local testability which is of significant interest is local testing in the "high-error" regime. Here, for every constant $\epsilon > 0$, one wants to query-efficiently distinguish between the cases $\Delta(w, \mathcal{C}) < 1/2 - \epsilon$, i.e., w is "close" to $\mathcal{C}$, and $\Delta(w, \mathcal{C}) \approx 1/2$, i.e., w is as far from $\mathcal{C}$ as a random point is (for codes over large alphabets, $1/2$ gets replaced by 1). For the Hadamard code, the

existence of such testers follows from the Fourier-analytic proof of the BLR linearity test [BCH+96]. For the code of degree 2 multivariate polynomials over $\mathbb{F}_2$, local testers in the high-error regime were given by Samorodnitsky [Sam07]. For the code of multivariate polynomials over large fields, local-testers in the high-error regime were given by Raz-Safra [RS97], Arora-Sudan [AS03] and Moshkovitz-Raz [MR06]. More recently, Dinur-Goldenberg [DG08] and Impagliazzo-Kabanets-Wigderson [IKW09] gave query-efficient local testing algorithms in the high-error regime for the combinatorial families: the direct-product and XOR codes. These algebraic and combinatorial high-error local-testers led to some remarkable constructions of PCPs with high soundness.

As in the case of list-decodability, we have the situation that all known locally-testable codes in the high-error regime are for highly structured algebraic or combinatorial codes. Again, Kaufman and Sudan [KS07] showed that a random sparse linear code is locally testable in the low-error regime by studying its weight distribution and the weight distribution of its dual. We [KS09] had an alternate proof of this result too. Popular belief [Sud09] again suggested that local-testability in the *high-error* regime could not be found in random linear codes.

Our second main result is that random codes *can* have query-efficient local testers in the high-error regime. Specifically, sparse and unbiased codes admit high-error local testers with constant query complexity.

Informal Theorem B For every constant $c, \gamma > 0$, every linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$ with N^c codewords and bias $N^{-\gamma}$ can be locally tested from $(1/2 - \epsilon)$-fraction errors using only $\mathrm{poly}(\frac{1}{\epsilon})$ queries to a received word.

The formal theorem appears as Theorem 4, where we in fact show something stronger (called *distance estimation in the high-error regime*): for such codes, using constantly many queries, one can distinguish between $\Delta(w, \mathcal{C}) > 1/2 - \epsilon_1$ and $\Delta(w, \mathcal{C}) < 1/2 - \epsilon_2$ for every constants $0 < \epsilon_1 < \epsilon_2 < 1/2$.

1.3 Subexponential time list-decoding

The techniques we develop to address the previous questions turn out to be useful for making progress on another fundamental algorithmic question in coding theory: that of *time-efficient* worst-case decoding of random linear codes. Given a random linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$ and an arbitrary received word $w \in \mathbb{F}_2^N$, we are interested in quickly finding all the codewords $c \in \mathcal{C}$ such that $\Delta(w, c) < \frac{1}{2} - \epsilon$, for constant $\epsilon > 0$. We show that this problem can be solved in *sub-exponential time*, using just the unbiasedness of $\mathcal{C}$. Our algorithm uses some recent breakthroughs on the problem of learning noisy-parities due to Blum-Kalai-Wasserman [BKW03] and Feldman-Gopalan-Khot-Ponnuswami [FGKP06].

Informal Theorem C For all constants $\alpha, \gamma > 0$, for every linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$ with dimension n, where $N = n^{1+\alpha}$, and bias $N^{-\gamma}$, and for every constant $\epsilon > 0$, $\mathcal{C}$ can be list-decoded from $(\frac{1}{2} - \epsilon)$-fraction errors in time $2^{O(n/\log\log n)}$.

The formal statement appears in Theorem 6.

In particular, the above theorem implies that if $\mathcal{C} \subseteq \mathbb{F}_2^N$ is a *random linear code* with dimension $n = N^{\frac{1}{1+\alpha}}$, then for every constant $\epsilon > 0$, $\mathcal{C}$ can be list-decoded from $(\frac{1}{2} - \epsilon)$-fraction errors in time $2^{O(n/\log\log n)}$.

Earlier, it was not even known how to *unique-decode* random linear codes from 0.1-*fraction* worst-case errors in time $2^{o(n)}$. For decoding random linear codes of inverse-polynomial rate from *random errors*, Lyubashevsky [Lyu05] gave a sub-exponential time algorithm, also based on algorithms for the Noisy Parity problem. Our result generalizes his in two ways: we decode from worst-case errors, and we give a natural, explicit criterion (namely low-bias) on the code $\mathcal{C}$ which guarantees the success of the algorithm.

A related result (and one that we use in our proof) is the sub-exponential time worst-case decoding of random linear codes in $\mathbb{F}_2^N$, of dimension $n = O(\log N \cdot \log\log N)$, in a weaker model [FGKP06, Theorem 10]. In this model, the adversary first corrupts the received bit associated to $(1/2 - \epsilon)$-fraction of the 2^n possible linear encoding functions, after which the code is randomly chosen. Our result has a more natural coding theory interpretation: the random code is chosen first, and then the adversary choses an arbitrary received word at distance $(1/2-\epsilon)$ from the code. In the language of learning theory, the [FGKP06] result concerns learning parities in the presence of *agnostic noise*, while our result deals with the model of learning parities in the presence of *nasty classification noise* [BEK02].

1.4 Time-efficient local algorithms for dual-BCH codes

For the family of *dual-BCH* codes, perhaps the most important family of sparse, unbiased codes, we show that the constant-query local list-decoding and local testing algorithms can be made to run in a *time-efficient* manner too. The dual-BCH codes form a natural family of polynomial-based codes generalizing the Hadamard code. They have a number of extremal properties which give them an important role in coding theory. For example, the dual-BCH code $\mathcal{C} \subseteq \mathbb{F}_2^N$ with N^t codewords has bias as small as $O(t \cdot N^{-1/2})$, which is *optimal* for codes with N^t codewords!

The key to making our earlier query-efficient local list-decoding and local testing algorithms run in a time-efficient manner for dual-BCH codes, is a time-efficient efficient algorithm for a certain sampling problem that arises in the local list-decoder and tester. This sampling problem turns out to be closely related to an algorithmic problem that was considered in the context of low-error testing of dual-BCH codes [KL05], that of sampling constant-weight BCH codewords. A variant of the sampling algorithm of [KL05] turns out to suffice for our problem too, and this leads to the following result.

Informal Theorem D For every constant c, the dual-BCH code $\mathcal{C} \subseteq \mathbb{F}_2^N$ with N^c codewords, can be locally list-decoded and locally tested from $(1/2 - \epsilon)$-fraction errors in time $\mathrm{poly}(\log N, \frac{1}{\epsilon})$ using only $\mathrm{poly}(\frac{1}{\epsilon})$ queries to a received word.

The formal statement appears as Theorem 5. The original algorithm for sampling constant-weight BCH codewords given in [KL05], and was based [Lit09] on results on the weight distribution of BCH codes [KL95]. We give an alternate (and possibly simpler) analysis in Section 6.

Organization of this paper: In Section 2, we give a brief overview of the techniques we use. In Section 3 we formally define local list-decoders and testers and state our main results. The proofs appear in Sections 4, 5 and 6.

2 Methods

In this section, we give an overview of the main ideas underlying our algorithms. Our goal in this section is to stress the simplicity and naturalness of our techniques.

The main component of our algorithms is a certain "self-correcting" operation which transforms a code $\mathcal{C}$ with a received word w into a simpler code $\mathcal{C}'$ and a related received word w', such that w is close to $\mathcal{C}$ if and only if w' is close to $\mathcal{C}'$. Repeated application of this self-correcting operation will allow us to reduce our list-decoding and testing problems for $\mathcal{C}$ to certain kinds of list-decoding and testing problems for a significantly simpler code $\mathcal{C}^*$ (in our case, $\mathcal{C}^*$ will be the Hadamard code). Query-efficient/time-efficient algorithms for the simpler code $\mathcal{C}^*$ then lead to query-efficient/time-efficient algorithms for the original code $\mathcal{C}$.

In order to simplify the description of the self-correcting operation, we first translate our problems into the language of list-decoding and testing under *distributions*. Let $\mathcal{C} \subseteq \{0,1\}^N$ be a linear code of dimension n, and let G be an $n \times N$ generator matrix for $\mathcal{C}$. Let $S = \{v_1, v_2, \ldots, v_N\} \subset \mathbb{F}_2^n$ denote the set of columns of G. We associate to $\mathcal{C}$ the distribution μ over $\mathbb{F}_2^n$ which is uniform over S. Note that if the code $\mathcal{C}$ has low bias, then the resulting distribution μ has small Fourier bias. Every word w in $\mathbb{F}_2^N$ can be viewed as a function $f_w : S \to \mathbb{F}_2$, where $f_w(v_i) = w_i$. Under this mapping, every *codeword* of $\mathcal{C}$ gets associated with a *linear* function.

Note that via this translation, the problem of testing if w is close to some codeword exactly translates into the problem of testing if f_w is close to some linear function *under the distribution* μ (where the distance of two functions g, h under μ is measured by the probability that g and h differ on a random sample from μ). Similarly, the problem of local list-decoding, i.e. the problem of finding all codewords close to w, translates into the problem of finding all linear functions that are close to f_w under the distribution μ.

We now come to the self-correcting operation on f and μ. The operation has the property that it maintains the property "f correlates with a linear function under μ", and at the same time it results in a distribution that is "simpler" in a certain precise sense.

Define $\mu^{(2)}$ to be the convolution of μ with itself; i.e., it is the distribution of the sum of two independent samples from μ. We define $f^{(2)} : \mathbb{F}_2^n \to \mathbb{F}_2$ to be the (probabilistic) function, where for a given x, $f^{(2)}(x)$ is sampled as follows: first sample y_1 and y_2 independently and uniformly from μ conditioned on $y_1 + y_2 = x$, and return $f(y_1) + f(y_2)$ (if there are no such y_1, y_2, then define $f^{(2)}(x)$ arbitrarily).

The following two simple facts are key to what follows:

- $\mu^{(2)}$ is "simpler" than μ: the statistical distance of $\mu^{(2)}$ to the uniform distribution on $\mathbb{F}_2^n$ is significantly smaller than the statistical distance of μ to the uniform distribution on $\mathbb{F}_2^n$ (this follows from the low Fourier bias of μ, which in turn came from the unbiasedness of $\mathcal{C}$).
- If f is $(\frac{1}{2} - \epsilon)$-close to a linear function g under μ, then $f^{(2)}$ is $(\frac{1}{2} - 2\epsilon^2)$-close to g under $\mu^{(2)}$: this is a formal consequence of our definition of $f^{(2)}$. In particular, if f is noticeably-close to g under μ, then so is $f^{(2)}$ under $\mu^{(2)}$.

This leads to a general approach for list-decoding/testing for linear functions under μ. First pick k large, and consider the distribution $\mu^{(k)}$ and the function $f^{(k)}$ (defined analogously to $\mu^{(2)}$ and

$f^{(2)}$). If k is chosen large enough, then $\mu^{(k)}$ will in fact be 2^{-10n}-close to the uniform distribution in statistical distance. Furthermore, if k is not too large, then $f^{(k)}$ will be noticeably-close under $\mu^{(k)}$ to the same linear functions that f is close to under μ. Thus, if k is suitable (as a function of the initial bias/sparsity of the code) $f^{(k)}$ is noticeably-close *under the uniform distribution* to the same linear functions that f is close to under μ. Now all we need to do is run a local list-decoding/testing algorithm on $f^{(k)}$ under the uniform distribution.

An important issue that was swept under the rug in this discussion, is the query/time-efficiency of working with $f^{(k)}$ and $\mu^{(k)}$. If we ignore running-time, one can simulate oracle access to $f^{(k)}$ using just a factor k larger number of queries to f. This leads to our query-efficient (but time-inefficient) algorithms for sparse, unbiased linear codes in the high-error regime (in this setting k only needs to be a constant). We stress that our proof of this result is significantly simpler and stronger than earlier analyses of local algorithms (in the low-error regime) of sparse, unbiased codes [KL05, KS07].

The bottleneck for implementing these local, query-efficient algorithms in a time-efficient manner is the following algorithmic "back-sampling" problem: given a point $x \in \mathbb{F}_2^n$, produce a sample from the distribution of $y_1, \ldots, y_k$ picked independently from μ conditioned on $\sum y_i = x$. A time-efficient back-sampling algorithm would allow us to time-efficiently simulate oracle access to $f^{(k)}$ given oracle access to f. For random sparse linear codes, solving this problem in time sublinear in N is impossible; however for specific, interesting sparse unbiased codes, this remains an important problem to address. For the special case of dual-BCH codes, perhaps the most important family of sparse, unbiased codes, we observe that the back-sampling problem can be solved using a small variant of an algorithm of Kaufman-Litsyn [KL05]. Thus for dual-BCH codes, we get $\mathrm{poly}\log(N)$-time, constant-query local testing and local list-decoding algorithms in the high-error regime.

For sub-exponential time list-decoding, we follow the same plan. Here too we will self-correct f to obtain a function $f^{(k)}$, such that every linear function that correlates with f under the μ distribution, also correlates with $f^{(k)}$ under the uniform distribution over $\mathbb{F}_2^n$. However, since we are now paying attention to running time (and we do not know how to solve the back-sampling problem for μ efficiently in general), we cannot afford to allow the list-decoder over the uniform distribution over $\mathbb{F}_2^n$ to query the value of $f^{(k)}$ at any point that it desires (since this will force us to back-sample in order to compute $f^{(k)}$ at that point). Instead, we will use some recent remarkable list-decoders ([FGKP06, BKW03]), developed in the context of learning noisy parities, which can find all linear functions close (under the uniform distribution) to an arbitrary function h in sub-exponential time by simply querying the function h at independent uniformly random points of $\mathbb{F}_2^n$! Using the unbiasedness of μ, it turns out to be easy to evaluate $f^{(k)}$ at independent uniformly random points of $\mathbb{F}_2^n$. This leads to our sub-exponential time list-decoding algorithm.

Relationship to the k-wise XOR on codes: Back in the language of codes, what happened here has a curious interpretation. Given a code $\mathcal{C} \subseteq \mathbb{F}_2^N$, the k-wise XOR of $\mathcal{C}$, $\mathcal{C}^{(\oplus k)}$, is the code contained in $\mathbb{F}_2^{N^k}$ defined as follows: for every codeword $c \in \mathcal{C}$, there is a codeword $c^{(\oplus k)} \in F_2^{[N]^k}$ whose value in coordinate $(i_1, \ldots, i_k)$ equals $c_{i_1} \oplus c_{i_2} \oplus \ldots \oplus c_{i_k}$. In terms of this operation, our algorithms simply do the following: given a code $\mathcal{C}$ and received word w, consider the code $\mathcal{C}^{(\oplus k)}$ with received word $w^{(\oplus k)}$. The crucial observation is, that for k chosen suitably as a function of the bias/sparsity of $\mathcal{C}$, the code $\mathcal{C}^{(\oplus k)}$ is essentially, up to repeating each coordinate a roughly-equal number of times, the Hadamard code! Additionally, $w^{(\oplus k)}$ is close to $c^{(\oplus k)}$ for a codeword c if and only if w is close c. Thus decoding/testing $w^{(\oplus k)}$ for the Hadamard code now suffices to complete the algorithm.

The k-wise XOR on codes is an operation that shows up often as a device for hardness amplification, to convert functions that are hard to compute into functions that are even harder to compute. Our algorithms use the XOR operation for "good": here the XOR operation is a vehicle to *transfer* query-efficient/time-efficient algorithms for the Hadamard code to query-efficient/time-efficient algorithms for arbitrary unbiased codes.

3 Definitions and Main Results

3.1 Codes

We begin by formalizing the notions of local list-decoding and local-testing of codes.

Definition 1 *Let $r \in \mathbb{F}_2^N$. A randomized algorithm A is said to* implicitly compute r with error δ *if for all $j \in [N]$ it holds that*

$$\Pr\left[A(j) = r_j\right] \geq 1 - \delta,$$

where the probability is over the internal randomness of A.

Below we formally define local list-decoders. In what follows, we denote by A^w the randomized algorithm A with oracle access to w.

Definition 2 (Local list-decoder) *Let $\mathcal{C} \subseteq \mathbb{F}_2^N$ be a code. A (δ, q, l)-local list-decoder for $\mathcal{C}$ is a randomized algorithm A, such that for every $w \in \mathbb{F}_2^N$, A outputs a list of randomized algorithms $\{A_1, \ldots, A_l\}$, such that:*

- *For each $i \in [l]$, and for each input $j \in [N]$, the algorithm $A_i^w(j)$ makes at most q queries into w during its execution.*
- *With probability at least $2/3$, for every codeword $c \in \mathcal{C}$ with $\Delta(w, c) < \delta$, there exists an index i such that the algorithm A_i^w implicitly computes c with error $1/3$ (in the sense of Definition 1).*

We now define local-testers.

Definition 3 (Local tester) *Let $\mathcal{C} \subseteq \mathbb{F}_2^N$ be a code. Let $0 < \delta_1 < \delta_2$. A (δ_1, δ_2, q)-local tester for $\mathcal{C}$ is a randomized algorithm A, such that for every $w \in \mathbb{F}_2^N$,*

- *A^w makes at most q oracle calls to w during its execution.*
- *If there is a codeword $c \in \mathcal{C}$ such that $\Delta(w, c) < \delta_1$, A^w accepts with probability at least $2/3$.*
- *If for all $c \in \mathcal{C}$, $\Delta(w, c) > \delta_2$, then A^w rejects with probability at least $2/3$.*

We can now formally state our main theorems regarding the local testability and local list-decodability of sparse unbiased codes.

Theorem 4 (Constant-query local list-decoding and testing of sparse, unbiased, linear codes) *Let $C \subseteq \mathbb{F}_2^N$ be a linear code with N^c codewords and with bias $\beta = N^{-\gamma}$. Then,*

- **Local list-decoding:** *for all $0 < \epsilon < 1/2$, there exists a $\left(1/2 - \epsilon, \left(\frac{1}{\epsilon}\right)^{O(c/\gamma)}, \left(\frac{1}{\epsilon}\right)^{O(c/\gamma)}\right)$-local list-decoder for C.*
- **Local testing:** *for all constants ϵ_1, ϵ_2 such that $0 < \epsilon_1 < \epsilon_2 < 1/2$, there exists a $\left(1/2 - \epsilon_2, 1/2 - \epsilon_1, \left(\frac{1}{\epsilon_2 - \epsilon_1}\right)^{O(c/\gamma)}\right)$-local tester for C.*

The next theorem states that the above algorithms can be implemented in a time-efficient manner for the dual-BCH codes. The proof appears in Section 6.

Theorem 5 (Time-efficient local algorithms for dual-BCH codes) *Let t be a constant, and let N be an integer of the form $2^s - 1$. Let $\mathcal{C}$ be the dual-BCH code of length N and dimension st. Then,*

- **Local list-decoding:** *for all $0 < \epsilon < 1/2$, there exists a $\left(1/2 - \epsilon, \text{poly}\left(\frac{1}{\epsilon}\right), \text{poly}\left(\frac{1}{\epsilon}\right)\right)$-local list-decoder for C which runs in time $\text{poly}(\log N, \frac{1}{\epsilon})$.*
- **Local testing:** *for all constants ϵ_1, ϵ_2 such that $0 < \epsilon_1 < \epsilon_2 < 1/2$, there exists a $\left(1/2 - \epsilon_2, 1/2 - \epsilon_1, \text{poly}\left(\frac{1}{\epsilon_2 - \epsilon_1}\right)\right)$-local tester for C which runs in time $\text{poly}(\log N, \frac{1}{\epsilon})$.*

Finally, we formally state our theorem on the sub-exponential time list-decoding of unbiased linear codes from worst-case errors. The proof appears in Section 5.

Theorem 6 (Subexponential time list-decoding of unbiased codes) *For all constants $\alpha, \gamma, \epsilon > 0$, there is an algorithm* SubExpListDecode*, which when given:*

- *the generator matrix G of a n-dimensional binary linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$, where $\mathcal{C}$ has bias $N^{-\gamma}$ and $N = n^{1+\alpha}$, and*
- *a vector $r \in \mathbb{F}_2^N$,*

runs in time $\exp(\frac{n}{\log \log n})$ and with high probability, finds all codewords $c \in \mathcal{C}$ such that $\Delta(c, r) < \frac{1}{2} - \epsilon$.

3.2 Distributions

As suggested earlier, it is greatly simplifying to work in the language of distributions over $\mathbb{F}_2^n$. We now formalize some notions in this context.

For sets $\mathcal{D}$ and $\mathcal{R}$, we will use the concept of a *probabilistic function* from $\mathcal{D}$ to $\mathcal{R}$. Formally, a probabilistic function h from $\mathcal{D}$ to $\mathcal{R}$ is given by a collection of distributions $(\nu_d)_{d \in \mathcal{D}}$, where each ν_d is a distribution over $\mathcal{R}$. Functionally, we view probabilistic functions as a black box, such that

whenever an element $d \in \mathcal{D}$ is fed into this box, a sample from ν_d is returned by the box. The samples returned from different invocations of this box are mutually independent (in particular, two different calls to the box on the same input may return different values). Abusing notation, we write $h : \mathcal{D} \to \mathcal{R}$. Note that ordinary (deterministic) functions can be viewed as a special case of probabilistic functions.

For two probabilistic functions $h_1, h_2 : \mathbb{F}_2^n \to \mathbb{F}_2$ and a distribution μ over $\mathbb{F}_2^n$, we define the μ-distance between h_1 and h_2 by:

$$\Delta_\mu(h_1, h_2) = \Pr_{x \in \mu}[h_1(x) \neq h_2(x)],$$

where the probability is over x chosen according to μ, and $h_1(x)$ and $h_2(x)$ being sampled independently from the probabilistic functions h_1 and h_2.

We now introduce some terminology regarding linear functions and characters. Every linear function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ is of the form $g(x) = a \cdot x \stackrel{\text{def}}{=} \sum_{i=1}^n a_i x_i$ for some $a \in \mathbb{F}_2^n$. A *character* of $\mathbb{F}_2^n$ is a function $\chi : \mathbb{F}_2^n \to \mathbb{R}$ such that for some linear function $g : \mathbb{F}_2^n \to \mathbb{F}_2$, we have $\chi(x) = (-1)^{g(x)}$ for all x. The character χ_0 with $\chi_0(x) = 1$ for all x is called the *trivial character*.

We can now define the analogous versions of local list-decoding and local testing for distributions.

Definition 7 (Local list-decoder for linear functions under μ) *Let μ be a distribution over $\mathbb{F}_2^n$. A (δ, q, l)-local list-decoder for linear functions under μ is a randomized algorithm A, such that for every probabilistic function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, A outputs a list of randomized algorithms $\{A_1, \ldots, A_l\}$, such that:*

- *For each $i \in [l]$, and for each input $x \in \mathbb{F}_2^n$, the algorithm $A_i^f(x)$ makes at most q queries into f during its execution.*
- *With probability at least $2/3$, for every linear function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $\Delta_\mu(f, g) < \delta$, there exists an index i such that the algorithm A_i^f implicitly computes g with error $1/3$ (in the sense of Definition 1).*

Definition 8 (Local tester for linear functions under μ) *Let μ be a distribution over $\mathbb{F}_2^n$. Let $0 < \delta_1 < \delta_2$. A (δ_1, δ_2, q)-local tester for linear functions, or linearity tester, under μ is a randomized algorithm A, such that for every probabilistic function $f : \mathbb{F}_2^n \to \mathbb{F}_2$,*

- *A makes at most q oracle calls to f.*
- *If there is a linear function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $\Delta_\mu(g, f) < \delta_1$, A^f accepts with probability at least $2/3$*
- *If for all linear functions $g : \mathbb{F}_2^n \to \mathbb{F}_2$, $\Delta_\mu(g, f) > \delta_2$, then A^w rejects with probability at least $2/3$*

We now state our main theorem on the local list-decoding and testing of linear functions under unbiased distributions. The proof appears in Section 4. In the next subsection, we will derive Theorem 4 from it.

Theorem 9 (Local list-decoding and testing of linear functions under unbiased distributions) *Let μ be a distribution over $\mathbb{F}_2^n$ such that for all nontrivial characters $\chi : \mathbb{F}_2^n \to \{-1, +1\}$,*

$$|\mathop{\mathbf{E}}_{x \in \mu}[\chi(x)]| < 2^{-\eta n}.$$

Then,

1. **Local list-decoder:** *for every $0 < \epsilon < 1/2$, there exists a $\left((1/2 - \epsilon), \left(\frac{1}{\epsilon}\right)^{O(1/\eta)}, \left(\frac{1}{\epsilon}\right)^{O(1/\eta)}\right)$-local list-decoder for linear functions under μ.*
2. **Local tester:** *for every $0 < \epsilon_1 < \epsilon_2 < 1/2$, there exists a $\left(1/2 - \epsilon_2, 1/2 - \epsilon_1, \left(\frac{1}{\epsilon_2 - \epsilon_1}\right)^{O(1/\eta)}\right)$-local tester for linear functions under μ.*

3.3 From Distributions to Codes

In this subsection, we will use Theorem 9 on local algorithms for unbiased distributions to prove Theorem 4 on local algorithms for sparse, unbiased codes.

Definition 10 (Distribution associated to a code) *Let $\mathcal{C} \subseteq \{0,1\}^N$ be a linear code of dimension n. Let G be any generator matrix for $\mathcal{C}$ (i.e., G is an $n \times N$ matrix whose rows are linearly independent and span $\mathcal{C}$). Then the* distribution associated to $\mathcal{C}$ via G *is distribution μ over $\mathbb{F}_2^n$ which is uniform over the columns of G.*

The connection between local algorithms for codes and local algorithms for distributions comes from the following simple theorem.

Theorem 11 *Let $\mathcal{C}$ be a code and let μ be a distribution associated to $\mathcal{C}$ via G. Then,*

- **Local list-decoding:** *If there exists a (δ, q, l)-local list-decoder for linear function under μ, then there exists a (δ, q, l)-local list-decoder for $\mathcal{C}$.*
- **Local testing:** *If there exists a (δ_1, δ_2, q)-local tester for linear functions under μ, then there exists a (δ_1, δ_2, q)-local tester for $\mathcal{C}$.*

Proof Let $v_1, \ldots, v_N \in \mathbb{F}_2^n$ be the columns of G. Let $w \in \mathbb{F}_2^N$ be a received word.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be the probabilistic function such that on input x, $f(x)$ is sampled as follows: If there exists $i \in [N]$ such that $x = v_i$, then pick an i uniformly at random from $\{i \in N \mid x = v_i\}$, and output w_i. If there is no $i \in [N]$ such that $x = v_i$, then output a uniformly random element of $\mathbb{F}_2$ (or define $f(x)$ arbitrarily).

Let $c = a \cdot G$ be a codeword of $\mathcal{C}$, and let $g : \mathbb{F}_2^n \to \mathbb{F}_2$ be the corresponding linear function given by $g(x) = a \cdot x$. Observe that our definition of f implies that

$$\Delta(c, w) = \Delta_\mu(g, f). \tag{1}$$

Also note that one can simulate oracle access to f given oracle access to w.

The local list-decoding and testing algorithms for $\mathcal{C}$ can now be easily described. Given a received word $w \in \mathbb{F}_2^N$, consider the corresponding probabilistic function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, and run the algorithms for local list-decoding/testing for linear functions under μ on f. The local tester for $\mathcal{C}$ accepts if and only if the corresponding local tester for linear functions under μ accepts. The local list-decoding algorithm for $\mathcal{C}$ outputs, for each linear function $g(x) = a \cdot x$ in the list output by the list-decoding algorithm for linear functions under μ, the codeword $c = a \cdot G$. Correctness follows immediately from Equation (1). ■

We now give a proof of the main theorem on local list-decodability and testability of sparse, unbiased codes, Theorem 4, assuming Theorem 9.

Proof of Theorem 4: Let $\mathcal{C}$ be a linear code with $|\mathcal{C}| \leq N^c$ and bias $\leq N^{-\gamma}$. Let G be a generator matrix for $\mathcal{C}$. Let $n = \log |\mathcal{C}|$. Let μ be the distribution associated to $\mathcal{C}$ via G.

We first notice that our assumptions on the sparsity and the bias of the code $\mathcal{C}$ imply that for all nontrivial characters $\chi : \mathbb{F}_2^n \to \{-1, +1\}$,

$$|\mathop{\mathbf{E}}_{x \in \mu}[\chi(x)]| < 2^{-\eta n}, \tag{2}$$

where $\eta = \gamma/c$. Indeed, if $\chi(x) = (-1)^{a \cdot x}$, then the codeword $a \cdot G$ of the code $\mathcal{C}$ has weight $\frac{1+\mathbf{E}_{x \in \mu}[\chi(x)]}{2}$. By assumption on the bias of $\mathcal{C}$, the weight of any nonzero codeword of $\mathcal{C}$ lies in the interval $(\frac{1-N^{-\gamma}}{2}, \frac{1+N^{-\gamma}}{2})$. By the sparsity of $\mathcal{C}$, this interval is contained in the interval $(\frac{1-2^{-\gamma n/c}}{2}, \frac{1+2^{-\gamma n/c}}{2})$, and Equation (2) follows.

Theorem 9 now implies that the existence of a suitable list-decoding algorithm and testing algorithm for linear functions under μ. By Theorem 11, this implies the existence of the desired local list-decoding and local testing algorithms for $\mathcal{C}$. ■

4 Local list-decoding and testing under distributions

In this section, we prove Theorem 9.

4.1 Some notation

We now introduce some notation.

Let U_n denote the uniform distribution over $\mathbb{F}_2^n$.

Let μ be a distribution over $\mathbb{F}_2^n$. For an integer $k > 0$, we let $\mu^{(k)}$ denote the distribution of $x_1 + \ldots + x_k$, where the x_i are picked independently from μ. The crucial fact that we need about this operation is given by the following lemma.

Lemma 12 *Suppose that for every nontrivial character χ of $\mathbb{F}_2^n$, we have*

$$|\mathbb{E}_{x \in \mu}[\chi(x)]| < \beta.$$

Then the distribution $\mu^{(k)}$ is $\beta^k \cdot 2^n$-close to U_n in statistical distance.

For an integer $k > 0$ and $y \in \mathbb{F}_2^n$, we let $\mu^{(k,y)}$ denote the distribution of $(x_1, \ldots, x_k) \in (\mathbb{F}_2^n)^k$, where the x_i are picked independently from μ *conditioned* on $\sum_{i=1}^k x_i = y$.

For a probabilistic function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, we define the *real function associated with* f to be the function $F : \mathbb{F}_2^n \to [-1, 1]$, where for each $x \in \mathbb{F}_2^n$, $F(x) = 2p_x - 1$, where p_x is the probability that $f(x)$ equals 0. Thus, if f was a deterministic function, then $F(x) = (-1)^{f(x)}$.

For functions $F, G : \mathbb{F}_2^n \to \mathbb{R}$ and a distribution μ over $\mathbb{F}_2^n$, we define the μ-correlation between F and G, denoted $\langle F, G \rangle_\mu$ be the quantity

$$\mathbb{E}_{x \in \mu} F(x)G(x).$$

Note that if f and g are probabilistic functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, and F and G are the real functions associated with them, then

$$\langle F, G \rangle_\mu = 1 - 2\Delta_\mu(f, g). \tag{3}$$

4.2 Local list-decoding and testing for distributions

In this subsection, we will prove Theorem 9 on the local list-decoding and testing of linear functions under unbiased distributions.

Recall the setup: we have a distribution μ over $\mathbb{F}_2^n$ and a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$. Our algorithms will work by reducing to the uniform distribution over $\mathbb{F}_2^n$. Specifically, we will produce a (probabilistic) function $h : \mathbb{F}_2^n \to \mathbb{F}_2$ (from which we can sample, given oracle access to f) such that the μ-distance of f from a linear function g "corresponds" to the U_n-distance of h from g. Then list-decoding and linearity-testing of h under the uniform distribution will enable us to list-decode and test the linearity of f under the μ-distribution.

We begin by introducing the local list-decoder for linear functions under the uniform distribution, given by the famous Goldreich-Levin algorithm[1].

Theorem 13 (Local list-decoder for the uniform distribution [GL89]) *For every $0 < \epsilon < 1/2$, there exists a $\left(\frac{1}{2} - \epsilon, \mathrm{poly}(\frac{1}{\epsilon}), \mathrm{poly}(\frac{1}{\epsilon})\right)$-local list-decoder for linear functions under U_n.*

This version of the Goldreich-Levin theorem, in conjunction with the BLR linearity test [BLR93][2], easily implies a constant query local tester for linear functions under the uniform distribution in the *high error* regime.

Theorem 14 (Local tester for the uniform distribution) *For every $0 < \epsilon_1 < \epsilon_2 < 1/2$, there exists a $\left(\frac{1}{2} - \epsilon_2, \frac{1}{2} - \epsilon_1, \mathrm{poly}\left(\frac{1}{\epsilon_2 - \epsilon_1}\right)\right)$-local tester for linear functions under U_n.*

[1] To be precise, we state a generalization of the usual Goldreich-Levin theorem to handle probabilistic functions; inspecting the original proof of the theorem immediately reveals that it applies verbatim to probabilistic functions too.

[2] suitably generalized to handle probabilistic functions.

In Appendix A, we outline a proof of Theorem 14 using Theorem 13 and the BLR linearity test.

We now give a proof of the main result of this section, Theorem 9.

Theorem 9 (Local list-decoding and testing of linear functions under unbiased distributions) *Let μ be a distribution over $\mathbb{F}_2^n$ such that for all nontrivial characters $\chi : \mathbb{F}_2^n \to \{-1, +1\}$,*

$$|\mathop{\mathbf{E}}_{x \in \mu}[\chi(x)]| < 2^{-\eta n}.$$

Then,

1. **Local list-decoder:** *For every ϵ such that $0 < \epsilon < 1/2$, there exists a $\left(\frac{1}{2} - \epsilon, (\frac{1}{\epsilon})^{O(1/\eta)}, \left(\frac{1}{\epsilon}\right)^{O(1/\eta)}\right)$-local list-decoder for linear functions under μ.*
2. **Local tester:** *For every ϵ_1, ϵ_2 such that $0 < \epsilon_1 < \epsilon_2 < 1/2$, there exists a $\left(\frac{1}{2} - \epsilon_2, \frac{1}{2} - \epsilon_1, \left(\frac{1}{\epsilon_2 - \epsilon_1}\right)^{O(1/\eta)}\right)$-local tester for linear functions under μ.*

Proof Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be the probabilistic function which we want to locally list-decode or locally test. Let k be an odd integer $\approx \frac{100}{\eta}$, and note that Lemma 12 implies that the distribution $\mu^{(k)}$ is 2^{-50n} close to U_n in statistical distance.

Let $h : \mathbb{F}_2^n \to \mathbb{F}_2$ be the probabilistic function defined as follows. For $y \in \mathbb{F}_2^n$, to sample $h(y)$, first sample $(x_1, \ldots, x_k)$ from $\mu^{(k,y)}$, and then output $\sum_{i=1}^k f(x_i)$. Clearly, any algorithm that is given oracle access to f can simulate oracle access to h with a $O(\frac{1}{\eta})$-blowup in the number of queries (although the *time complexity* of this simulation need not be bounded). See also the Remark following the proof.

We will assume without loss of generality in the rest of the proof that $(\frac{1}{\epsilon})^k$ and $(\frac{1}{\epsilon_2 - \epsilon_1})^k$ are all $2^{o(n)}$, since otherwise the trivial local testers and list-decoders that query the function at every point of the domain will satisfy the bounds in the theorem.

The crux of our algorithms is the following claim (the proof appears in the analysis of the algorithms).

Claim 15 *For every linear function $g : \mathbb{F}_2^n \to \mathbb{F}_2$, letting $\Delta_\mu(f, g) = \frac{1-\beta}{2}$ and $\Delta_{U_n}(h, g) = \frac{1-\alpha}{2}$, we have*

$$\alpha = \beta^k \pm 2^{-50n}.$$

We now present our local list-decoding and testing algorithms for linear functions. Both these algorithms will assume they have oracle access to the probabilistic function h, which by the earlier discussion, can be simulated by using oracle access to f.

Algorithm for Local List-Decoding: Fix any ϵ such that $0 < \epsilon < 1/2$. We now describe a $\left(\frac{1}{2} - \epsilon, (\frac{1}{\epsilon})^{O(1/\eta)}, \left(\frac{1}{\epsilon}\right)^{O(1/\eta)}\right)$-local list-decoder, D_μ, for linear functions under μ. Let $\alpha = \frac{(2\epsilon)^k - 2^{-50n}}{2}$.

By Theorem 13, there is a $\left(\frac{1}{2} - \alpha, \mathrm{poly}(\frac{1}{\alpha}), \mathrm{poly}(\frac{1}{\alpha})\right)$-local list-decoder, D_{U_n}, for linear functions under U_n. The list-decoder for linear functions under μ, D_μ^f, works as follows given oracle access to the probabilistic function f:

- Run D_{U_n} and get as output a list of randomized algorithms $\{D_1, \ldots, D_l\}$
- Return a list of randomized algorithms $\{D_{\mu,1}, \ldots, D_{\mu,l}\}$, where $D_{\mu,i}$ works as follows: The algorithm $D^f_{\mu,i}(x)$ simulates $D^h_i(x)$ (recall that given oracle access to f, one can simulate oracle access to h) and returns the value returned by $D^h_i(x)$.

Algorithm for Testing: Fix any ϵ_1, ϵ_2 such that $0 < \epsilon_1 < \epsilon_2 < 1/2$. We now describe a $\left(\frac{1}{2} - \epsilon_2, \frac{1}{2} - \epsilon_1, \left(\frac{1}{\epsilon_2 - \epsilon_1}\right)^{O(1/\eta)}\right)$-local tester, T_μ, for linear functions under μ.

Let $\alpha_1 = \frac{(2\epsilon_1)^k + 2^{-50n}}{2}$ and let $\alpha_2 = \frac{(2\epsilon_2)^k - 2^{-50n}}{2}$. By Theorem 14, there is a $\left(\frac{1}{2} - \alpha_2, \frac{1}{2} - \alpha_1, \text{poly}\left(\frac{1}{\alpha_2 - \alpha_1}\right)\right)$-local tester, T_{U_n}, for linear functions under U_n.

The linearity-tester under μ, T_μ, works as follows given oracle access to the probabilistic function f:

- Simulate $T^h_{U_n}$ (again recall that given oracle access to f, one can simulate oracle access to h), and accept if and only if $T^h_{U_n}$ accepts.

Analysis: The correctness of the algorithms will follow easily from Claim 15. We first prove this claim.

Let $F : \mathbb{F}_2^n \to [-1, +1]$ be the real function associated with f. Let $H : \mathbb{F}_2^n \to [-1, +1]$ be the real function associated with h.

Let χ_g be the real function associate with g. Note that since g is a deterministic function, $\chi_g(x)$ simply equals $(-1)^{g(x)}$.

Note that by definition,

$$H(x) = \mathbb{E}_{(x_1,\ldots,x_k)\in\mu^{(k,y)}}\left[\prod_{i=1}^{k} F(x_i)\right].$$

We want to estimate $\Delta_{U_n}(h, g)$ in terms of $\Delta_\mu(f, g)$. Equivalently, by Equation (3), we want to bound $\langle H, \chi_g\rangle_{U_n}$ in terms of $\langle F, \chi_g\rangle_\mu$. We begin by expanding out the definition of $\langle H, \chi_g\rangle_{U_n}$:

$$\begin{aligned}
\langle H, \chi_g\rangle_{U_n} &= \mathbb{E}_{y\in U_n}[H(y)\chi_g(y)] \\
&= \mathbb{E}_{y\in\mu^{(k)}}[H(y)\chi_g(y)] \pm 2^{-50n} \qquad \text{by Lemma 12 and choice of } k \\
&= \mathbb{E}_{y\in\mu^{(k)}}\left[\left(\mathbb{E}_{(x_1,\ldots,x_k)\in\mu^{(k,y)}}\left[\prod_{i=1}^{k} F(x_i)\right]\right)\chi_g(y)\right] \pm 2^{-50n} \\
&= \mathbb{E}_{y\in\mu^{(k)}}\left[\mathbb{E}_{(x_1,\ldots,x_k)\in\mu^{(k,y)}}\left[\prod_{i=1}^{k} (F(x_i)\chi_g(x_i))\right]\right] \pm 2^{-50n} \\
&= \mathbb{E}_{x_1,\ldots,x_k\in\mu}\left[\prod_{i=1}^{k} (F(x_i)\chi_g(x_i))\right] \pm 2^{-50n} \\
&= (\mathbb{E}_{x\in\mu}[F(x)\chi_g(x)])^k \pm 2^{-50n}. \\
&= \langle F, \chi_g\rangle^k_\mu \pm 2^{-50n}.
\end{aligned}$$

By Equation (3), we get that $\alpha = \beta^k \pm 2^{-50n}$. This proves the claim.

Analysis of decoder: Claim 15 together with Equation (3) implies that if g is a linear function such that $\Delta_\mu(g, f) < 1/2 - \epsilon$, then $\Delta_{U_n}(g, h) < 1/2 - \alpha$, where $\alpha = \frac{(2\epsilon)^k - 2^{-50n}}{2}$. Now, the decoder D^f_μ simulates $D^h_{U_n}$, and hence returns an algorithm computing every linear function g such that $\Delta_{U_n}(g, h) < 1/2 - \alpha$. Hence, it also returns an algorithm computing every linear function g such that $\Delta_\mu(g, f) < 1/2 - \epsilon$.

Observe that the query complexity of D^f_μ is given by the query complexity of $D^h_{U_n}$, but blown up by a factor of $k = O(1/\eta)$, since each each query to h is made by making k queries to f. Plugging in the parameters of D_{U_n}, we obtain the desired query complexity for D^f_μ

Analysis of tester: Just as we stated in the analysis of the decoder, Claim 15 together with Equation (3) implies that

- If g is a linear function such that $\Delta_\mu(g, f) < 1/2 - \epsilon_2$, then $\Delta_{U_n}(g, h) < 1/2 - \alpha_2$, where $\alpha_2 = \frac{(2\epsilon_2)^k - 2^{-50n}}{2}$.
- If g is a linear function such that $\Delta_\mu(g, f) > 1/2 - \epsilon_1$, then $\Delta_{U_n}(g, h) > 1/2 - \alpha_1$, where $\alpha_1 = \frac{(2\epsilon_1)^k + 2^{-50n}}{2}$. (For this implication we use that k is odd.)

Now, the tester T^f_μ simulates $T^h_{U_n}$, and outputs the output of $T^h_{U_n}$. If f is $1/2 - \epsilon_2$ close to a linear function under μ, then by the above implications, h is $1/2 - \alpha_2$ close to a linear function under U_n, and hence $T^h_{U_n}$ accepts. If f is $1/2 - \epsilon_1$ far from all linear function under μ, then by the above implications, h is $1/2 - \alpha_1$ far from all linear function under U_n, and hence $T^h_{U_n}$ rejects.

Just as in the case of the local list-decoder, observe that the query complexity of T^f_μ is given by the query complexity of $T^h_{U_n}$, but blown up by a factor of $k = O(1/\eta)$, since each each query to h is made by making k queries to f. Plugging in the parameters of T_{U_n}, we obtain the desired query complexity for T^f_μ ■

Remark (Time Complexity) Suppose there is an algorithm $\mu\mathsf{Sample}$ that runs in time T, which when given $y \in \mathbb{F}_2^n$, produces a vector $x \in (\mathbb{F}_2^n)^k$ whose distribution is λ-close to to $\mu^{(k,y)}$. Then in time poly(T), one can sample (upto λ-statistical distance) the value of $h(y)$ for any given y, using oracle access to f. This will allow the algorithms of the previous section to run in time which is a poly(T) factor larger than the corresponding running times of the invoked algorithms for the uniform distribution. In particular, if $\lambda < 2^{-\Omega(n)}$, the list-decoder will run in time poly $\left(T, \left(\frac{1}{\epsilon}\right)^{1/\eta}\right)$, and the linearity-tester will run in time poly $\left(T, \left(\frac{1}{\epsilon_2 - \epsilon_1}\right)^{1/\eta}\right)$.

In the Section 6, we show that show how to implement the $\mu\mathsf{Sample}$ algorithm in time poly(n) when μ is the distribution associated with the dual-BCH code via a particular generator matrix.

5 Sub-exponential time list-decoding of unbiased linear codes

In this section, we prove Theorem 6 on the sub-exponential time list-decoding of unbiased linear codes.

As in the local algorithms given in Section 4, we first change our language to that of distributions on $\mathbb{F}_2^n$. Given a linear code $\mathcal{C}$ and a received word r from which we want to list-decode, we get a distribution μ and a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, and we want to find all linear functions $g : \mathbb{F}_2^n \to \mathbb{F}_2$ that are close to f in μ-distance.

Following the local algorithms, here too we will "self-correct" f to obtain a function h, such that every linear function that correlates with f under the μ distribution, also correlates with h under the uniform distribution over $\mathbb{F}_2^n$. However, since we are now paying attention to running time (and we do not know how to solve the μSample problem efficiently in general), we cannot afford to allow the list-decoder over the uniform distribution over $\mathbb{F}_2^n$ to query the value of h at any point that it desires. Instead, we will use some recent remarkable list-decoders, developed in the context of learning noisy parities, which can list-decode in sub-exponential time by simply querying the function h at independent uniformly random points of $\mathbb{F}_2^n$! Using the unbiasedness of μ, it turns out to be easy to evaluate h at independent uniformly random points of $\mathbb{F}_2^n$. This completes the informal description of the algorithm.

Below we state the result on agnostically learning parities ([FGKP06]) that gives a list-decoding algorithm for any function h over the uniform distribution over $\mathbb{F}_2^n$ in sub-exponential time, in the model where the algorithm receives independent and uniform samples from h. The parameters in Theorem 16 below are obtained by combining the parameters for the running time and query complexity of the Blum-Kalai-Wasserman [BKW03] algorithm for learning *noisy parities* with random errors in sub-exponential time with the Feldman-Gopalan-Khot-Ponnuswami [FGKP06] algorithm for reducing the problem of learning noisy parities with agnostic errors to the problem of learning noisy parities with random errors.

Theorem 16 ([FGKP06]) *There is an algorithm* AgnosticParity*, such that for every probabilistic function $h : \mathbb{F}_2^n \to \mathbb{F}_2$, every $\beta > 0$ and every choice of parameter $t > 0$,:*

- *If* AgnosticParity *is given as input the parameter β and $s \geq \exp(\log(1/\beta)n^{1/t} + \frac{tn}{\log n})$ independent samples $(x, h(x))$, where x is chosen uniformly $\mathbb{F}_2^n$, then* AgnosticParity *returns the set of all linear functions $g : \mathbb{F}_2^n \to \mathbb{F}_2$ such that $\Delta_{U_n}(h, g) < \frac{1-\beta}{2}$.*
- AgnosticParity *runs in time* $\exp(\log(1/\beta)n^{1/t} + \frac{tn}{\log n})$

We are now ready to prove our main theorem, Theorem 6, on the sub-exponential time list-decoding of unbiased codes.

Theorem 6 (Subexponential time list-decoding of unbiased codes, restated) *For all constants $\alpha, \gamma, \epsilon > 0$, there is an algorithm* SubExpListDecode*, which when given:*

- *the generator matrix G of a n-dimensional binary linear code $\mathcal{C} \subseteq \mathbb{F}_2^N$, where $\mathcal{C}$ has bias $N^{-\gamma}$ and $N = n^{1+\alpha}$, and*

- *a vector* $r \in \mathbb{F}_2^N$,

runs in time $\exp(\frac{n}{\log\log n})$ *and with high probability, finds all codewords* $c \in \mathcal{C}$ *such that* $\Delta(c,r) < \frac{1-\epsilon}{2}$.

Proof

Let v_i denote the i^{th} column of G.

Below we present the list-decoding algorithm. For concreteness, we will present the algorithm in the language of codes. The proof of correctness will employ the language of distributions over $\mathbb{F}_2^n$.

- Initialize the set $Z = \emptyset$.
- Set $k = \frac{100n}{\gamma \log N}$, $t = 2\frac{\log n}{\log\log n}$, and $s = \exp(\log(1/\epsilon)kn^{1/t} + \frac{tn}{\log n})$.
- Repeat s times:
  - Sample $i_1, \ldots, i_k$ uniformly at random from $[N]$.
  - Include $(\sum_{j=1}^k v_{i_j}, \sum_{j=1}^k r_{i_j})$ in Z.
- Choose the parameter $t = 2\frac{\log n}{\log\log n}$, and feed $\epsilon^k/2$ and Z as input to AgnosticParity, and let its output be $\{g_1, \ldots, g_l\}$.
- For each $i \in [l]$, write the linear function g_i in the form $g_i(x) = a_i \cdot x$, and output the codeword $a_i \cdot G$.

Clearly, (by Theorem 16), this algorithm runs in time $\exp(\log(1/\epsilon)kn^{1/t} + \frac{tn}{\log n})$. By choice of k and t, and since ϵ is constant and $N = n^{1+\alpha}$ for constant α, this equals $\exp(\frac{n}{\log\log n})$.

We now show that this algorithm is indeed a list-decoder as claimed. Let $c = a \cdot G$ be a codeword of $\mathcal{C}$ such that $\Delta(c,r) < \frac{1-\epsilon}{2}$. Let $g : \mathbb{F}_2^n \to \mathbb{F}_2$ be the linear function given by $g(x) = a \cdot x$.

Let μ be the uniform distribution over the columns of G (i.e., μ is the distribution associated to $\mathcal{C}$ via G). Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a probabilistic function, where to sample $f(x)$, we pick an $i \in [N]$ uniformly at random from the set of all i such that $x = v_i$, and output r_i (if there are no such i, then we can define $f(x)$ arbitrarily).

Let $h : \mathbb{F}_2^n \to \mathbb{F}_2$ be the probabilistic function defined as follows. For $y \in \mathbb{F}_2^n$, to sample $h(y)$, first sample $(x_1, \ldots, x_k)$ from $\mu^{(k,y)}$ (recall that $\mu^{(k,y)}$ denotes the distribution of $(x_1, \ldots, x_k) \in (\mathbb{F}_2^n)^k$, where the x_i are picked independently from μ *conditioned* on $\sum_{i=1}^k x_i = y$), and then output $\sum_{i=1}^k f(x_i)$.

As in the local setting, we have the following claim. We omit the proof (which is identical to the proof Claim 15).

Claim 17 *Let* $\Delta_{U_n}(h,g) = \frac{1-\alpha'}{2}$. *Then,*

$$\alpha' > \epsilon^k \pm 2^{-50n}.$$

Now, if $\epsilon^k \leq 2^{-n}$, then the trivial brute force algorithm that searches over all possible linear function also satisfies the parameters of the theorem. Hence, from now on we assume without loss of generality that $\epsilon^k > 2^{-n}$. In particular, the claim above implies that $\alpha' > \epsilon^k/2$.

Let us analyze a single iteration of the loop in the algorithm. We will show that the distribution of the point $(\sum_j v_{i_j}, \sum_j r_{i_j})$ (which gets included in the set Z) is 2^{-50n}-close to the distribution of $(y, h(y))$, where y is sampled uniformly at random from $\mathbb{F}_2^n$, and $h(y)$ is sampled independently from the probabilistic function h.

Assuming this for the moment, it is easy to see that the algorithm will include c in its output. Indeed, the set of samples Z fed into the algorithm AgnosticParity is 2^{-50n}-close to the distribution of

$$((x_1, h(x_1)), (x_2, h(x_2)), \ldots, (x_s, h(x_s))),$$

where the x_i are independent. Thus by Theorem 16, since $\Delta_{U_n}(h, g) < \frac{1-\epsilon^k/2}{2}$, we see that the algorithm AgnosticParity will return g in its output. Thus c will be included in the output of the algorithm SubExpListDecode, as desired.

We now analyze the distribution of $(\sum_j v_{i_j}, \sum_j r_{i_j})$. By definition, the distribution of this is identical to the distribution of $(\sum_{j=1}^k x_j, \sum_{j=1}^k f(x_j))$, where $x_1, \ldots, x_k$ are sampled independently and uniformly from μ. By Lemma 12, the distribution of $\sum_{j=1}^k x_j$ is 2^{-50n}-close to uniformly distributed over $\mathbb{F}_2^n$. For any given $y \in \mathbb{F}_2^n$, conditioned on $\sum_{j=1}^k x_j = y$, the distribution of $\sum_{j=1}^k f(x_j)$ has a distribution identical to that of $h(y)$. Thus the distribution of $(\sum_j v_{i_j}, \sum_j r_{i_j})$ is 2^{-50n}-close to the distribution of $(y, h(y))$, for uniformly sampled y.

This completes the proof of correctness of the algorithm. ■

Since the bias of random linear codes is small, we immediately get the following corollary.

Corollary 18 *For all constants $\alpha, \epsilon > 0$, there is an algorithm which when given the $n \times N$ generator matrix G of a random binary linear code $\mathcal{C}$, where $N = n^{1+\alpha}$, and when given a vector $r \in \mathbb{F}_2^N$, runs in time $\exp(\frac{n}{\log \log n})$ and finds all codewords $c \in \mathcal{C}$ such that $\Delta(c, r) < \frac{1}{2} - \epsilon$.*

6 Efficient local algorithms for dual-BCH codes

In this section, we will prove Theorem 5 on the existence of time-efficient and query-efficient algorithms for list-decoding and testing dual-BCH codes. As mentioned earlier, we will do this by showing that the μ-sampling problem associated to the dual-BCH codes can be solved efficiently. The algorithm presented here is a small variant of one given by Kaufman-Litsyn [KL05] for sampling constant weight BCH codewords. The analysis we present is self-contained (and possibly simpler).

Let us recall the definition of the dual-BCH code of length $2^s - 1$ and parameter t, $\mathsf{dBCH}(s, t)$. First identify $\mathbb{F}_{2^s}$ with $\mathbb{F}_2^s$ in a linear fashion. For an element $\alpha \in \mathbb{F}_{2^s}^*$, let $\mathbf{v}_{\alpha,t} \in \mathbb{F}_{2^s}^t$ be the vector given by

$$\mathbf{v}_{\alpha,t} = (\alpha, \alpha^3, \alpha^5, \ldots, \alpha^{2t-1}).$$

Define $\overline{\mathbf{v}}_{\alpha,t} \in (\mathbb{F}_2^s)^t$ to be the vector obtained from $\mathbf{v}_{\alpha,t}$ by interpreting each of its coordinates as elements of $\mathbb{F}_2^s$. Then the dual-BCH code $\mathsf{dBCH}(s,t)$ is the code with the $(st) \times (2^s - 1)$ generator matrix G, where the columns of G are the vectors $\overline{\mathbf{v}}_{\alpha,t}$, for $\alpha \in \mathbb{F}_{2^s}^*$.

We denote by $S_{t,s} \subseteq \mathbb{F}_{2^s}^t$ the set of all vectors $\mathbf{v}_{t,\alpha}$, where $\alpha \in \mathbb{F}_{2^s}^*$. By definition, the distribution μ associated to $\mathsf{dBCH}(s,t)$ via G is the uniform distribution over $S_{t,s}$. In this setting, in the $\mu\mathsf{Sample}$ problem we are given a vector $\mathbf{b} \in \mathbb{F}_{2^s}^t$, and want to sample uniformly from the set of all $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in S_{t,s}^k$ such that $\sum_{i=1}^k \mathbf{x}_i = \mathbf{b}$. Below we give an algorithm to sample uniformly from the set of all $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in S_{t,s}^k$ such that the $\mathbf{x}_i$ are all *distinct*, and $\sum_{i=1}^k \mathbf{x}_i = \mathbf{b}$. Later we will show that this distribution is in fact $2^{-\Omega(s)}$-close to the actual distribution that we want.

Before proceeding with the sampling algorithm, we state two of the fundamental properties of dual-BCH codes that will be useful in the analysis.

Proposition 19 (Unbiasedness of dual-BCH codes) *For every character $\chi : \mathbb{F}_{2^s}^t \to \mathbb{C}$, we have*

$$\left|\mathbb{E}_{x \in S_{t,s}} [\chi(x)]\right| \leq 2t \cdot 2^{-s/2}.$$

This is a consequence of the Weil bound on character sums [LN94].

Proposition 20 (Distance of BCH codes [MS81]) *For every $V \subseteq S_{t,s}$ with $1 \leq |V| \leq 2t$, we have $\sum_{v \in V} v \neq 0$.*

Theorem 21 *There is an randomized algorithm* $\mathsf{BCHSample}$*, which when given as input:*

- *integers t, k, s with $4t < k < s$,*
- *a vector $\mathbf{b} \in \mathbb{F}_{2^s}^t$,*

runs in time $\mathrm{poly}(t^t, s)$*, and outputs a random set $T \subseteq \mathbb{F}_{2^s}$ such that:*

- $|T| = k$.
- *The distribution of T is $2^{-\Omega(s)}$-close to the uniform distribution over the set*

$$\mathcal{F} = \{R \subseteq \mathbb{F}_{2^s} \mid ||R| = k, \sum_{\alpha \in R} \mathbf{v}_{\alpha,t} = \mathbf{b}\}.$$

Proof We give the algorithm $\mathsf{BCHSample}$ below. The main idea is to first pick all but t elements of the set T uniformly at random, and to then consider the residual problem of picking the last t elements such that $\sum_{\alpha \in T} \mathbf{v}_{\alpha,t}$ equals $\mathbf{b}$. We show that with probability at least $\frac{1}{t!}$ (which is noticeable if t is a constant), there do exist t elements with this property. Finally, we notice that the residual problem of finding the last t elements (when they exist) is precisely the problem of *syndrome decoding* of BCH codes up to half the minimum distance, for which an efficient algorithm was recently given by Dodis, Ostrovsky, Reyzin and Smith [DORS08].

We begin by stating the theorem describing the syndrome-decoding algorithm for BCH codes.

Theorem 22 (Syndrome-decoding of BCH codes [DORS08]) *There is an algorithm* SyndromeDecode*, which when given integers s, t and a vector $\mathbf{y} \in \mathbb{F}_{2^s}^t$, runs in time* $\mathrm{poly}(s)$ *and returns the unique set $U \subseteq \mathbb{F}_{2^s}^*$ (if it exists) with:*

- $|U| \leq t$,
- $\sum_{\alpha \in U} \mathbf{v}_{\alpha,t} = \mathbf{y}$.

We can now formally present our algorithm BCHSample.

- Repeat $\mathrm{poly}(t!, s)$ times:
  - Pick $\alpha_1, \ldots, \alpha_{k-t} \in \mathbb{F}_{2^n}$ uniformly at random.
  - Let $\mathbf{c} = \sum_{i=1}^{k-t} \mathbf{v}_{\alpha,t}$.
  - Run SyndromeDecode$(s, t, \mathbf{b}-\mathbf{c})$. If it runs successfully, let $U \subseteq \mathbb{F}_{2^s}$ be the set it returned.
  - Let $T = \{\alpha_1, \ldots, \alpha_{k-t}\} \cup U$.
  - If $|T| = k$, then return T and EXIT.
- FAIL.

The following lemma analyzes a single iteration of the loop, and immediately implies the correctness of the algorithm.

Lemma 23 *In a single iteration of the loop, a set T is returned with probability $1/t! - o_s(1)$, and the distribution of this set T is uniform over $\mathcal{F}$.*

Proof The lemma will follow from a sequence of claims.

Claim 24 *The distribution of* $\mathbf{c}$ *is $2^{-\Omega(st)}$-close to the uniform distribution over $\mathbb{F}_{2^s}^t$.*

Proof This is a direct consequence of Proposition 19 and Lemma 12. Let μ be the uniform distribution over S. Notice that the distribution of $\mathbf{c}$ is precisely $\mu^{(k-t)}$.

Proposition 19 implies that for every nontrivial character $\chi : \mathbb{F}_{2^s}^t \to \mathbb{R}$

$$|\mathbb{E}_{x \in \mu}[\chi(x)]| \leq 2t \cdot 2^{-s/2}.$$

Lemma 12 then implies that the statistical distance of the distribution $\mu^{(k-t)}$ from the uniform distribution over $\mathbb{F}_{2^s}^t$ is at most $2^{st} \cdot (2t)^{k-t} \cdot 2^{-(k-t)s/2} \leq 2^{-\Omega(st)}$, as desired. ■

Claim 25 *If $\mathbf{y} \in \mathbb{F}_{2^s}^t$ is picked uniformly at random, then with probability at least $1/t! - o_s(1)$,* SyndromeDecode$(s, t, \mathbf{y})$ *will succeed and output a set U with $|U| = t$. Furthermore, the distribution of this U is uniformly distributed over the set of all t-element subsets of $\mathbb{F}_{2^s}$.*

Proof Recall that $\mathsf{SyndromeDecode}(s,t,\mathbf{y})$ will succeed and output a set U of size t, if and only if $\mathbf{y}$ can be written as the sum of exactly t distinct vectors of S. However, and this is the crucial point, any $\mathbf{y}_0 \in \mathbb{F}_{2^s}^t$ can be written as the sum of t distinct elements of S in at most 1 way. Indeed, if there were 2 different ways of writing some $\mathbf{y}_0 \in \mathbb{F}_{2^s}^t$ as a sum of t distinct elements in $\mathbb{F}_{2^s}^t$, then there would be a nonempty set of size at most $2t$ elements from S which sum to 0. But this contradicts the distance property of BCH codes, Property 20.

Thus the set of all $\mathbf{y}_0 \in \mathbb{F}_{2^s}^t$ which can be written as a sum of exactly t distinct elements of S has cardinality exactly $\binom{|S|}{t} \approx \frac{2^{st}}{t!}(1-o_s(1))$. The claim follows. ■

We can now complete the proof of Lemma 23, and with that the proof of Theorem 21. By the previous 2 claims, with probability at least $1/t! - 2^{-\Omega(s)}$, $\mathsf{SyndromeDecode}(s,t,\mathbf{b}-\mathbf{c})$ will output a set U of cardinality t which is disjoint from $\{\alpha_1,\ldots,\alpha_{k-t}\}$. In this case, the algorithm will return the set T.

It remains to see that the distribution of T is uniform over $\mathcal{F}$. By design, the distribution of T is supported on $\mathcal{F}$. For every element R of $\mathcal{F}$, T will equal R if and only if $\alpha_1,\ldots,\alpha_{k-t}$ are distinct elements of R. The probability of this occurring does not depend on R, and thus the distribution of T is uniform over $\mathcal{F}$. ■ ■

Remark Observe that what we just showed also implies that for every $\mathbf{b}$, the number of $(\mathbf{x}_1,\ldots,\mathbf{x}_k) \in S_{t,s}^k$ such that the $\mathbf{x}_i$ are all distinct and $\sum \mathbf{x}_i = \mathbf{b}$ is $2^{s(k-t)}(1-o(1))$. Consequently, for every $k' < k$, and for every $\mathbf{b}$, the number of $(\mathbf{x}_1,\ldots,\mathbf{x}_{k'}) \in S_{t,s}^{k'}$ such that the $\mathbf{x}_i$ are all distinct and $\sum \mathbf{x}_i = \mathbf{b}$ is at most $2^{s(k-t-1)}(1-o(1))$. This implies that the distribution that $\mathsf{BCHSample}$ samples from (the uniform distribution over the set of all $(\mathbf{x}_1,\ldots,\mathbf{x}_k) \in S_{t,s}^k$ such that the $\mathbf{x}_i$ are all distinct and $\sum \mathbf{x}_i = \mathbf{b}$) is $2^{-\Omega(s)}$-close to the distribution $\mu^{(k,\mathbf{b})}$ (the uniform over the set of all $(\mathbf{x}_1,\ldots,\mathbf{x}_k) \in S_{t,s}^k$ such that $\sum \mathbf{x}_i = \mathbf{b}$ is $2^{s(k-t)}(1-o(1))$.

We can now complete the proof of Theorem 5.

Proof of Theorem 5: Follows by combining Theorem 21 and the remark following it, with Theorem 9 and the remark following it. ■

Acknowledgements

We are very grateful to Madhu Sudan for suggesting these problems to us, for constant encouragement, valuable discussions and for conjecturing the impossibility of our results. We are also very grateful to Adam Kalai for his enthusiastic support and many helpful discussions. Many thanks also to Henry Cohn and Tali Kaufman for all their encouragement.

Finally, we would like to thank Microsoft Research, New England for its hospitality.

References

[AKK+03] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over GF(2). In *Proceedings of RANDOM 2003, LNCS, vol. 2764*, pages 188–199, New York, 2003. Springer.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[AS03] Sanjeev Arora and Madhu Sudan. Improved low degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. Preliminary version in Proceedings of ACM STOC 1997.

[BCH+96] Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos Kiwi, and Madhu Sudan. Linearity testing over characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, November 1996.

[BEK02] Nader H. Bshouty, Nadav Eiron, and Eyal Kushilevitz. PAC learning with nasty noise. *Theor. Comput. Sci.*, 288(2):255–275, 2002.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[DG08] Irit Dinur and Elazar Goldenberg. Locally testing direct product in the low error range. In *FOCS*, pages 613–622. IEEE Computer Society, 2008.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[FGKP06] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *FOCS*, pages 563–574, 2006.

[GKZ08] Parikshit Gopalan, Adam R. Klivans, and David Zuckerman. List-decoding reed-muller codes over small fields. In Ladner and Dwork [LD08], pages 265–274.

[GL89] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, May 1989.

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM Journal on Discrete Mathematics*, 13(4):535–570, November 2000.

[IJKW08] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: simplified, optimized, and derandomized. In Ladner and Dwork [LD08], pages 579–588.

[IKW09] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. New direct-product testers and 2-query pcps. In *STOC*, pages 131–140, 2009.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR Lemma. *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, May 1997.

[KL95] Ilia Krasikov and Simon Litsyn. On spectra of BCH codes. *IEEE Transactions on Information Theory*, 41(3):786–788, 1995.

[KL05] T. Kaufman and S. Litsyn. Almost orthogonal linear codes are locally testable. In *Proceedings of the Forty-sixth Annual Symposium on Foundations of Computer Science*, pages 317–326, 2005.

[KS07] Tali Kaufman and Madhu Sudan. Sparse random linear codes are locally decodable and testable. In *FOCS*, pages 590–600. IEEE Computer Society, 2007.

[KS09] Swastik Kopparty and Shubhangi Saraf. Tolerant linearity testing and locally testable codes. In *APPROX-RANDOM*, pages 601–614, 2009.

[LD08] Richard E. Ladner and Cynthia Dwork, editors. *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. ACM, 2008.

[Lit09] Simon Litsyn, 2009. Personal Communication.

[LN94] Rudolf Lidl and Harald Niedereitter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, 2nd edition, 1994.

[Lyu05] Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *APPROX-RANDOM*, pages 378–389, 2005.

[MR06] Dana Moshkovitz and Ran Raz. Sub-constant error low degree test of almost-linear size. In Jon M. Kleinberg, editor, *STOC*, pages 21–30. ACM, 2006.

[MS81] F. J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier/North-Holland, Amsterdam, 1981.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, April 1996.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, 1997. ACM Press.

[Sam07] Alex Samorodnitsky. Low-degree tests at large distances. In *STOC*, pages 506–515, 2007.

[STV99] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 537–546, 1999.

[Sud09] Madhu Sudan, 2009. Personal Communication.

A The Goldreich-Levin local list-decoder and tester

In this section, we prove Theorem 14, which gives a constant query linearity tester under the uniform distribution (in the sense of Definition 8) in the *high error* regime. This linearity tester is a simple combination of the Goldreich-Levin theorem, Theorem 13, and the BLR linearity test [BLR93].

We begin by formally stating these results[3].

Theorem 13 ([GL89]) *For every $0 < \epsilon < 1/2$, there exists a $\left((1/2-\epsilon), \mathrm{poly}(\frac{1}{\epsilon}), \mathrm{poly}(\frac{1}{\epsilon})\right)$-local list-decoder for linear functions under U_n.*

Theorem 26 ([BLR93], The BLR Linearity Tester) *There is a tester A_{BLR} such that for any parameter ϵ such that $0 < \epsilon < 0.1$, there is a $\left(\epsilon, 50\epsilon, \mathrm{poly}(\frac{1}{\epsilon})\right)$-local tester for linearity under U_n.*

We can now prove Theorem 14.

Theorem 14 (Local List-Decoder $\implies$ Local Tester) *For every $0 < \epsilon_1 < \epsilon_2 < 1/2$, there exists a $\left(\frac{1}{2}-\epsilon_2, \frac{1}{2}-\epsilon_1, \mathrm{poly}\left(\frac{1}{\epsilon_2-\epsilon_1}\right)\right)$-local tester for linear functions under U_n.*

Proof of Theorem 14: Fix ϵ_1, ϵ_2 such that $0 < \epsilon_1 < \epsilon_2 < 1/2$, and let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a probabilistic function. Let $\delta = \frac{\epsilon_2-\epsilon_1}{100}$.

We now describe the $\left(\frac{1}{2}-\epsilon_2, \frac{1}{2}-\epsilon_1, \mathrm{poly}(\frac{1}{\epsilon_2-\epsilon_1})\right)$-local tester for linearity under U_n.

The Tester: The tester works in three stages:

1. Run the $\left((1/2-\epsilon_2), \mathrm{poly}(\frac{1}{\epsilon_2}), \mathrm{poly}(\frac{1}{\epsilon_2})\right)$-local list-decoder $A_{decoder}$ for linear functions under U_n, given by Theorem 13, $O(1)$ times. Each time $A_{decoder}$ outputs a list of $\mathrm{poly}(\frac{1}{\epsilon_2})$ randomized algorithms. Let $\{A_1, \ldots, A_l\}$ be the union of all these lists, where $l = \mathrm{poly}(\frac{1}{\epsilon_2})$.

   For each $i \in [l]$, let $B_i : \mathbb{F}_2^n \to \mathbb{F}_2$ be the randomized algorithm which on input x, executes A_i^f on x $O(\log(\frac{1}{\delta}))$ times, and outputs the majority of the outputs of these executions. Thus, if A_i computes a linear function g with error $1/3$, then B_i computes that linear function with error $\delta^{\Omega(1)}$.

2. For each B_i $(1 \leq i \leq l)$, run the BLR linearity test (given by Theorem 26) with parameter δ to "test" if B_i is δ-close to a linear function. Repeat the test $\mathrm{poly}(l)$ times and take majority of the output accept/reject decision.

3. For each B_i, estimate the U_n-distance of f from B_i in the following manner:

   (a) uniformly and independently sample $\mathrm{poly}(\frac{1}{\delta}, l)$ elements of $\mathbb{F}_2^n$, say $x_1, x_2, \ldots, x_m$,

[3]To be precise, we state generalizations of the usual Goldreich-Levin and Blum-Luby-Rubinfeld theorems to handle probabilistic functions; inspecting the original proofs of these theorems immediately reveals that they apply verbatim to probabilistic functions too.

(b) for each sample x_j, $1 \leq j \leq m$, take a sample of the evaluation of both f and B_i on x_j (recall that f and B_i are probabilistic functions and hence the evaluations are random variables).

(c) let the *estimated distance* between f and B_i be the fraction of the x_j ($i \leq j \leq m$) for which $f(x_j) \neq B_i(x_j)$.

If there is any B_i such that B_i got accepted by the majority vote of the repeated BLR test, and the estimated distance of f from B_i is less than $1/2 - \epsilon_1 - 60\delta$, then the Tester accepts, else it rejects.

Analysis of the Tester: First observe that the number of oracles calls to f made by the tester (in stages 1, 2 and 3) is poly$(l, \frac{1}{\delta}$. Now $l = \text{poly}(\frac{1}{\epsilon_2})$ and $\delta = \frac{\epsilon_2 - \epsilon_1}{100} < \epsilon_2$. Hence the number of oracle calls is at most poly$(\frac{1}{\epsilon_2 - \epsilon_1})$.

We want to show that for any (probabilistic) function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ whose U_n-distance from the set of linear functions is less than $1/2 - \epsilon_2$, the above Tester accepts with probability at least $2/3$, and for any function f whose U_n-distance from the set of linear functions is at least $1/2 - \epsilon_1$, the Tester rejects with probability at least $2/3$.

Case 1: Let f be a function such that the U_n-distance of f from the set of linear functions is less than $1/2 - \epsilon_2$. Let $g : \mathbb{F}_2^n \to \mathbb{F}_2$ be a linear function such that $\Delta_{U_n}(f, g) < 1/2 - \epsilon_2$. By Theorem 13, each time the Goldreich-Levin $\Big((1/2 - \epsilon_2), \text{poly}(\frac{1}{\epsilon_2}), \text{poly}(\frac{1}{\epsilon_2})\Big)$-local list-decoder $A_{decoder}$ is run, with probability at least $2/3$, $A_{decoder}$ returns some function A_i such that A_i^f implicitly computes g. In stage 1 of the tester, since we run $A_{decoder}$ multiple times, with probability at least $9/10$, in at least one of the iterations $A_{decoder}$ will return an algorithm A_i such that A_i^f implicitly computes g. By construction, this implies that $\Delta_{U_n}(B_i, g) \leq \delta$. Thus the BLR linearity test (given by Theorem 26) with parameter δ when applied to B_i will return "Accept" with probability at least $2/3$. Thus the repeated version of the BLR test given in stage 2 will accept B_1 with probability at least $9/10$. Finally, by the Chernoff bound, with probability at least $9/10$, the estimated distance between f and B_i is at most $\Delta_{U_n}(f, B_i) + \delta/2 < \Delta_{U_n}(g, B_i) + \delta + \delta/2 < 1/2 - \epsilon_2 + 3\delta/2 < 1/2 - \epsilon_1 - 60\delta$.

Thus with probability at least $7/10 > 2/3$, the tester accepts.

Case 2: Let f be a function such that the U_n-distance of f from the set of linear functions is greater than $1/2 - \epsilon_1$. Fix any B_i returned by stage 1. Now, if B_i is at least 50δ far from linear, then then BLR linearity test will accept B_i with probability at most $1/3$, and the repeated BLR test given in stage 2 will accept f with probability at most $(1/100) \cdot l$. If the U_n-distance of B_i from linear is at most 50δ, then since the distance of f from the set of linear functions is greater than $1/2 - \epsilon_1$, the triangle inequality implies that $\Delta_{U_n}(B_i, f) \geq 1/2 - \epsilon_1 - 50\delta$. Hence, for B_i to be accepted, the estimated distance of B_i from f must deviate from the true U_n-distance by at least 10δ. By the Chernoff bound, this happens with probability at most $(1/100) \cdot l$. Hence, in either case, the tester will accepted with probability at most $(1/50) \cdot l$. Since there are a most l different functions B_i, by the union bound, the probability that the Tester accepts f is at most $1/50 < 1/3$.

■

ECCC ISSN 1433-8092

http://eccc.hpi-web.de