Electronic Colloquium on Computational Complexity, Report No. 60 (1998)

Learning polynomials with queries: The highly noisy case[*]

ODED GOLDREICH[†] RONITT RUBINFELD[‡] MADHU SUDAN[§]

October 8, 1998

Abstract

Given a function f mapping n-variate inputs from a finite field F into F, we consider the task of reconstructing a list of all n-variate degree d polynomials which agree with f on a tiny but non-negligible fraction, δ, of the input space. We give a randomized algorithm for solving this task which accesses f as a black box and runs in time polynomial in $\frac{1}{\delta}, n$ and exponential in d, provided δ is $\Omega(\sqrt{d/|F|})$. For the special case when $d = 1$, we solve this problem for all $\epsilon \stackrel{\rm def}{=} \delta - \frac{1}{|F|} > 0$. In this case the running time of our algorithm is bounded by a polynomial in $\frac{1}{\epsilon}$ and n. Our algorithm generalizes a previously known algorithm, due to Goldreich and Levin, that solves this task for the case when $F = \mathrm{GF}(2)$ (and $d = 1$).

In the process we provide new bounds on the number of degree d polynomials that may agree with any given function on $\delta \geq \sqrt{d/|F|}$ fraction of the inputs. This result is derived by generalizing a well-known bound from coding theory on the number of codewords from an error-correcting code that can be "close" to an arbitrary word — our generalization works for codes over arbitrary alphabets, while the previous result held only for binary alphabets. This result may be of independent interest.

1 Introduction

We consider the following archetypal *reconstruction problem*:

Given: An oracle (black box) for an arbitrary function $f : F^n \to F$, a class of functions $\mathcal{C}$, and a parameter δ.
Output: A list of all functions $g \in \mathcal{C}$ that agree with f on at least δ fraction of the inputs.

[*]A preliminary version of this paper appeared in *36th Annual Symposium on Foundations of Computer Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 October 1995. IEEE.

[†]Weizmann Institute of Science. Currently visiting MIT Lab for Computer Science. `email: oded@wisdom.weizmann.ac.il`. Supported by grant No. 92-00226 from the United States – Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[‡]Cornell University. `email: ronitt@cs.cornell.edu`. This work is supported by ONR Young Investigator Award N00014-93-1-0590 and grant No. 92-00226 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[§]MIT Laboratory for Computer Science, Cambridge, MA 02139. Email: `madhu@lcs.mit.edu`.

ISSN 1433-8092

The reconstruction problem can be interpreted in several ways within the framework of computational learning theory. First, it falls into the paradigm of learning with persistent noise. Here one assumes that the function f is derived from some function in the class $\mathcal{C}$ by "adding" noise to it. Typical works in this direction either tolerate only small amounts of noise [2, 41, 21, 39] (i.e., that the function is modified only at a small fraction of all possible inputs) or assume that the noise is random [1, 26, 20, 25, 33, 13, 36] (i.e., that the decision of whether or not to modify the function at any given input is made by a random process). In contrast, we take the setting to an extreme, by considering a *very large amount* of (possibly adversarially chosen) noise. In particular, we consider situations in which the noise disturbs the outputs for almost all inputs.

A second interpretation of the reconstruction problem is within the paradigm of "agnostic learning" introduced by Kearns et al. [23] (see also [29, 30, 24]). In the setting of agnostic learning, the learner is to make no assumptions regarding the natural phenomena underlying the input/output relationship of the function, and the goal of the learner is to come up with a simple explanation which best fits the examples. Therefore the best explanation may account for only part of the phenomena. In some situations, when the phenomena appears very irregular, providing an explanation which fits only part of it is better than nothing. Kearns et al. did not consider the use of queries (but rather examples drawn from an arbitrary distribution) as they were skeptical that queries could be of any help. We show that queries do seem to help (see below).

Yet another interpretation of the reconstruction problem, which generalizes the "agnostic learning" approach, is the following. Suppose that the natural phenomena can be explained by several simple explanations which together cover most of the input-output behavior but not all of it. Namely, suppose that the function f agrees almost everywhere with one of a small number of functions $g_i \in \mathcal{C}$. In particular, assume that each g_i agrees with f on at least a δ fraction of the inputs but that for some (say 2δ) fraction of the inputs f does not agree with any of the g_i's. This setting is very related to the setting investigated by Ar et al. [3], except that their techniques require that the fraction of inputs left unexplained by any g_i be smaller than the fraction of inputs on which each g_i agrees with f. We believe that our relaxation makes the setting more appealing and closer in spirit to "agnostic learning".

1.1 Our Results

In this paper, we consider the special case of the reconstruction problem when the hypothesis class is the set of n-variate polynomials of bounded total degree d. The most interesting aspect of our results is that they relate to very small values of the parameter δ (the fraction of inputs on which the hypothesis has to fit the function f). Our main results are

- An algorithm that given d, F and $\delta = \Omega(\sqrt{d/|F|})$, and provided oracle access to an arbitrary function $f : F^n \to F$, runs in time $(n/\delta)^{O(d)}$ and outputs a list including all degree d polynomials which agree with f on δ fraction of the inputs.
- An algorithm that given F and $\epsilon > 0$, and provided oracle access to an arbitrary function $f : F^n \to F$, runs in time $\text{poly}(n/\epsilon)$ and outputs a list including all linear functions (degree $d = 1$ polynomials) which agree with f on a $\delta \stackrel{\text{def}}{=} \frac{1}{|F|} + \epsilon$ fraction of

the inputs.

- A new bound on the number of degree d polynomials that may agree with a given function $f : F^n \to F$ on a $\delta \geq \sqrt{d/|F|}$ fraction of the inputs. This bound is derived from a more general result about the number of codewords from an error-correcting code that may be close to a given word.

A special case of interest is when the function f is obtained as a result of picking an arbitrary degree d polynomial p and letting f agree with p on an arbitrary $\delta = \Omega(\sqrt{\frac{d}{|F|}})$ fraction of the inputs and be set at random otherwise.[1] In this case, with high probability, only one polynomial (i.e., p) agrees with f on a δ fraction of the inputs (see Section 5). Thus, in this case, the above algorithm will output only the polynomial p.

Remarks:

1. Any algorithm for the (explicit) reconstruction problem as stated above would need to output all the coefficients of such a polynomial, requiring time at least $\binom{n+d}{d}$. Moreover the number of such polynomials could grow as a function of $\frac{1}{\delta}$. Thus it seems reasonable that the running time of such a reconstruction procedure should grow as a polynomial function of $\frac{1}{\delta}$ and $\binom{n}{d}$.

2. For $d < |F|$, the value $\frac{d}{|F|}$ seems a natural threshold for our investigation since there are exponentially (in n) many degree d polynomials which are at distance $\approx \frac{d}{|F|}$ from some functions (see Proposition 21).

3. Queries seem essential to our learning algorithm since for the case $F = GF(2)$ and $d = 1$ the problem reduces to the well-known problem of "learning parity with noise" [20] which is commonly believed to be hard when one is only allowed uniformly and independently chosen examples [20, 7, 22]. (Actually, learning parity with noise is considered hard even for random noise, whereas here the noise is adversarial.)

4. In Section 6, we give evidence that the reconstruction problem may be hard, for δ very close to $d/|F|$, even in the case where $n = 2$. A variant is shown to be hard even for $n = 1$.

List decoding of error-correcting codes: For positive integers N, K, D and q, an $[N, K, D]_q$ error-correcting code is a collection of q^K sequences of N-elements each from $\{1, \ldots, q\}$, called codewords, in which no two strings have a "Hamming distance" of less than D (i.e., every pair of codewords disagree on at least D locations). Polynomial functions lead to some of the simplest known constructions of error-correcting codes: A function from F^n to F may be viewed as an element of $F^{|F|^n}$ — by writing down the function's value explicitly on all $|F|^n$ inputs. Then the "distance property" of polynomials yields that the set of sequences corresponding to bounded-degree polynomial functions form an error-correcting code with non-trivial parameters (for details, see Proposition 16). These constructions have been studied in the literature on coding theory. The case $n = 1$ leads to "Reed-Solomon

[1] This is different from "random noise" as the set of corrupted inputs is selected adversarially – only the values at these inputs are random.

codes", while the case of general n is studied under the name "Reed-Muller codes". Our reconstruction algorithm may be viewed as an algorithm that takes an arbitrary word from $F^{|F|^n}$ and finds a list of all codewords from the Reed-Muller code that agree with the given word in δ fraction of the coordinates (i.e., $1-\delta$ fraction of the coordinates have been corrupted by errors). This task is referred to in the literature as the "list-decoding" problem [11]. For codes achieved from setting d such that $d/|F| \to 0$, our list decoding algorithm recovers from errors when the rate of errors approaches 1. We are not aware of any other case where an approach other than brute force can be used to perform list decoding with the error-rate approaching 1. Furthermore, our decoding algorithm works without examining the entire codeword. Our algorithms seem to be non-trivial and have better running times than the brute force algorithm for list-decoding.

1.2 Related Previous Work

Polynomial interpolation: When the noise rate is 0, our problem is simply that of polynomial interpolation. In this case the problem is well analyzed and the reader is referred to [46], for instance, for a history of the polynomial interpolation problem.

Self-Correction: In the case when the noise rate is positive but small, one approach used to solving the reconstruction problem is to use *self-correctors*, introduced independently in [8] and [28]. Self-correctors convert programs that are known to be correct on a fraction δ of inputs into programs that are correct on each input. Self-correctors for values of δ that are larger than 3/4 have been constructed for several functions [5, 8, 9, 28, 34]. Self-correctors correcting $\frac{1}{\Theta(d)}$ fraction of error for f that are degree d polynomial functions over a finite field F, $|F| \geq d+2$, were found by [5, 28]. For $d/|F| \to 0$, the fraction of errors that a self-corrector could correct was improved to almost 1/4 by [14] and then to almost 1/2 by [15] (using a solution for the univariate case given by [43]). These self-correctors correct a given program using only the information that the program is supposed to be computing a low-degree polynomial. Thus, when the error is larger than $\frac{1}{2}$ (or, alternatively $\delta < 1/2$), such self-correction is no longer possible since there could be more than one polynomial that agrees with the given program on an $\delta < 1/2$ fraction of the inputs.

Linear Polynomials: A special case of the (explicit) reconstruction problem for $d = 1$ and $F = \mathrm{GF}(2)$ was studied by Goldreich and Levin [17]. They solved the reconstruction problem in this case for every $\delta > \frac{1}{2}$. (Notice that the self-corrector of [15] mentioned above does not apply to this case, since here $d/|F| = 1/2$ and does not tend to 0.) Goldreich and Levin [17] use the solution to the reconstruction problem to prove the security of a certain "hardcore predicate" relative to any "one-way function". Their ideas were also subsequently used by Kushilevitz and Mansour [25] to devise an algorithm for learning Boolean decision trees.

Reconstruction of polynomials under structured error models: Ar et al. [3] have considered the problem of reconstructing a list of polynomials which together explain the input-output relation of a given black-box. However, they have required that the fraction of inputs left uncovered by any of the polynomials be smaller than the fraction of inputs covered by any single polynomial.

1.3 Subsequent work

At the time this work was done (and first published [18]) no algorithm (other than brute force) was known for reconstructing a list of degree d polynomials agreeing with an arbitray function on a vanishing fraction of inputs, for any $d \geq 2$. Our algorithm solves this problem with exponential dependence on d, but with polynomial dependence on n, the number of variables. Subsequently some new reconstruction algorithms for polynomials have been developed. In particular, Sudan [40], and Guruswami and Sudan [19] have provided new algorithms for reconstructing univariate polynomials from large amounts of noise. Their running time depends only polynomially in d and works for $\delta = \Omega(\sqrt{d/|F|})$. Notice that the agreement required in this case is larger than the level at which our NP-hardness result holds. The results of [40] also provide some reconstruction algorithms for multivariate polynomials, but not for as low an error as given here. Also in his case, the running time grows exponentially with n. Wasserman [42] gives an algorithm for reconstructing polynomials from noisy data that works *without* making queries. The running time here also grows exponentially in n and polynomially in d.

As noted earlier (see Remark 1 in Section 1.1), the running time of any explicit reconstruction algorithm has to have an exponential dependence on either d or n. However this need not be true for implicit reconstruction algorithms, i.e., algorithms which produce as output a sequence of oracle machines, such that for every multivariate polynomial that has agreement δ with the function f, one of these oracle machines, given access to f, computes that polynomial. Recently, Arora and Sudan [4] gave an algorithm for this implicit reconstruction problem. The running time of their algorithm is bounded by a polynomial in n, d and works correctly provided $\delta \geq (d^{O(1)})/|F|^{\Omega(1)}$ (i.e., their algorithm needs a much higher agreement, but works in time polynomial in all parameters). It is easy to see that such a result implies an algorithm for the explicit reconstruction problem with running time that is polynomial in $\binom{n+d}{d}$ and $\frac{1}{\delta}$.

1.4 Rest of this paper

The rest of the paper is organized as follows. In Section 2 we motivate our algorithm and in particular present the case of reconstruction of linear polynomials. The algorithm is described formally in Section 3, along with an analysis of its correctness and running time assuming an upper bound on the number of polynomials which can agree with a given function at δ fraction of the inputs. In Section 4 we provide two such upper bounds. These bounds do not use any special (i.e., algebraic) property of polynomials, but apply in general to collections of functions that have large distance between them. In Section 5 we consider a random model for the noise in the function. Specifically, the output of the black box either agrees with a fixed polynomial or is random. In such a case we provide a stronger upper bound (i.e., 1) on the number of polynomials that may agree with the black box. In Section 6 we give evidence that the reconstruction problem may be hard for small values of the agreement parameter δ even in the case when $n = 1$.

Notations: In what follows, we use GF(q) to denote the finite field on q elements. We assume arithmetic in this field (addition, subtraction, multiplication, division and comparison with zero) may be performed for unit cost. For a finite set A, we use the notation $a \in_R A$ to denote that a is a random variable chosen uniformly at random from A. For a positive integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$.

2 Motivation to the algorithm

We start by presenting the algorithm for the linear case, and next present some of the ideas underlying the generalization to higher degrees.

2.1 Reconstructing linear polynomials

We are given oracle access to a function $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ and need to find a polynomial (or actually all polynomials) of degree d which agrees with f on an $\delta = \frac{1}{q} + \epsilon$ fraction of the inputs, where $\epsilon > 0$.

Our starting point is the linear case (i.e., $d = 1$); namely, we are looking for a polynomial of the form $p(x_1, ..., x_n) = \sum_{i=1}^{n} c_i x_i$. In this case our algorithm is a generalization of an algorithm due to Goldreich and Levin [17][2]. (The original algorithm is regained by setting $q = 2$.) To proceed, we need the following definition: The *i-prefix* of a linear polynomial $p(x_1, ..., x_n)$ is the polynomial which results by summing up all of the monomials in which only the first i variables appear. That is, the i-prefix of the polynomial $\sum_{j=1}^{n} c_j x_j$ is $\sum_{j=1}^{i} c_j x_j$. The algorithm proceeds in n rounds, so that in the i^{th} round we find a list of candidates for the i-prefixes of p.

The list of i-prefixes is generated by extending the list of $(i-1)$-prefixes. A simple (and inefficient) way to perform this extension is to first extend each $(i-1)$-prefix in all q possible ways, and then to screen the resulting list of i-prefixes. A good screening is the essence of the algorithm. It should guarantee that the i-prefix of the correct solution p does pass and that not too many other prefixes pass (as otherwise the algorithm consumes too much time).

The screening is done by subjecting each candidate prefix, $(c_1, ..., c_i)$, to the following test. Pick $m = \mathrm{poly}(n/\epsilon)$ sequences uniformly from $\mathrm{GF}(q)^{n-i}$. For each such sequence $(s_{i+1}, ..., s_n)$ and for every $\sigma \in \mathrm{GF(q)}$, estimate the quantity

$$P(\sigma) \stackrel{\mathrm{def}}{=} \mathbf{Pr}_{r_1,\ldots,r_i \in \mathrm{GF}(q)} \left[f(\bar{r}, \bar{s}) = \sum_{j=1}^{i} c_j r_j + \sigma \right] \qquad (1)$$

(where $\bar{r}, \bar{s}$ denotes the vector $(r_1, \ldots, r_i, s_{i+1}, \ldots, s_n)$). The value σ can be thought of as a guess for $\sum_{j=i+1}^{n} c_j s_j$. All these probabilities can be approximated simultaneously by using a sample of $\mathrm{poly}(n/\epsilon)$ sequences $(r_1, ..., r_i)$ (regardless of q). The actual algorithm is presented in Figure 1. We say that a candidate $(c_1, ..., c_i)$ passes the test if for at least one sequence of $(s_{i+1}, ..., s_n)$ there exists a σ so that the estimate for $P(\sigma)$ is greater than $\frac{1}{q} + \frac{\epsilon}{3}$. An application of Markov's inequality yields that for the correct candidate, an $\epsilon/2$ fraction of the suffixes $(s_{i+1}, \ldots, s_n)$ are such that for this suffix, $P(\sigma)$ is at least $\frac{1}{q} + \frac{\epsilon}{2}$; thus the correct candidate

[2]We refer to the original algorithm as in [17], not to a simpler algorithm which appears in later versions (cf., [27, 16]).

```
Test-prefix(f, ε, n, (c_1, ..., c_i))
  Repeat poly(n/ε) times:
    Pick s_{i+1}, ..., s_n ∈_R GF(q).
    Let t =def poly(n/ε).
    for k = 1 to t do
      Pick r_1, ..., r_i ∈_R GF(q)
      σ^(k) ← f(r̄, s̄) − Σ_{j=1}^{i} c_j r_j .
      endfor
    If ∃ σ s.t.  σ^(k) = σ for at least 1/q + ε/3 fraction of the k's
      then output accept and halt.
    endRepeat.
  If all iterations were completed without accepting, then reject.
```

Figure 1: Implementing the screeing process

passes the test with overwhelming probability. On the other hand, for any polynomial to pass the test with non-negligible probability, it must have agreement at least $\frac{1}{q}+\frac{\epsilon}{4}$ with f, and it is possible to bound the number of polynomials that have so much agreement with any function f. (Sections 4 contains some such bounds.)

The above yields a poly(nq/ϵ)-time algorithm. In order to get rid of the q factor in running-time, we need to modify the process by which candidates are formed. Instead of extending each $(i-1)$-prefix, $(c_1,...,c_{i-1})$, in all q possible ways, we do the following: We pick uniformly $\overline{s} \stackrel{\text{def}}{=} (s_{i+1},...,s_n) \in \text{GF}(g)^{n-i}$, $\overline{r} \stackrel{\text{def}}{=} (r_1,...,r_{i-1}) \in \text{GF}(q)^{i-1}$ and $r', r'' \in \text{GF}(q)$. Note that if p is a solution to the reconstruction problem for f then for at least an $\epsilon/2$ fraction of the sequences $(\overline{r},\overline{s})$, the polynomial p satisfies $p(\overline{r},r,\overline{s}) = f(\overline{r},r,\overline{s})$ for at least an $\epsilon/2$ fraction of the possible r's. We may assume that $1/q < \epsilon/4$ (since otherwise $q < 4/\epsilon$ and we can afford to perform the simpler procedure above). Denote by y the unknown value of the sum $\sum_{j=i+1}^{n} c_j s_j$ (where these c_j's are the coefficient of the polynomial close to f) and by x the coefficient we are looking for (i.e., the i^{th} coefficient c_i). Then, with probability $\Omega(\epsilon^3)$ over the choices of $r_1,\ldots,r_{i-1},s_{i+1},\ldots,s_n$ and r',r'', the following two equations hold:

$$
\begin{aligned}
r'x + y &= f(r_1,...,r_{i-1},r',s_{i+1},...,s_n) - \sum_{j=1}^{i-1} c_j r_j \\
r''x + y &= f(r_1,...,r_{i-1},r'',s_{i+1},...,s_n) - \sum_{j=1}^{i-1} c_j r_j
\end{aligned}
$$

where $r' \neq r''$o. (I.e., with probability at least $\frac{\epsilon}{2}$, the suffix $\bar{s}$ is good, and conditioned on this event r' is good with probability at least $\frac{\epsilon}{/}2$, and the same applies to r'' while conditioning on $r'' \neq r'$.) Thus solving for x we get the desired extension. We emphasize that we do not know whether the equalities hold or not, but rather solve assuming they hold and add the solution to the list of candidates. In order to guarantee that the correct prefix always appears in our candidate list, we repeat the above extension poly(n/ϵ) times for each $(i-1)$-prefix. Extraneous prefixes can be removed from the candidate list via the screening process mentioned above. Using Theorem 18 of Section 4, we have:

Theorem 1 *Given oracle access to a function f and parameters ϵ, k, our algorithm runs in* poly$(\frac{k \cdot n}{\epsilon})$*-time and outputs, with probability at least $1 - 2^{-k}$, a list containing all linear polynomials which agree with f on at least a $\delta = \frac{1}{q} + \epsilon$ fraction of the inputs. Furthermore, the list does not contain polynomials which agree with f on less than a $\frac{1}{q} + \frac{\epsilon}{4}$ fraction of the inputs.*

2.2 Generalizing to higher degree

Dealing with polynomials of degree $d > 1$ is more involved. Our plan is (again) to first "isolate" the terms/monomials of in the first i variables and find (candidates for) their coefficients. The isolation can be performed by concentrating on small subcubes of the domain. In particular, if $p(x_1, \ldots, x_n)$ is a degree d polynomial on n variables then $p(x_1, \ldots, x_i, 0, \ldots, 0)$ is a degree $\leq d$ polynomial on i variables which has the same coefficients as p on all monomials involving variables in $\{1, \ldots, i\}$. Thus, $p(x_1, \ldots, x_i, 0, \ldots, 0)$ is the *i-prefix* of p.

We show how to extend a list of candidates for the $(i-1)$-prefixes polynomials agreeing with f into a list of candidates for the i-prefixes. Suppose we get the $(i-1)$-prefix p which we want to extend. We select $d+1$ distinct elements $r^{(1)}, ..., r^{(d+1)} \in \mathrm{GF}(q)$ ($r^{(j)} = j$ is a reasonable choice). Now consider the functions

$$f^{(j)}(x_1, ..., x_{i-1}) \stackrel{\text{def}}{=} f(x_1, ..., x_{i-1}, r^{(j)}, 0, ..., 0) - p(x_1, ..., x_{i-1}). \tag{2}$$

Suppose that f equals some degree d polynomial and that p is indeed the $(i-1)$-prefix of this polynomial. Then $f^{(j)}$ is a polynomial of degree $d-1$ (since all the degree d monomials in the $i-1$ variables have been canceled by p). Furthermore, *given* $f^{(1)}, ..., f^{(d+1)}$, we can find (by interpolation) the extension of p to a i-prefix. The last assertion deserves some elaboration. Consider the i-prefix of f, denoted $p' = p'(x_1, ..., x_{i-1}, x_i)$. In each $f^{(j)}$ the monomials of p' which agree on the exponents of $x_1, ..., x_{i-1}$ are collapsed together (since x_i is instantiated and so monomials containing different powers of x_i are added together). However, using the $d+1$ collapsed values, we can retrieve the coefficients of the different monomials (in p'). (Actually, we obtain a degree d polynomial in variables $x_1, ..., x_i$ which matches each $f^{(j)}$ when instantiating $x_i = r^{(j)}$, but only the degree d monomials in this polynomial correspond to p' – as they are not affected by the instantiation of $x_{i+1}, ..., x_n$.) To complete the high level description of the procedure we need to get the polynomial representing the $f^{(j)}$'s. Since in reality we have only have access to a (possibly highly noisy) oracle for the $f^{(j)}$'s, we use the main procedure for finding a list of candidates for these polynomials. We point out that the recursive call is to a problem of degree $d-1$, which is lower than the degree we are currently handling.

One difficulty encountered is the following: Suppose that there is some polynomial p^* that agrees with f on at least a δ fraction of inputs in $\mathrm{GF}(q)^n$, but p^* does not agree with f at all on the inputs in $\mathrm{GF}(q)^i 0^{n-i}$. Then solving the subproblem gives us no information about p^*. Our solution is to perform a random linear transformation of the coordinate system as follows: Pick a random nonsingular matrix R and define new variables $y_1, \ldots, y_n$ as $(y_1, \ldots, y_n) = \bar{y} \equiv R\bar{x}$ (each y_i is a random linear combination of the x_i's and vice versa). This transformation can be used to define a new interpolation problem in terms of the y_i's, where (1) the total degree of the problem is preserved (2) the points are mapped pairwise independently so that there are "good" points in all subspaces of the new problem and (3) one can easily transform the coordinate system back to the x_i's, so that it is possible to construct a new black box consistent with f that takes $\bar{y}$ as an input.

Comment: The above solution to the above difficulty is different than the one in the original version of this paper [18]. The solution there was to pick many different suffixes (instead of 0^{n-i}) and then apply Markov's inequality. Picking many different suffixes creates other problems, that needed to be dealt with carefully; and this resulted in a more complicated algorithm in the original version.

3 Algorithm for degree $d > 1$ polynomials

The main algorithm Find-all-poly will use several subroutines: Compute-coefficients, Test-valid, Constants, Brute-force, and Extend. The main algorithm is recursive, in n and d, with the base case $d = 0$ being handled by the subroutine Constants and the other bases cases, $n \leq 4$ being handled by the subroutine Brute-force. Most of the work is done in Find-all-poly and Extend, which are mutually recursive.

The algorithms have a number of parameters in their input. We describe the commonly occuring parameters first: q is the size of the field we will be working with (and, unlike other parameters, this never changes in the recursive calls). f will be a function on many variables from GF(q) to GF(q) given as an oracle. n will denote the number of variables of f. d will denote the degree of the polynomial we are hoping to reconstruct and δ, the agreement parameter, is threshold such that every degree d polynomial with agreement at least δ with f will be reconstructed by the algorithm. Many of the algorithms are probabilistic and make two-sided error. ψ will be the error parameter controlling the probability with which a valid solution may be omitted from the output. ϕ will be the error parameter controlling the error with which an invalid solution is included in the output list. Picking a random element of GF(q) is assumed to take unit time, as are field operations and calls to the oracle f.

The symbol x will typically stand for a vector in GF(q), while the notation x_i will refer to the ith coordinate of x. When picking a sequence of vectors, we will use superscripts to denote the vectors in the sequence. Thus, $x_i^{(j)}$ will denote the ith coordinate a the jth element of the sequence of vectors $x^{(1)}, x^{(2)}, \ldots$. For two polynomials p_1 and p_2, we write $p_1 \equiv p_2$ if p_1 and p_2 are identical polynomials. We now generalize the notion of the prefix of a polynomial in two ways. We extend it to arbitrary functions, and then extend it to arbitrary suffixes (and not just 0^i.

Definition 2 *For $1 \leq i \leq n$ and $a_1, \ldots, a_{n-i} \in F$, the $(a_1, \ldots, a_{n-i})$-prefix of a function $f : F^n \to F$, denoted $f|_{a_1,\ldots,a_{n-i}}$, is the i-variate function $f|_{a_1,\ldots,a_{n-i}} : F^i \to F$, given by $f|_{a_1,\ldots,a_{n-i}}(x_1, ..., x_i) = f(x_1, \ldots, x_i, a_1, \ldots, a_{n-i})$. The i-prefix of f is the function $f|_{0^{n-i}}$.*

Remark: When specialized to a polynomial p, the i-prefix of p yields a polynomial on the variables $x_1, \ldots, x_i$ whose coefficients are exactly the coefficients of p on monomials involving only $x_1, \ldots, x_i$.

We will use the notation $N_{n,d,\delta}$ to denote the maximum (over all possible f) of the number of polynomials of degree d in n variables that have agreement δ with f. In this section we will first determine our running time as a function of $N_{n,d,\delta}$ and then use bounds on $N_{n,d,\delta}$ derived in Section 4 to derive the absolute running times. We include the intermediate bounds since it is possible that the bounds of Section 4 may be improved, and this would improve our running time as well. We only highlight some properties that we need to use about this quantity. By definition $N_{n,d,\delta}$ is monotone non-decreasing in d and n and monotone non-increasing in δ. This will be used in the analysis.

3.1 The subroutines

We first axiomatize the behavior of each of the subroutines. Then we present an implemention of the subroutine and then analyze it with respect to the axiomatization.

(P1) $\mathsf{Constants}(f, \delta, n, q, \psi)$, with probability at least $1 - \psi$, returns every degree 0 (i.e., constant) polynomial p such that f and p agree on δ fraction of the points. [3]

$\mathsf{Constants}$ works as follows: Set $k = O((\frac{1}{\delta^2}) \log \frac{1}{\psi})$ and pick $x^{(1)}, \ldots, x^{(k)}$ independently and uniformly at random from $\mathrm{GF(q)}^n$. Output the list of all constants a (or equivalently the polynomial $p_a = a$) such that $|\{i \in [k] | f(x^{(i)}) = a\}| \geq \frac{3}{4}\delta k$.

An easy application of Chernoff bounds indicates that the setting $k = O(\frac{1}{\delta^2} \log \frac{1}{\psi})$ suffices to ensure that the error probability is at most ψ. Thus the running time of $\mathsf{Constants}$ is bounded by the time to pick $x^{(1)}, \ldots, x^{(k)} \in \mathrm{GF(q)}^n$ which is $O(\frac{1}{\delta^2} n \log \frac{1}{\psi})$.

Proposition 3 $\mathsf{Constants}(f, \delta, n, q, \psi)$ *satisfies* **(P1)**. *Its running time is* $O(\frac{1}{\delta^2} n \log \frac{1}{\psi})$.

Another simple procedure is the testing of agreement between a given polynomial and a black box.

(P2) $\mathsf{Test\text{-}valid}(f, p, \delta, n, d, q, \psi, \phi)$ returns true, with probability at least $1 - \psi$, if p is an n-variate degree d polynomial with agreement δ with f. It returns false with probability at least $1 - \phi$ if the agreement between f and p is less than $\frac{\delta}{2}$. (It may return anything if the agreement is between $\frac{\delta}{2}$ and δ.)

$\mathsf{Test\text{-}valid}$ works as follows: Set $k = O(\frac{1}{\delta^2} \log \frac{1}{\min\{\psi, \phi\}})$ and pick $x^{(1)}, \ldots, x^{(k)}$ independently and uniformly at random from $\mathrm{GF(q)}^n$. If $f(x^{(i)}) = p(x^{(i)})$ for at least $\frac{3}{4}\delta$ fraction of the values of $i \in [k]$ then output true else false.

Again an application of Chernoff bounds yields the correctness of $\mathsf{Test\text{-}valid}$. The running time of $\mathsf{Test\text{-}valid}$ is bounded by the time to pick the k points from $\mathrm{GF(q)}^n$ and the time to evaluate p on them which is $O(\frac{1}{\delta^2} (\log \frac{1}{\psi}) \binom{n+d}{d} \cdot dn)$.

[3] Notice that we do not make any claims about the probability with which constants that do not have significant agreement with ϕ may be reported. In fact we do not need such a condition for our analysis. If required, such a condition may be explicitly enforced by "testing" every constant that is returned for sufficient agreement.

Brute-force$(f, \delta, n, d, q, \psi, \phi)$
  Set $l = \binom{n+d}{d}$
    $k = O(\log \frac{1}{\psi} (\frac{1}{\delta - \frac{d}{q}})^l)$
    $\mathcal{L} \leftarrow \lambda$.
  Repeat k times
    Pick $x^{(1)}, \ldots, x^{(l)} \in_R \mathrm{GF(q)}^n$.
    Multivariate interpolation step:
      Find $p : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ of degree d s.t. $\forall i \in [l]$, $p(x^{(i)}) = f(x^{(i)})$.
    If Test-valid$(f, p, \delta, n, d, q, \frac{1}{2}, \phi)$ then $\mathcal{L} \leftarrow \mathcal{L} \cup \{p\}$.

Figure 2: Brute-force

Proposition 4 Test-valid$(f, p, \delta, n, d, q, \psi)$ *satisfies* **(P2)**. *Its running time is bounded by* $O(\frac{1}{\delta^2}(\log \frac{1}{\psi}) \binom{n+d}{d}^{O(1)})$.

(P3) Brute-force$(f, \delta, n, d, q, \psi, \phi)$ returns a list that includes, with probability $1 - \psi$, every degree d polynomial p such that f and p agree on δ fraction of the points.

Notice that the goal of Brute-force is what one would expect to be the goal of Find-all-poly. Its weakness will be its running time, which is doubly exponential in n and exponential in d. However, we only invoke it for $n \leq 4$. In this case its running time is of the order of δ^{-d^4}. The description of Brute-force is given in Figure 2.

Lemma 5 Brute-force$(f, \delta, n, d, q, \psi, \phi)$ *satisfies* **(P3)**. *Its runs in time* $O(kl^3(\log \frac{1}{\psi})(\log \frac{1}{\phi}))$ *where* $l = \binom{n+d}{d}$ *and* $k = O((\log \frac{1}{\psi})(\delta - \frac{d}{q})^{-l})$.

Proof: The running time of Brute-force is immediate from its description and the fact that a naive interpolation algorithm for a (multivariate) polynomial with N coefficients runs in time $O(N^3)$. To prove that with probability at least $1 - \psi$, it outputs every polynomial p with δ agreement f, let us fix p and argue that in any one of the k iterations, p is likely to be added to the output list with probability $\zeta = \frac{1}{2(\delta - \frac{d}{q})^l}$. The lemma follows from the fact that the number of iterations is a sufficiently large multiple of $\frac{1}{\zeta}$.

To prove that p is likely to be the candidate in each iteration with probability ζ, we show that with probability 2ζ it is the polynomial interpolated in the iteration. The lemma follows from the fact that Test-valid will return true with probability at least $\frac{1}{2}$.

To show that p is the polynomial returned in the interpolation step, we look at the task of finding p as the task of solving a linear system. Let $\vec{p}$ denote the l dimensional vector corresponding to the coefficients of p. Let M be the $l \times l$ dimensional matrix whose rows correspond to the points $x^{(1)}, \ldots, x^{(l)}$ and whose columns correspond to the monomials in

p. Specifically, the entry $M_{i,j}$, where j corresponds to the monomial $x_1^{d_1} \dots x_n^{d_n}$, is given by $(x_1^{(i)})^{d_1} \dots (x_n^{(i)})^{d_n}$. Finally let $\vec{f}$ be the vector $(f(x^{(1)}), \dots, f(x^{(l)}))$. For p to be the polynomial returned in this step, we must have that M is of full rank and $p(x^{(i)}) = f(x^{(i)})$ for every i. Let $M^{(i)}$ denote the $i \times l$ matrix with the first i rows of M.

Fix $x^{(1)}, \dots, x^{(i-1)}$ such that $p(x^{(j)}) = f(x^{(j)})$ for every $j \in [i-1]$. We argue that the choice of $x^{(i)}$ is such that $p(x^{(i)}) = f(x^{(i)})$ AND the rank of $M^{(i)}$ is greater than that of $M^{(i-1)}$ with probability at least $\delta - \frac{d}{q}$. The lemma follows immediately. It is easy to see that $f(x^{(i)}) = p(x^{(i)})$ with probability at least δ. To complete the proof it suffices to establish that the probability, over a random choice of $x^{(i)}$, that $M^{(i)}$ has the same rank as $M^{(i-1)}$ is at most $\frac{d}{q}$. Consider two polynomials p_1 and p_2 such that $p_1(x^{(j)}) = p_2(x^{(j)})$ for every $j \in [i-1]$. Then for the rank of $M^{(i)}$ to be the same as the rank of $M^{(i-1)}$ it must be that $p_1(x^{(i)}) = p_2(x^{(i)})$ (else the solutions to the ith system are not the same as the solutions to the $i-1$th system). But for distinct polynomials p_1 and p_2 the event $p_1(x^{(i)}) = p_2(x^{(i)})$ happens with probability at most $\frac{d}{q}$ for randomly chosen $x^{(i)}$. This concludes the proof of the lemma. ▌

As an extension of univariate interpolations, we have:

(P4) Compute-coefficients$(p^{(1)}, \dots, p^{(d+1)}, r^{(1)}, \dots, r^{(d+1)}, n, d, q, \psi)$ takes as input $d+1$ polynomials $p^{(j)}$ in $n-1$ variables of degree $d-1$ and $d+1$ values $r^{(j)} \in \mathrm{GF(q)}$ and returns a degree d polynomial $p : \mathrm{GF(q)}^n \rightarrow \mathrm{GF(q)}$ such that $p|_{r^{(j)}} \equiv p^{(j)}$ for every $j \in [d+1]$, if such a polynomial p exists (else may return anything).

Compute-coefficients works as a simple interpolation algorithm: Specifically it finds $d+1$ univariate polynomials $h_1, \dots, h_{d+1}$ such that $h_i(r^{(j)})$ equals 1 if $i = j$ and 0 otherwise and then returns the polynomial $p(x_1, \dots, x_n) = \sum_{j=1}^{d+1} h_j(x_n) \cdot p^{(j)}(x_1, \dots, x_{n-1})$. Note that the computation of the h_i's depends only on the $r^{(j)}$'s.

Proposition 6 Compute-coefficients$(p^{(1)}, \dots, p^{(d+1)}, r^{(1)}, \dots, r^{(d+1)}, n, d, q, \psi)$ *satisfies* **(P4)**. *Its running time is* $O(n^3 + \binom{n+d}{d})$.

3.2 The main routines

As mentioned earlier, the main subroutines are Find-all-poly and Extend, whose inputs and properties are described next. They take, among other inputs, a special parameter α which will be fixed later.

(P5) Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$ returns a list of polynomials containing every polynomial of degree d on n variables that agrees with f in δ fraction of the inputs. Specifically, the output list contains every degree d polynomial p with agreement δ with f, with probability at least $1 - \psi$.

The algorithm is described formally in Figure 3. Informally, the algorithm uses the ("trivial") subroutines for the base cases $n \leq 4$ or $d = 0$, and in the remaining (interesting) cases it iterates a randomized process several times. Each iteration is initiated by a random linear transformation of the coordinates. Then in this new coordinate system, Find-all-poly finds (using the "trivial" subroutine Brute-force) a list of all 4-variate polynomials having significant agreement with the 4-prefix of the oracle. It then extends each polynomial in the list one variable at a time till it finds the n-prefix of the polynomial (which is the polynomial itself). Thus the crucial piece of the work is relegated to the subroutine Extend which is supposed to extend a given $(i-1)$-prefix of a polynomial with significant agreement with f to its i-prefix. The goals of Extend are described next.

(P6) Extend$(f, p, \delta, n, d, q, \psi, \phi, \alpha)$ takes as input a degree d polynomial p in $n-1$ variables and returns a list of degree d polynomials in n variables. The guarantee is that is if p^* is a polynomial with agreement δ with f and further $p^*|_j$ has agreement $\alpha \cdot \delta$ with $f|_j$ for every $j \in \{0, \ldots, d\}$ (implying, in particular, that p equals $p^*|_0$, the $(n-1)$-prefix of p^*), then p^* is in the output list (with probability $1 - \psi$).

Figure 4 describes the algorithm formally. Extend returns all n-variable extensions p^*, of a given $(n-1)$-variable polynomial p, provided p^* agrees with f in a strong sense: p^* has significant agreement with f and for every $j \in \{0, \ldots, d\}$, $p^*|_j$ also has significant agreement with $f|_j$ (the latter agreement may be slightly less than the former). To recover p^*, Extend first invokes Find-all-poly to find the polynomials $p^*|_j$ for $d+1$ values of j. This is feasible only if a polynomial $p^*|_j$ has good agreement with $f|_j$, for every $j \in \{0, \ldots, d\}$. Thus it is crucial that when Extend is called with f and p, all extensions p^*'s with good agreement with f also satisfy the stronger agreement property (above). We will show that the calling program (i.e., Find-all-poly at the higher level of recursion) will, with high probablity, satisfy this property, by virtue of the random linear transformation of coordinates.

All the recursive calls (of Find-all-poly within Extend) always involve a smaller degree parameter, thereby ensuring that the algorithms terminate (quickly). Having found a list of possible values of $p^*|_j$, Extend uses a simple interpolation (subroutine Compute-coefficients) to find a candidate for p^*. It then uses Test-valid to prune out the many invalid polynomials that are generated this way, returning only polynomials that are close to f.

Notice that we do not make any claims (in **(P5)** and **(P6)**) about the probability with which polynomials with low-agreement may be included in the output. We will consider this issue later when analyzing the running times of Find-all-poly and Extend.

We now go on the formal analysis of the correctness of Find-all-poly and Extend.

3.3 Correctness of Find-all-poly and Extend

Lemma 7 *If $\alpha \leq 1 - \frac{1}{q}$, $\delta \geq \frac{d+1}{q}$, and $q \geq 3$ then* Find-all-poly *satisfies* **(P5)** *and* Extend *satisfies* **(P6)**.

Proof: We prove the lemma by a double induction, first on d and for any fixed d, we induct on n.

Assume that Find-all-poly is correct for every $d' < d$ (for every $n' \leq n$ for any such d'.) We use this to establish the correctness of Extend$(f, p, n', d, q, \psi, \alpha)$ for every $n' \leq n$. Fix a polynomial p^* satisfying the hypothesis in **(P6)**. We will prove that p^* is in the output list

```
Find-all-poly(f, δ, n, d, q, ψ, φ, α);
If d = 0 return(Constants(f, δ, n, q, ψ));
If n ≤ 4 return(Brute-force(f, δ, n, d, q, ψ, φ));
L ← {};

Repeat O(log N_{n,d,δ}/ψ) times:
  Pick a random nonsingular n × n matrix R over GF(q)
  Pick a random vector b ∈ GF(q)^n.
  Let g denote the oracle given by g(y) = f(R^{-1}(y − b)).
  L_4 ← Brute-force(g|_{0^{n-4}}, δ, 4, d, q, 1/(10n), φ).

  for i = 5 to n do
    L_i ← {} /* List of (d, i)-prefixes */
    for every polynomial p ∈ L_{i-1} do
      L_i = L_i ∪ Extend(g|_{0^{n-i}}, p, δ, i, d, q, 1/(10n), φ, α)
    endfor
  endfor

  Untransform L_n:   L'_n ← {p'(x) =def p(Rx + b) | p ∈ L_n}.
  L ← L ∪ L'_n.
endRepeat

return(L)
```

Figure 3: Find-all-poly

with probability $1 - \frac{\psi}{N_{n',d,\delta}}$. The correctness of Extend follows from the fact that there are at most $N_{n',d,\delta}$ such polynomials p^* and the probability that there exists one for which the condition is violated is at most ψ.

To see that p^* is part of the output list, notice that for any fixed $j \in \{0, \ldots, d\}$, $p^*|_j - p$ is included in $\mathcal{L}^{(j)}$ with probability $1 - \frac{\psi}{2(d+1)N_{n',d,\delta}}$. This follows from the fact that $p^*|_j - p$ and $f|_j - p$ have agreement at least δ, the fact that $p^*|_j - p = p^*|_j - p^*|_0$ is a degree $d-1$ polynomial[4], and thus, by the inductive hypothesis on the correctness of Find-all-poly, such a polynomial should be in the output list. By the union bound, we have that for every $j \in \{0, \ldots, d\}$ $p^*|_j - p$ is included in $\mathcal{L}^{(j)}$ with probability $1 - \frac{\psi}{2N_{n',d,\alpha\cdot\delta}}$ and in such a case $p^* - p$ will be one of the polynomials returned by an invocation of Compute-coefficients. In

[4]To see that $p^*|_j - p^*|_0$ is a polynomial of total degree at most $d-1$, notice that $p^*(x_1, \ldots, x_n)$ can be expressed uniquely as $r(x_1, \ldots, x_{n-1}) + x_n q(x_1, \ldots, x_n)$, where degree of q is at most $d-1$. Then $p^*|_j - p^*|_0(x_1, \ldots, x_{n-1}) = j \cdot q(x_1, \ldots, x_{n-1}, j)$ is also of degree $d-1$.

```
Extend(f, δ, p, n, d, q, ψ, φ, α).

  L' ← {}.
  L^(0) ← {0̄} (where 0̄ is the constant 0 polynomial).
  for j = 1 to d do
       f^(j) ← f|_j − p.
       L^(j) ←Find-all-poly(f^(j), α·δ, n, d−1, q, ψ/(2N_{n,d,α·δ}(d+1)), φ, α).
  endfor

  for every (d+1)-tuple (p^(0), ..., p^(d)) with p^(k) ∈ L^(k) do
     p' ← Compute-coefficients(p^(0), ..., p^(d), 0, ..., d;  n, d, q).
     if Test-valid(f, p+p', δ, n, d, q, ψ/(2N_{n,d,α·δ}), φ) then
        L' ← L' ∪ {p+p'};
  endfor
  return(L').
```

Figure 4: Extend

such a case p^* will be tested by Test-valid and accepted with probability at least $1 - \frac{\psi}{2N_{n',d,\alpha\cdot\delta}}$. Again summing up all the error probabilities, we have that p^* is in the output list with probability at least $1 - \frac{\psi}{N_{n',d,\alpha\cdot\delta}}$. This concludes the correctness of Extend.

We now move on to the correctness of Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$. Here we will try to establish that for a fixed polynomial p with agreement δ with f, the polynomial p is added to the list $\mathcal{L}$ with constant probability in each iteration of the Repeat loop. Thus the probability that it is not added in any of the iterations is at most $\frac{\psi}{N_{n,d,\delta}}$ and thus the probability that there exists a polynomial that is not added in any iteration is at most ψ. We may assume that $n \geq 5$ and $d \geq 1$ (or else correctness is guaranteed by the trivial subroutines).

Fix a degree d polynomial p with agreement δ with the function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$. We first argue that (R, b) form a "good" linear transformation with constant probability. Recall that from now onwards Find-all-poly works with the oracle $g : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ given by $g(y) = f(R^{-1}(y - b))$. Analogously define $p'(y) = p(R^{-1}(y - b))$, and notice p' is also a polynomial of degree d. For any $i \in \{5, \ldots, n\}$ and $j \in \{0, \ldots, d\}$, we say that (R, b) is *good for* (i, j) if the agreement between $g|_{j,0^{n-i}}$ and $p'|_{j,0^{n-i}}$ is at least $\alpha\delta$. Lemma 8 (below) shows that the probability that (R, b) is good for (i, j) with probability at least $1 - \frac{1}{q^{i-1}} \cdot \left(1 + \frac{1}{\delta(1-\alpha)^2}\right)$. Now call (R, b) *good* if it is good for every pair (i, j), $i \in \{5, \ldots, n\}$ and $j \in \{0, \ldots, d\}$. Summing up the probabilities that R is not good for (i, j) we find that

R is not good with probability at most

$$\begin{aligned}
& (d+1)\left(1+\frac{1}{\delta(1-\alpha)^2}\right)\sum_{i=5}^{n} q^{-i+1} \\
< \;& (d+1)\left(1+\frac{1}{\delta(1-\alpha)^2}\right) q^{-3}\frac{1}{q-1} \\
\leq \;& \frac{d+1}{q^3(q-1)} + \frac{1}{q-1} \quad \text{(Using } \alpha \leq 1-\tfrac{1}{q} \text{ and } \delta \geq \tfrac{d+1}{q}.) \\
\leq \;& \frac{1}{18}+\frac{1}{2} \quad \text{(Using } d+1 \leq q \text{ and } q \geq 3.) \\
< \;& \frac{2}{3}
\end{aligned}$$

Conditioned upon (R, b) being good, we find that $\mathcal{L}_4$ does not contain the 4-prefix of p with probability at most $\frac{1}{10n}$. Inductively, we have that the i-prefix of p is not contained in the list $\mathcal{L}_i$ with probability at most $\frac{i}{10n}$. (There is a probability of at most $\frac{1}{10n}$ that the $(i-1)$-prefix of p is in $\mathcal{L}_{i-1}$ and the i-prefix is not returned by Extend.) Thus with probability at most $\frac{1}{10}$, the polynomial p is not included in $\mathcal{L}_n$ (conditioned upon (R, b) being good). Adding back the probability that (R, b) is not good, we find that with probability at most $\frac{2}{3}+\frac{1}{10} \leq \frac{3}{4}$, the polynomial p is not in $\mathcal{L}_n$ in any single iteration. This concludes the proof of the correctness of Find-all-poly. ▮

3.4 Analysis of the random linear transformation

We now fill in the missing lemma establishing the probability of the "goodness" of a random linear transformation.

Lemma 8 *Let f and g be functions mapping $\mathrm{GF(q)}^n$ to $\mathrm{GF(q)}$ that have δ agreement with each other, and let R be a random non-singular $n \times n$ matrix and b be a random element of $\mathrm{GF(q)}^n$. Then, for every $i \in \{1, \ldots, n\}$ and $j \in \mathrm{GF(q)}$:*

$$\Pr_{R,b}\left[f'|_{j,0^{n-i}} \textit{ and } g'|_{j,0^{n-i}} \textit{ have less than } \alpha\delta \textit{ agreement}\right] \leq \frac{1}{q^{i-1}} \cdot \left(1+\frac{1}{\delta(1-\alpha)^2}\right),$$

where $f'(y) = f(R^{-1}(y-b))$ and $g'(y) = g(R^{-1}(y-b))$.

Proof: Let $G = \{x \in \mathrm{GF(q)}^n | f(x) = g(x)\}$, be the set of "good" points. Observe that $\delta = |G|/q^n$. Let $S_{R,b} = \{x \in \mathrm{GF(q)}^n | Rx + b \text{ ends in } j, 0^{n-i}\}$. Then we wish to show that

$$\Pr_{R,b}\left[\frac{|S_{R,b} \cap G|}{|S_{R,b}|} < \alpha \cdot \frac{|G|}{q^n}\right] \leq \frac{1}{q^{i-1}}\left(1+\frac{1}{\delta(1-\alpha)^2}\right). \tag{3}$$

Observe that the set $S_{R,b}$ can be expressed as the preimage of $(j, 0^{n-i})$ in the map $\pi : \mathrm{GF(q)}^n \to \mathrm{GF(q)}^m$, where $m = n - i + 1$, given by $\pi(x) = R'x + b'$ where R' is the $m \times n$ matrix obtained by taking the bottom m rows of R and b' is the vector obtained by taking the last m elements of b. Note that R' is a uniformly distributed $m \times n$ matrix of full rank over GF(q) and b' is just a uniformly distributed m-dimensional vector over GF(q). We first analyze what happens when one drops the full-rank condition on R'.

Claim 9 *Let R' be a random $m \times n$ matrix over* GF(q) *and b' be a random element of* $\mathrm{GF(q)}^m$. *For some fixed vector $\vec{s} \in \mathrm{GF(q)}^m$ let $S = \{x | R'x + b' = \vec{s}\}$. Then, for any set $G \subseteq \mathrm{GF(q)}^n$,*

$$\Pr_{R',b'}\left[\frac{|S \cap G|}{|S|} < \alpha \cdot \frac{|G|}{q^n}\right] \leq \frac{q^m}{(1-\alpha)^2 |G|}.$$

Proof: For $x \in \mathrm{GF(q)}^n$, let $I(x)$ denote the indicator variable that is 1 if $x \in S$ (i.e., $R'x + b' = \vec{s}$) and 0 otherwise. Then, the expected value of $I(x)$, over the choice of (R', b'), is q^{-m}. Furthermore, the random variables $I(x_1)$ and $I(x_2)$ are independent, for any distinct pair x_1 and x_2. The event whose probability is being estimated in the claim is the event that $\sum_{x \in G} I(x)$ is smaller than α times its expectation. A standard application of Chebychev's inequality yields the desired bound. ▌

To fill the gap caused by the "full rank clause", we use the following claim.

Claim 10 *The probability that a randomly chosen $m \times n$ matrix over* GF(q) *is not of full rank is at most $q^{-(n-m)}$.*

Proof: We can consider the matrix as being chosen one row at a time. The probability that the jth row is dependent on the previous $j-1$ rows is at most q^{j-1}/q^n. Summing up over j going from 1 to m we get that the probability of getting a matrix not of full rank is at most $q^{-(n-m)}$. ▌

Finally we establish (3). Let $E_{R',b'}$ denote the event that $\frac{|S \cap G|}{|S|} < \alpha \frac{|G|}{q^n}$ (recall that $S = S_{R',b'}$) and let $F_{R',b'}$ denote the event that R' is of full row rank.

Then considering the space of uniformly chosen matrices R' and uniformly chosen vectors b' we are interested in the quantity:

$$\begin{aligned} \Pr_{R',b'}[E_{R',b'} | F_{R',b'}] &= \frac{\Pr_{R',b'}[E_{R',b'} \text{ and } F_{R',b'}]}{\Pr_{R',b'}[F_{R',b'}]} \\ &\leq \Pr_{R',b'}[E_{R',b'}] + \Pr_{R',b'}[\neg(F_{R',b'})] \\ &\leq \frac{q^m}{(1-\alpha)^2 |G|} + q^{-(n-m)}. \end{aligned}$$

The lemma follows by substituting $m = n - i + 1$. ▌

3.5 Analysis of the running time of Find-all-poly

Lemma 11 *For integers d_0, n_0 and $\alpha, \delta_0 \in [0,1]$, let $M = \max_{0 \leq d \leq d_0} \{N_{n_0,d,(\alpha^{d_0-d}) \cdot (\delta_0/2)}\}$. Then, the running time of* Find-all-poly$(f, \delta_0, n_0, d_0, q, \psi, \beta, \alpha)$ *is bounded by a polynomial in M^{d_0+1}, $(n_0+d_0)^{d_0}$, $(\frac{1}{\delta_0})^{(d_0+4)^4}$ and $\log \frac{1}{\psi}, \log \frac{1}{\phi}$, with probability $1 - \phi \cdot (n_0^2 (d_0+1)^2 M \log M)^{d_0+1}$.*

Proof: We fix n_0 and d_0. Observe that in all recursive calls to Find-all-poly, δ and d are related by the invariant $\delta = \alpha^{d_0-d}\delta_0$. Thus assuming the algorithms run correctly, they should only return polynomials with agreement at least $\delta/2$ (which motivates the quantity M). Observe further that the parameter ϕ never changes and the parameter ψ only affects the number of iterations of the outermost call to Find-all-poly. In all other calls, this parameter is at least $\psi_0 = \frac{1}{20n_0(d_0+1)M}$. Let T denote an upper bound on the running time of any of the subroutine calls to Compute-coefficients, Brute-force, Constants, Test-valid. Then

$$T \leq \max\left\{\left(n_0 d_0 \binom{n_0+d_0}{d_0}\right), \left(\frac{2}{\alpha^{d_0-d}\delta_0}\right)^{\binom{d_0+4}{4}} \cdot \log \frac{1}{\min\{\psi_0, \phi\}}\right\}.$$

Let $P(d)$ denote an upper bound on the probability that any of the recursive calls made to Find-all-poly by Find-all-poly$(f, \alpha^{d_0-d}\delta_0, n, d, q, \psi, \phi, \alpha)$ returns a list of length greater than M, maximized over f, $1 \leq n \leq n_0$, $\psi \geq \psi_0$. Let $F(d)$ denote an upper bound on the running time on Find-all-poly$(f, \alpha^{d_0-d}\delta_0, n, d, q, \psi, \phi, \alpha)$, conditioned upon the event that no recursive call returns a list of length greater than M. Similarly let $E(d)$ denote an upper bound on the running time of Extend, under the same condition.

We first derive recurrences for P. Notice that the subroutine Constants never returns a list of length greater than $\frac{2}{\alpha^{d_0}\delta_0}$ (every constant output must have a fraction of $\frac{\alpha^{d_0}\delta_0}{2}$ representation in the sampled points). Thus $P(0) = 0$. To bound $P(d)$ in other cases, every iteration of the Repeat loop in Find-all-poly contributes as follows: ϕ from the call to Brute-force; at most $n_0 - 4$ times the probability that Extend returns an invalid polynomial, i.e., a polynomial with agreement less than $\delta_d/2$ with its input function f. The probability that Extend returns such an invalid polynomial is bounded by the sum of $(d+1) \cdot P(d-1)$ (from the recursive calls to Find-all-poly) and $M^{d+1}\phi$ (from the calls to Test-valid). (Notice that to get the final bound we use the fact that we estimate this probability only when previous calls do not produce too long a list.) Finally the number of iterations of the Repeat loop Find-all-poly is at most $\log(30n_0(d_0+1)M^2)$, by the condition of $\psi \geq \psi_0$ and the assumption on M in the lemma statement. We replace this quantity by a gross bound of $n_0(d_0+1)(\log M)$, which is certainly an upper bound for large enough n_0. Thus summing up all the error probabilities and simplifying a little we get:

$$P(d) < (n_0^2(d_0+1)\log M) \cdot \left((d+1)P(d-1) + M^{d+1}\right) \cdot \phi$$

Clearly $P(d) \leq (n_0^2(d_0+1)^2 M \log M)^{d+1}\phi$.

A similar analysis for F and E yields the following recurrences:

$$\begin{aligned} F(0) &\leq T \\ F(d) &\leq n_0^2(d_0+1)(\log M) \cdot E(d) \\ E(d) &\leq (d+1)F(d-1) + M^{d+1}T \end{aligned}$$

Solving the recurrence yields $F(d) \leq (n_0^2(d_0+1)^2 M \log M)^{d+1}T$. This concludes the proof of the lemma. ▮

Lemma 12 *For integers d_0, n_0 and $\alpha, \delta_0 \in [0,1]$, let $M = \max_{0\leq d\leq d_0}\{N_{n_0,d,(\alpha^{d_0-d})\cdot(\delta_0/2)}\}$. If $\alpha \geq 1 - \frac{1}{d_0+1}$ and $\delta_0 \geq 2e\sqrt{\frac{d_0}{q}}$ then $M \leq O(\frac{1}{\delta_0^2})$.*

Proof: We use Part (2) of Theorem 17, which claims that $N_{n,d,\delta} \leq \frac{1}{\delta^2-(d/q)}$, provided $\delta^2 \geq d/q$. Let $\delta_d = \alpha^{d_0-d}\delta_0$. Then $\delta_d/2 \geq (1-\frac{1}{d_0+1})^{d_0+1} \cdot (\delta_0/2) \geq \delta_0/2e \geq \sqrt{d/q}$, by the condition in the lemma. Thus M is at most $\frac{1}{\delta_d^2-(d/q)} \leq \frac{2}{\delta_d^2} = O(\frac{1}{\delta_0^2})$. ▮

Theorem 13 *Given oracle access to a function f and Suppose δ, k, d and q are parameters satisfying $\delta \geq \max\{\frac{d+1}{q}, 2e\sqrt{d/q}\}$ and $q \geq 3$. Let $\alpha = 1 - \frac{1}{d+1}$, $\psi = 2^{-k}$ and $\phi = 2^{-k} \cdot (n(d+1)\frac{1}{\delta_0^2})^{-2(d+1)}$. Then, given oracle access to a function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$, the algorithm* Find-all-poly$(f, \delta, n, d, q, \psi, \phi, \alpha)$ *runs in* $\mathrm{poly}((k \cdot nd/\delta)^{O(d^4)})$*-time and outputs, with probability at least $1 - 2^{-k}$, a list containing all degree d polynomials which agree with f on at least an δ fraction of the inputs. Furthermore, the list does not contain any polynomials which agree with f on less than an $\frac{\delta}{2}$ fraction of the inputs.*

Remarks:

1. Thus, combining Theorems 1 and 13, we get reconstruction algorithms for all $d < q$, provided δ is large enough. Specifically, for the case $q = 2$ and $d = 1$, we invoke Theorem 1.
2. The constant $2e$ in the lower bound on δ can be replaced by $(1+\epsilon)e^{d/q}$, for any $\epsilon > 0$, by recalibrating the subroutine Test-valid and by setting $\alpha = 1 - \frac{1}{q}$.

Proof: The correctness follows from Lemma 7 and running time from Lemmas 11 and 12. ▮

4 Counting: Worst Case

In this section we give a worst-case bound on the number of polynomials which agree with a given function f on δ fraction of the points. In the case of linear polynomials our bound works for any $\delta > \frac{1}{q}$, while in the general case our bound works only for δ that is large enough.

4.1 General derivation

The bounds are derived using a very elementary property of polynomial functions, namely that two of them do not agree on too many points. In fact we state and prove the theorem for any generic "error correcting code" and then specialize the bound to the case of polynomials later. We first recall the standard definition of error-correcting codes. To do so we refer to strings over an alphabet $[q]$. For a string $R \in [q]^N$ (R for recieved word) and $i \in [N]$, we let $R(i)$ denote the ith coordinate of R. The Hamming distance between strings R_1 and R_2, denoted $\Delta(R_1, R_2)$, is the number of coordinates i where $R_1(i) \neq R_2(i)$.

Definition 14 (Error correcting code) *For integers N, K, D and q an $[N, K, D]_q$ code $\mathcal{C}$ is a family of q^K strings from $[q]^N$ such that for any two strings $C_1, C_2 \in \mathcal{C}$, the Hamming distance between them is at least D.*

Theorem 15 *Let $\delta > 0$ and $R \in [q]^N$. Suppose $C_1, \ldots, C_m \in [q]^N$ are distinct codewords from an $[N, K, D]_q$ code that satisfy $\forall j \in \{1, \ldots, m\}$, $\Delta(R, C_j) \leq (1-\delta)N$. Let γ denote the quantity $\frac{N-D}{N}$. Then the following bounds hold:*

1. *If $\delta \geq \sqrt{2 + \frac{\gamma}{4}} \cdot \sqrt{\gamma} - \frac{\gamma}{2}$ then $m < \frac{2}{\delta + \frac{\gamma}{2}}$.*
   (For small γ the above expressions are approximated by $\delta > \sqrt{2\gamma}$ and $m < 2/\delta$, respectively.)
2. *If $\delta \geq \frac{1}{q} + \sqrt{(\gamma - \frac{1}{q})(1 - \frac{1}{q})}$ then $m \leq (1-\gamma)\left(1 - \frac{1}{q}\right) \cdot \frac{1}{(\delta-(1/q))^2 - (1-\frac{1}{q})(\gamma - \frac{1}{q})}$.*
   (For small γ the above expressions are approximated by $\delta > \min\{\sqrt{\gamma}, \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}\}$ and $m < \frac{1-\gamma}{\delta^2 - \gamma} < \frac{1}{\delta^2 - \gamma}$, respectively. For $\gamma = \frac{1}{q}$, the bounds hold for every $\delta \geq \frac{1}{q}$.) [5]

Remark: The above bounds apply in different situations and yield different bounds on m. The first applies for somewhat larger values of δ and yields a stronger bound which is $O(\frac{1}{\delta})$. The second bound applies also for smaller values of δ and yields a bound which grows as $O(\frac{1}{\delta^2})$.

Proof: The bound in (1) is proven by a simple inclusion-exclusion argument. Let $m' \leq m$. We count the number of coordinates $i \in [N]$ that satisfy the property that one of the first m' codewords agree with R on coordinate i. Namely, let $\chi_j(i) = 1$ if $C_j(i) = R(i)$ and $\chi_j(i) = 0$ otherwise. Then, by inclusion-exclusion we get

$$\begin{aligned} N &\geq |\{i : \exists j \ \chi_j(i) = 1\}| \\ &\geq \sum_{j=1}^{m'} \sum_i \chi_j(i) - \sum_{1 \leq j_1 < j_2 \leq m'} \sum_i \chi_{j_1}(i)\chi_{j_2}(i) \\ &\geq m' \cdot \delta N - \binom{m'}{2} \cdot \max_{1 \leq j_1 < j_2 \leq m'} |\{i : C_{j_1}(i) = C_{j_2}(i)\}| \end{aligned}$$

where the last inequality is due to the fact that C_j agrees with R on at least δN coordinates. Since two codewords R_1 and R_2 can agree on at most $N - D$ coordinates, we get:

$$\forall m' \leq m, \qquad m'\delta N - \frac{m'(m'-1)}{2} \cdot (N - D) \leq N. \tag{4}$$

Consider the function $g(y) \stackrel{\text{def}}{=} \frac{\gamma}{2} \cdot y^2 - (\delta + \frac{\gamma}{2}) \cdot y + 1$. Then (4) says that $g(m') \geq 0$, for every integer $m' \leq m$. Let α_1 and α_2 be the roots of g. Suppose that the roots are real and additionally satisfy $|\alpha_1 - \alpha_2| \geq 1$, then $m \leq \min\{\alpha_1, \alpha_2\}$. We now show first that under the condition given on δ, $|\alpha_1 - \alpha_2| \geq 1$. Then we show that $\min\{\alpha_1, \alpha_2\} < \frac{2}{\delta + \frac{\gamma}{2}}$. This will suffice to prove Part (1) of the theorem.

Let $\beta = \gamma/2$. Then $g(y) = \beta m^2 - (\beta + \delta) \cdot m + 1$. The roots, α_1 and α_2 are real, provided that $\zeta \stackrel{\text{def}}{=} (\beta + \epsilon)^2 - 4\beta$ is positive which follows from a stronger requirement (see below).

[5] The bounds given by the "approximations" are always applicable. In case γ is small, they are not much weaker than the more complicated expressions. This note applies to Theorem 17 as well.

Let α_1 be the smaller root. To guarantee $\alpha_2 - \alpha_1 \geq 1$ we require $2 \cdot \frac{\sqrt{\zeta}}{2\beta} \geq 1$ which translates to $\zeta \geq \beta^2$ (and hence $\zeta > 0$ as required above). We need to show that

$$(\beta+\delta)^2 - 4\beta \geq \beta^2$$

which occurs if $\delta \geq \sqrt{\beta^2 + 4\beta} - \beta$. Plugging in the value of β we find that the last inequality is exactly what is guaranteed in the hypothesis of Part (1) of the theorem statement. Thus $|\alpha_1 - \alpha_2| \geq 1$. Lastly, we upper bound the smaller of the two roots, i.e., α_1:

$$\begin{aligned}
\alpha_1 &= \frac{\beta+\delta-\sqrt{(\beta+\delta)^2-4\beta}}{2\beta} \\
&= \frac{\beta+\delta}{2\beta} \cdot \left[1 - \left(1 - \frac{4\beta}{(\beta+\delta)^2}\right)^{1/2}\right] \\
&< \frac{\beta+\delta}{2\beta} \cdot \left[1 - 1 + \frac{4\beta}{(\beta+\delta)^2}\right] \\
&= \frac{2}{\beta+\delta}
\end{aligned}$$

The inequality follows by $\zeta > 0$. Again by plugging in the value of β we get the desired bound.

We now prove Part (2). We first introduce some notation. In what follows we will use the arithmetic of integers modulo q to simplify some of our notation. This arithmetic will be used on the letters of the alphabet, i.e., the set $[q]$. For $j \in \{1, \ldots, m\}$ and $i \in [N]$ let $\Gamma_j(i) = 1$ if $C_j(i) \neq R(i)$ and 0 otherwise. (Notice that $\Gamma_j(i) = 1 - \chi_j(i)$.) For $j \in \{1, \ldots, m\}$, $t \in \{0, \ldots, q-1\}$ and $i \in [N]$ let $\Gamma_j^{(t)}(i) = 1$ if $C_j(i) - R(i) = t (\bmod q)$ and 0 otherwise. Thus $\Gamma_j(i) = 1$ if and only if there exists $t \neq 0$ such that $\Gamma_j^{(t)}(i) = 1$. Let $w_j \equiv |\{i : C_j(i) \neq R(i)\}|$ and let $\overline{w} = \frac{\sum_{j=1}^m w_j}{m}$. The fact that the C_j's are close to R implies that for all j, $w_j \leq (1-\delta) \cdot N$.

Our proof generalizes a proof due to S. Johnson (c.f., MacWilliams and Sloane [31]) for the case $q = 2$. The central quantity used to bound m in the binary case can be generalized in one of the two following ways:

$$S \equiv \sum_{j_1, j_2, i} \Gamma_{j_1}(i) \Gamma_{j_2}(i).$$

$$S' \equiv \sum_{j_1, j_2, i} \sum_{t \neq 0} \Gamma_{j_1}^{(t)}(i) \Gamma_{j_2}^{(t)}(i).$$

The first sums over all j_1, j_2, the number of coordinates for which C_{j_1} and C_{j_2} both differ from R. The second sums over all j_1, j_2, the number of coordinate where C_{j_1} and C_{j_2} agree with each other, but disagree from R by t. (Notice that the two quantities are the same for the case $q = 2$.) While neither one of the two quantities are sufficient for our analysis, their sum provides good bounds.

Lower bound on $S + S'$: The following bound is shown using counting arguments which consider the worst way to place a limited number of differences between the C_j's and R. Let

$N_i = |\{j | C_j(i) \neq R(i)\}| = \sum_j \Gamma_j(i)$ and let $N_i^{(t)} = |\{j | C_j(i) - R(i) = t (\bmod q)\}| = \sum_j \Gamma_j^{(t)}(i)$. Then we can lower bound S as follows:

$$S = \sum_{j_1, j_2, i} \Gamma_{j_1}(i) \Gamma_{j_2}(i) = \sum_i N_i^2 \geq \frac{(m\overline{w})^2}{N}.$$

The last inequality above follows from the fact that subject to the condition $\sum_i N_i = m\overline{w}$, the sum of N_i's squared is minimized when all the N_i's are equal. Similarly, using $\sum_i \sum_{t \neq 0} N_i^{(t)} = m\overline{w}$, we lower bound S' as follows:

$$S' = \sum_{j_1, j_2, i} \sum_{t \neq 0} \Gamma_{j_1}^{(t)}(i) \Gamma_{j_2}^{(t)}(i) = \sum_i \sum_{t \neq 0} (N_i^{(t)})^2 \geq \frac{(m\overline{w})^2}{(q-1)N}.$$

By adding the two lower bounds above we obtain:

$$S + S' \geq \frac{(m\overline{w})^2}{N} + \frac{(m\overline{w})^2}{(q-1)N} = \frac{\frac{q}{q-1} m^2 \overline{w}^2}{N}. \tag{5}$$

Upper bound on $S + S'$: For the upper bound we perform a careful counting argument using the fact that the C_j's are codewords from an error-correcting code. For fixed $j_1, j_2 \in \{1, \ldots, m\}$ and $t_1, t_2 \in [q]$, let

$$M_{t_1 t_2}^{(j_1 j_2)} \equiv |\{i | \Gamma_{j_1}^{(t_1)}(i) = \Gamma_{j_2}^{(t_2)}(i) = 1\}|.$$

For every j_1, j_2, we view these as elements of a $q \times q$ matrix $M^{(j_1 j_2)}$. S and S' can be expressed as sums of some of the elements of the matrices $M^{(j_1 j_2)}$. Summing over the $(q-1) \times (q-1)$ minors of all the matrices we get:

$$S = \sum_{j_1, j_2} \sum_{t_1 \neq 0} \sum_{t_2 \neq 0} M_{t_1 t_2}^{(j_1 j_2)}$$

and summing the diagonal elements of $M^{(j_1 j_2)}$ over all $j_1 j_2$, we get

$$S' = \sum_{j_1 j_2} \sum_{t \neq 0} M_{tt}^{(j_1 j_2)}.$$

We start by upper bounding the internal sum above for fixed pair (j_1, j_2), $j_1 \neq j_2$. Since the C_j's are codewords from an $[N, K, D]_q$ code we have $R_{j_1}(i) = R_{j_2}(i)$ for at most $N - D$ values of i, so

$$\sum_{t \neq 0} M_{tt}^{(j_1 j_2)} \leq N - D - M_{00}^{(j_1 j_2)}.$$

Note that the sum of the values of all elements of $M^{(j_1 j_2)}$ equals N, and $N - w_{j_1}$ (resp. $N - w_{j_2}$) is equal to the sum of the values of the 0^{th} column (resp. row) of $M^{(j_1 j_2)}$. To bound the remaining term in the summation above we use inclusion-exclusion as follows:

$$\begin{aligned}
&\sum_{t_1 \neq 0} \sum_{t_2 \neq 0} M_{t_1 t_2}^{(j_1 j_2)} \\
&= \sum_{t_1} \sum_{t_2} M_{t_1 t_2}^{(j_1 j_2)} - \sum_{t_1} M_{t_1 0}^{(j_1 j_2)} - \sum_{t_2} M_{0 t_2}^{(j_1 j_2)} + M_{00}^{(j_1 j_2)} \\
&= N - (N - w_{j_1}) - (N - w_{j_2}) + M_{00}^{(j_1 j_2)} \\
&= w_{j_1} + w_{j_2} - N + M_{00}^{(j_1 j_2)}.
\end{aligned}$$

Combining the bounds above we have (for $j_1 \neq j_2$)

$$\sum_{t\neq 0} M_{tt}^{(j_1j_2)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)} \le (\gamma N - M_{00}^{(j_1j_2)}) + (w_{j_1} + w_{j_2} - N + M_{00}^{(j_1j_2)}) = w_{j_1} + w_{j_2} - N(1-\gamma).$$

(The key point above is the cancellation of $M_{00}^{(j_1j_2)}$.) Observe that if $j_1 = j_2 = j$, then the quantity $\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(jj)} = \sum_{t\neq 0} M_{tt}^{(jj)} = w_j$.

We now combine the bounds above as follows:

$$\begin{aligned} S + S' &= \sum_j \left(\sum_{t\neq 0} M_{tt}^{(jj)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(jj)}\right) + \sum_{j_1\neq j_2}\left(\sum_{t\neq 0} M_{tt}^{(j_1j_2)} + \sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(j_1j_2)}\right) \\ &\le \sum_j 2w_j + \sum_{j_1\neq j_2}(w_{j_1} + w_{j_2} - N(1-\gamma)) \end{aligned}$$

Simplifying the right hand side above, we get:

$$S + S' \le 2m^2\overline{w} - m(m-1)(1-\gamma)N. \tag{6}$$

Putting it together: Combining (5) and (6), we find that

$$\begin{aligned} m &\le (1-\gamma)\cdot\frac{1}{(\frac{\overline{w}}{N})^2\frac{q}{q-1} + 1 - \gamma - 2\frac{\overline{w}}{N}} \\ &\le (1-\gamma)\cdot\frac{1}{(1-\delta)^2\frac{q}{q-1} + 1 - \gamma - 2(1-\delta)} \quad \text{(Using } \overline{w}/N \le 1-\delta.\text{)} \end{aligned}$$

provided $(1-\delta)^2\frac{q}{q-1}+1-\gamma-2(1-\delta) \ge 0$. Let $g(x) \stackrel{\text{def}}{=} \frac{q}{q-1}x^2-2x+(1-\gamma)$. We need to bound δ so that $g(1-\delta) > 0$. Observe first that $g(x) = \frac{q}{q-1}\left(\frac{q-1}{q} - x\right)^2 + \gamma - \frac{1}{q}$. Thus $g(x) > 0$ if $\frac{q-1}{q} - x > \sqrt{\frac{q-1}{q}\cdot(\gamma - \frac{1}{q})}$. In other words, $g(1-\delta) > 0$ provided, $\delta > \frac{1}{q} + \sqrt{\left(1-\frac{1}{q}\right)\cdot\left(\gamma-\frac{1}{q}\right)}$. In this case the bound obtained on m is $\frac{1-\gamma}{g(1-\delta)}$. This is exactly as claimed in Part (2). We now move on to the approximations claimed there.

The relation $g(1-\delta) > 0$ for $\delta > \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}$ follows immediately from the above and the inequality:

$$\frac{1}{q} + \sqrt{\gamma - \frac{1}{q}} > \frac{1}{q} + \sqrt{(1-\frac{1}{q})\cdot(\gamma - \frac{1}{q})}.$$

Next, we verify that $g(1-\delta) > 0$ for every $\delta > \sqrt{\gamma}$. Let $x = 1-\delta$. Then $1-x = \delta > \sqrt{\gamma}$). In this case we have:

$$\begin{aligned} g(x) &= \left(1 + \frac{1}{q-1}\right)x^2 - 2x + 1 - \gamma \\ &= (1-x)^2 + \frac{1}{q-1}x^2 - \gamma \\ &\ge (1-x)^2 - \gamma \\ &\ge 0 \end{aligned}$$

Thus $g(1-\delta) > 0$ provided $\delta > \min\{\sqrt{\gamma}, \frac{1}{q} + \sqrt{\gamma - \frac{1}{q}}\}$. We now derive the claimed upper bounds on m. Setting $x = 1-\delta$, and using $g(x) \geq (1-x)^2 - \gamma$, we get $g(1-\delta) \geq \delta^2 - \gamma$. Thus $m \leq \frac{1-\gamma}{g(1-\delta)} \leq \frac{1-\gamma}{\delta^2-\gamma} < \frac{1}{\delta^2-\gamma}$.

Finally, for $\gamma = 1/q$ we have $g(x) = \frac{q}{q-1} \cdot \left(x - \frac{q-1}{q}\right)^2 > 0$ for all $x \neq 1 - \frac{1}{q}$. Thus, in this case, the bound holds for all $\delta > 1/q$. ▮

4.2 The special case of polynomials

Notice that a function $f : \mathrm{GF(q)}^n \to \mathrm{GF(q)}$ may be viewed as a string of length q^n with letters from the set $[q]$. Viewed in this way we get the following construction of a code using multivariate polynomials. These codes are known as Reed-Muller codes in the coding theory literature.

Proposition 16 *The collection of degree d polynomials in n variables over* GF(q) *form an* $[N, K, D]_q$ *code, for* $N = q^n$, $K = q^{\binom{n+d}{d}}$ *and* $D = (q-d)q^{n-1}$.

Proof: The parameters N and K follow by definition. The distance D is equivalent to the well-known fact [10, 38, 44] that two degree d (multivariate) polynomials over GF(q) may agree in at most d/q fraction of the inputs. ▮

Combining Theorem 15 with Proposition 16 (and using $\gamma = \frac{d}{q}$ in the theorem), we get the following upper bound on the number of polynomials with δ agreement with an arbitrary function.

Theorem 17 *Let $\delta > 0$ and $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$. Suppose $p_1, \ldots, p_m : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ are distinct degree d polynomials that satisfy $\forall j \in \{1, \ldots, m\}$, $\mathrm{Pr}_{x \in \mathrm{GF}(q)^n}[f(x) = p_i(x)] \geq \delta$. Then the following bounds hold:*

1. *If $\delta \geq \sqrt{2 + \frac{d}{4q}} \cdot \sqrt{\frac{d}{q}} - \frac{d}{2q}$ then $m < \frac{2}{\delta + (\frac{d}{2q})}$.*

   (For small $\frac{d}{q}$ the above expressions are approximated by $\delta > \sqrt{2d/q}$ and $m < 2/\delta$, respectively.)

2. *If $\delta \geq \frac{1+\sqrt{(d-1)(q-1)}}{q}$ then $m \leq \frac{(q-d)(q-1)}{q^2} \cdot \frac{1}{\left(\delta - \frac{1}{q}\right)^2 - \frac{(q-1)(d-1)}{q^2}}$.*

   (For small $\frac{d}{q}$ the above expressions are approximated by $\delta > \min\{\sqrt{d/q}, \frac{1}{q} + \sqrt{\frac{d-1}{q}}\}$ and $m < \frac{1}{\delta^2 - (d/q)}$, respectively.)

For the special case of linear polynomials, the bound in Part (2) above is always applicable. This is emphasized in the following theorem.

Theorem 18 *Let $\epsilon > 0$. For a function $f : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$, if $p_1, \ldots, p_m : \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ are distinct linear functions which satisfy*

$$\forall j \in \{1, \ldots, m\}, \quad \Pr_{x \in \mathrm{GF}(q)^n}[f(x) = p_i(x)] \geq \frac{1}{q} + \epsilon,$$

$$\textit{then } m \leq \left(1 - \frac{1}{q}\right)^2 \cdot \frac{1}{\epsilon^2}$$

4.3 Tightness of our bounds

We show that several aspects of the bounds presented above are tight. We start with the observation that Theorem 15 can not be extended further (i.e., to smaller δ). without (possibly) relying on some special properties of the code.

Proposition 19 *Let δ_0, γ_0 satisfy the identity*

$$\delta_0 = \frac{1}{q} + \sqrt{(\gamma_0 - \frac{1}{q})(1 - \frac{1}{q})}. \tag{7}$$

Then for any $\epsilon > 0$, and for sufficiently large N, there exists an $[N, K, D]_q$ code $\mathcal{C}$, with $\frac{N-D}{N} \leq \gamma_0 + \epsilon$ and a word $R \in [q]^N$ and $M \geq 2^{\Omega(\epsilon^2 N)}$ codewords $C_1, \ldots, C_M$ from $\mathcal{C}$ such that $\Delta(R, C_j) \leq (1 - (\delta_0 - \epsilon))N$ for every $j \in [M]$.

Remark: The proposition above should be compared against Part (2) of Theorem 15. Note that Theorem 15 says that for δ_0 and γ_0 satisfying (7) and any $[N, K, D]_q$ code with $\frac{N-D}{N} \geq \gamma_0$, there exist at most $O(\frac{1}{\delta_0^2})$ codewords at distance at most $(1-\delta)N$ from any string of length N, while the proposition shows that if δ_0 is reduced slightly and γ_0 increased slightly, then there could be exponentially many codes at this distance.

Proof: The bound is proven by a standard probabilistic argument. $\mathcal{C}$ will consist only of codewords $C_1, \ldots, C_M$. The codewords C_j's are chosen randomly and independently by the following process. Let $p \in [0, 1]$ — this will be determined shortly.

For every codeword C_j, each coordinate is chosen independently as follows: With probability p it is set to be 1, and with probability $1 - p$ it is chosen uniformly from $\{2, \ldots, q\}$. The string R is simply 1^N.

Observe that for any fixed j, the expected number of coordinates where R and C_j agree is pN. Thus with probability at most $2^{-\Omega(\epsilon^2 N)}$, the agreement between R and C_j is less than $(p - \epsilon)N$. It is possible to set $M = 2^{\Omega(\epsilon^2 N)}$ so that the probability that there exists such a word C_j is less than $\frac{1}{2}$.

Similarly the expected agreement between C_i and C_j is $(p^2 + \frac{(1-p)^2}{q-1})N$. Thus the probability that the distance between a fixed pair is ϵN larger than this number is at most $2^{-\Omega(\epsilon^2 N)}$. Again it is possible to set $M = 2^{\Omega(\epsilon^2 N)}$ such that the probability that such a pair C_i and C_j exists is less than $\frac{1}{2}$.

Thus there is a positive probability that the construction yields an $[N, \Omega(\frac{\epsilon^2 N}{\log q}), D]_q$ code with $\frac{N-D}{N} = p^2 + \frac{(1-p)^2}{q-1} + \epsilon$, so that all codewords are within a distance of $(1 - (p - \epsilon))N$ of the word R. Thus setting $\delta_0 = p$ and $\gamma_0 = p^2 + \frac{(1-p)^2}{q-1}$ would yield the proposition, once it is verified that this setting satisfies (7). This is easily verified by the following algebraic manipulations, starting with our setting of δ_0 and γ_0.

$$\gamma_0 = \delta_0^2 + \frac{(1-\delta_0)^2}{q-1}$$

$$\begin{aligned}
&\Leftrightarrow \quad \frac{q}{q-1}\delta_0^2 - \frac{2}{q-1}\delta_0 + \frac{1}{q-1} - \gamma_0 = 0 \\
&\Leftrightarrow \quad \delta_0^2 - \frac{2}{q}\delta_0 = \frac{q-1}{q}\gamma_0 - \frac{1}{q} \\
&\Leftrightarrow \quad \left(\delta_0 - \frac{1}{q}\right)^2 = \frac{1}{q^2} + \left(\frac{q-1}{q}\gamma_0 - \frac{1}{q}\right) = \left(\gamma_0 - \frac{1}{q}\right)\left(1 - \frac{1}{q}\right) \\
&\Leftrightarrow \quad \delta_0 = \frac{1}{q} + \sqrt{(\gamma_0 - \frac{1}{q})(1 - \frac{1}{q})}
\end{aligned}$$

This concludes the proof. ▮

Next we move on to the tightness of functions described by polynomials. We show that Theorem 18 (i.e., the linear case) is tight for $\delta = O(1/q)$, whereas Part (1) of Theorem 17 is tight for $\delta = \Theta(1/\sqrt{q})$ and $d = 1$. The results below show that for a given value of δ that meets the conditions of the appropriate theorem, the value of m can not be made much smaller.

Proposition 20 *Given a prime p, and an integer k satisfying $1 < k \leq p/3$, let $\delta = k/p$. Then, there exists a function $f : \mathrm{GF}(p) \to \mathrm{GF}(p)$ and at least $m \stackrel{\text{def}}{=} \frac{1}{5(k-1)\delta^2}$ linear functions $f_1, \ldots, f_m : \mathrm{GF}(p) \to \mathrm{GF}(p)$ such that for all $i \in \{1, \ldots, m\}$, $|\{x | f_i(x) = f(x)\}| \geq \delta p$.*

Proof: We start by constructing a relation $R \subset \mathrm{GF}(p) \times \mathrm{GF}(p)$ such that $|R| \leq p$ and there exist many linear functions $g_1, \ldots, g_m$ such that $|R \cap \{(x, g_i(x)) | x \in \mathrm{GF}(p)\}| \geq k$ for all i. Later we show how to transform R and the g_i's so that R becomes a function which still agrees with each g_i on k inputs.

Let $l = \lfloor p/k \rfloor$. Notice $l \approx \frac{1}{\delta}$, where $\delta = k/p$. The relation R simply consists of the pairs in the square $\{(i, j) | 0 \leq i < k, 0 \leq j < l\}$. Let $\mathcal{G}$ be the set of all linear functions which agree with R in at least k places. We now show that $\mathcal{G}$ has size at least $l^2/2(k-1)$. Given non-negative integers a, b, s.t. $a(k-1)+b < l$, consider the linear function $g_{a,b}(x) = ax+b (\bmod p)$. Then, $g_{a,b}(i) \in \{0, \ldots, l-1\}$ for $i \in \{0, \ldots, k-1\}$. Thus, $g_{a,b}(i)$ intersects R in k places. Lastly we observe that there are at least $l^2/2(k-1)$ distinct pairs (a, b) s.t. $a(k-1) + b < l$. Fix $a < l$. Then there are at least $l - (k-1)a - 1$ possible values for b, so that the total number of pairs becomes

$$\begin{aligned}
\sum_{a=0}^{\frac{l-1}{k-1}} l - (k-1)a - 1 \quad &= \quad \frac{l-1}{k-1} \cdot l - (k-1) \cdot \binom{\frac{l-1}{k-1}}{2} - \left(\frac{l-1}{k-1} + 1\right) \\
&> \quad \frac{l^2}{2(k-1)} \qquad \text{(Using } l \geq 3 \text{ since } k \leq p/3.)
\end{aligned}$$

Now we convert the relation R into a function in two stages. First we *stretch* the relation by a factor of l to get a new relation R'. Explicitly, $R' = \{(li, j) | (i, j) \in R\}$. Given $g(x) = ax + b \in \mathcal{G}$, let $g'(x) = (a \cdot l^{-1})x + b$, where l^{-1} is the multiplicative inverse of $l (\bmod p)$. If $g(i) = j$, then $g'(li) = j$. Thus if $(i, g(i)) \in R$, then $(li, g'(li)) \in R'$. Thus if we use $\mathcal{G}'$ to denote the set of linear functions which agree with R' in k places, then $g' \in \mathcal{G}'$

if $g \in \mathcal{G}$. Moreover the map from g to g' is one-to-one, implying $|\mathcal{G}'| \geq |\mathcal{G}|$. (Actually the argument above extends to show that $|\mathcal{G}'| = |\mathcal{G}|$.)

Last we introduce a slope to R', so that it becomes a function. Explicitly $R'' = \{(li + j, j) | (i, j) \in R\}$. Notice that for any pair $(i_1, j_1), (i_1, j_2) \in R''$, we have $i_1 \neq i_2$ implying that R'' can be extended to a function $f : \mathrm{GF}(p) \rightarrow \mathrm{GF}(p)$, which satisfies if $(i, j) \in R''$ then $j = f(i)$. Now for every function $g'(x) = a'x + b' \in \mathcal{G}'$, consider the function $g''(x) = a''x + b''$ where $a'' = a'/(1 + a')$ and $b'' = b'/(1 + a')$. Observe that if $g'(x) = y$, then $g''(x + y) = y$. Thus if g' agrees with R' in k places, then g'' agrees with R'' and hence f in at least k places. Again, if we use $\mathcal{G}''$ to denote the set of linear functions which agree with f in k places, then $|\mathcal{G}''| \geq |\mathcal{G}'|$.

Thus $f : \mathrm{GF}(p) \rightarrow \mathrm{GF}(p)$ has at least $l^2/2(k-1)$ linear functions agreeing with it in k places. Expressing k as δp, we have $l > \frac{1}{(1/p)+\delta}$, and the proposition follows (using $(1/p) + \delta < \frac{3}{2} \cdot \delta$ and $2 \cdot \left(\frac{3}{2}\right)^2 < 5$). ▌

Finally we note that the bounds in Theorem 17 always require δ to be larger than d/q. Such a threshold is also necessary, or else there can be exponentially many degree d polynomials close to the given function. This is shown in the following proposition, which actually asserts the existence of many such polynomials.

Proposition 21 *Let q be a prime-power, $d < q$ and $\delta = \frac{d}{q} - \frac{d-1}{q^2}$. Then, for every n-variate degree d polynomial, f, over $\mathrm{GF}(q)$, there are at least q^{n-1} degree d polynomials which agree with f on at least a δ fraction of the inputs.*

Proof: It suffices to consider the all-zero function, denoted by f. Consider the family of polynomials having the form $\prod_{i=1}^{d-1}(x_1 - i) \cdot \sum_{i=2}^{n} c_i x_i$, where $c_2, ..., c_n \in \mathrm{GF}(q)$. Clearly, each member of this family is a degree d polynomial and the family contains q^{n-1} different polynomials. Now, each polynomial in the family is zero on inputs $(a_1, ..., a_n)$ satisfying either $a_1 \in \{1, ..., (d-1)\}$ or $\sum_{i=2}^{n} c_i a_i = 0$. Since at least a $\frac{d-1}{q} + (1 - \frac{d-1}{q}) \cdot \frac{1}{q}$ fraction of the inputs satisfy this condition, the proposition follows. ▌

5 Counting: Random Case

In this section we present a bound on the number of polynomials which can agree with a function f if f is chosen to look like a polynomial p on some domain D and random on other points.

Theorem 22 *Let $\delta \geq \frac{2(d+1)}{q}$. Suppose that D is an arbitrary subset of density δ in $GF(q)^n$ and $p(x_1, ..., x_n)$ is a degree d polynomial. Consider a function f selected as follows:*

1. *f agrees with p on D;*
2. *the value of f on each of the remaining points in $GF(q)^n - D$ is uniformly and independently chosen.*

Then, with probability at least $1 - \exp\{(n^d \log_2 q) - (\delta/2)^2 q^{n-3}\}$, the polynomial p is the only degree d polynomial which agrees with f on at least an $\delta/2$ fraction of the inputs. Thus for *functions constructed in this manner the output of our reconstruction algorithm will be a single polynomial — namely, p.*

Proof: We use the fact that for two polynomials $p_1 \neq p_2$ in $GF(q)^n$, $p_1(x) = p_2(x)$ on at most d/q fraction of the points in $GF(q)^n$ [10, 38, 44]. Thus, except for p, no other degree d polynomial can agree with f on more than $\frac{d}{q} \cdot q^n$ points in D. The probability that any polynomial p' agrees with f on more than a $\frac{1}{q}+\epsilon$ fraction of the points in which f is randomly chosen is at most $\exp\{-\epsilon^2 q^{n-1}\}$. Furthermore, in order to agree with f on more than an $\delta/2$ fraction of all points, p' must agree with f on at least $(\delta/2 - \frac{d}{q}) \cdot q^n$ of the randomly selected points and so we can use $\epsilon \geq \frac{(\delta/2)-(d/q)}{1-\delta/2} - \frac{1}{q} \geq \frac{\delta}{2} - \frac{d+1}{q} + \frac{\delta}{2q} > \frac{\delta}{2q}$. Thus, the probability that there exists a degree d n-variate polynomial, other than p, that agrees with f on at least an $\delta/2$ fraction of all points is at most $q^{n^d} \cdot \exp\{-\left(\frac{\delta}{2q}\right)^2 q^{n-1}\}$ and the theorem follows. ∎

6 Hardness Results

In this section we give evidence that the (explicit or implicit) reconstruction problem may be hard for some choices of d and the agreement parameter δ, even in the case when $n = 1$. We warn the reader that the problem shown to be hard does differ in some very significant ways from the reconstruction problems considered earlier. In particular, the problems will consider functions and relations defined on some finite subset of a large field, either the field of rational numbers or a sufficiently large field of prime order, where the prime is specified in binary. The hardness results use the "large" field size crucially.

Furthermore, the threshold for which the problem is shown is very small. For example, the hardness results of Section 6.2, defines a function $f : H_1 \times H_2 \to F$, where F is a large field and H_1, H_2 are small subsets of F. In such a hardness result, one should compare the threshold δ of agreement that is required, against $\frac{d}{\max\{|H_1|,|H_2|\}}$, since it the latter ratio that determines the "distance" between two polynomials on this subset of the inputs. Our hardness results typically hold for $\delta \approx \frac{d+2}{\max\{|H_1|,|H_2|\}}$.

6.1 NP-hardness for a variant of the univariate reconstruction problem

We define the following interpolation problem PolyAgree:

Input: Integers d, k, m, and a set of pairs $P = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ such that $\forall i \in [m]$, $x_i \in F$, $y_i \in F$, where F is either the field of rationals or a prime field given by its size in binary.

Question: Does there exist a degree d polynomial $p : F^n \to F$ for which $p(x_i) = y_i$ for at least k different i's?

When F is the field of rational numbers, then the input elements are assumed to be given as a ratio of two N-bit integers and then the input size is measured in terms of the total bit length of all inputs.

Theorem 23 *PolyAgree is NP-hard.*

Remark: This result should be contrasted with the results of [40, 19]. [40, 19] show that PolyAgree is easy provided $k \geq \sqrt{dm}$, while our result shows it is hard without this condition. In particular, the proof uses $m = 2d + 3$ and $k = d + 2$.

Proof: We present the proof for the case of the field of rational numbers only. It is easy to verify that the proof also holds if the field F has prime order that is sufficiently large.

We reduce from subset sum: Given integers $B, a_1, \ldots, a_\ell$, does there exist a subset of the a_i's that sum to B (without loss of generality, $a_i \neq 0$ for all i).

In our reduction we use the fact that degree d polynomials satisfy certain interpolation identities. In particular, let $\alpha_i = (-1)^{i+1}\binom{d+1}{i}$ for $1 \leq i \leq d+1$ and $\alpha_0 = -1$. Then $\sum_{i=0}^{d+1} \alpha_i f(i) = 0$ if and only if $(0, f(0)), (1, f(1)), \ldots, (d+1, f(d+1))$ lies on a degree d univariate polynomial.

We construct the following instance of PolyAgree. Set $d = l-1$, $m = 2d+3$ and $k = d+2$. Next, set $x_i \leftarrow i$, $x_{d+1+i} \leftarrow i$, $y_i \leftarrow a_i/\alpha_i$, and $y_{d+1+i} \leftarrow 0$ for $1 \leq i \leq d+1$. Finally, set $x_{2d+3} \leftarrow 0$ and $y_{2d+3} \leftarrow B$.

No polynomial can pass through both $(x_i, y_i) = (i, a_i/\alpha_i)$ and $(x_{d+1+i}, y_{d+1+i}) = (i, 0)$ for any i, since $a_i \neq 0$. We show that there is a polynomial of degree d that passes through $(0, B)$ and one of either $(i, 0)$ or $(i, a_i/\alpha_i)$ for each $1 \leq i \leq d+1$ if and only if there is a subset of $a_1, \ldots, a_{d+1}$ whose sum is B.

Assume that there is a polynomial p of degree d that passes through $(0, B)$ and one of $(i, 0)$ and $(i, a_i/\alpha_i)$ for each $1 \leq i \leq d+1$. Let S denote the set of indices for which $p(i) = a_i/\alpha_i$ (and $p(i) = 0$ for $i \in [d+1]\backslash S$). Then

$$0 = \sum_{i=0}^{d+1} \alpha_i p(i) = \alpha_0 \cdot B + \sum_{i \in S} \alpha_i \cdot \frac{a_i}{\alpha_i} = -B + \sum_{i \in S} a_i$$

Similarly, if there is set of indices S such that $\sum_{i \in S} a_i = B$, then we define f so that $f(0) = B$, $f(i) = a_i/\alpha_i$ for $i \in S$ and $f(i) = 0$ for $i \in [d+1]\backslash S$. Observing that $\sum_{i=0}^{d+1} \alpha_i f(i) = 0$ it follows that there is a degree d polynomial which agrees with f on $i = 0, ..., d+1$. ∎

6.2 NP-hardness of the reconstruction problem for $n \geq 2$

In the above problem, we did not require that the x_i's be distinct. Thus this result does not directly relate to the black box model used in this paper. The following result applies to our black box model for n-variate functions, for any $n \geq 2$.

We define a multivariate version of PolyAgree requires that the x_i's be distinct. We actually define a parametrized family $\mathsf{FunctionalPolyAgree}_n$, for any $n \geq 1$.

Input: Integer d, a field F, a finite subset $H \subseteq F^n$. a real number δ, and a function $f : H \to F$, given as a table of values.

Question: Does there exist a degree d polynomial $p : F^n \to F$ for which $p(x) = f(x)$ for at least δ fraction of the x's from H?

Theorem 24 *For every $n \geq 2$, FunctionalPolyAgree$_n$ is NP-hard.*

Proof: We prove the theorem for $n = 2$. The other cases follow by simply making an instance where the function depends only on the first two variables and is independent of the remaining $n - 2$ variables.

The proof of this theorem builds on the previous proof. As above we reduce from subset sum. Given an instance $B, a_1, \ldots, a_l$ of the subset sum problem, we set $d = l - 1$ and $k = 2(d+1)$ and F to be the field of rationals. (We could also work over any prime field GF(p), provided $p \geq \sum_{i=1}^{n} a_i$.) Let $\delta = \frac{d+3}{2(d+2)}$. We set $H_1 = \{0, \ldots, d+1\}$, $H_2 = [2k]$. and let $H = H_1 \times H_2$. For $i \in H_1$ we let $\binom{\alpha_i = (-1)^{i+1}(d+1}{i)}$ as before. For $i \in H_1 - \{0\}$, let $y_i = a_i/\alpha_i$ as before. The function f is defined as follows:

$$f(i,j) = \begin{cases} B & \text{if } i = 0 \\ y_i & \text{if } i \in H_1 - \{0\} \text{ and } j \in [k] \\ 0 & \text{otherwise (i.e., if } ii \in H_1 - \{0\} \text{ and } j \in \{k+1, \ldots, 2k\} \end{cases}$$

This completes the specification of the instance of the FunctionalPolyAgree$_2$ problem. We now argue that if the subset sum instance is satisfiable then there exists a polynomial p with agreement δ (on inputs from H) with f. Let $S \in [l]$ be a subset such that $\sum_{i \in S} a_i = B$. Then the function

$$p(i,j) \stackrel{\text{def}}{=} p'(i) \stackrel{\text{def}}{=} \begin{cases} B & \text{if } i = 0 \\ y_i & \text{if } i \in S \\ 0 & \text{if } i \in H_1 \setminus S \end{cases}$$

is a polynomial in i of degree d (since $\sum_{i=0}^{d+1} \alpha_i p'(i) = -B + \sum_{i \in S} a_i = 0$). Furthermore, p and f agree in $2k + k(d+1)$ inputs from H. In particular $p(0,j) = f(0,j) = B$ for every $j \in [2k]$, $p(i,j) = f(i,j) = y_i$ if $i \in S$ and $j \in [k]$ and $p(i,j) = f(i,j) = 0$ if $i \notin S$ and $j \in \{k+1, \ldots, 2k\}$. Thus p and f agree on a fraction $\frac{2k+k(d+1)}{2(d+2)k} = \frac{d+3}{2(d+2)} = \delta$ of the inputs from H, as required.

We now argue that if the reduction leads to a satisfiable instance of the FunctionalPolyAgree$_2$ problem then the subset sum instance is satisfiable. Fix a polynomial p that has agreement δ with f; i.e., $p(i,j) = f(i,j)$ for at least $2k + k(d+1)$ inputs from H. We argue first that in such a case $p(i,j) = p'(i)$ for some polynomial $p'(i)$ and then the proof will be similar to that of Theorem 23. The following claim is crucial in this proof.

Claim 25 *For any $i \in [d+1]$, if $|\{j | p(i,j) = f(i,j)\}| \geq k$, then there exists $c_i \in \{0, y_i\}$ s.t. $p(i,j) = c_i$ for every $j \in [2k]$.*

Proof: Consider the function $p^{(i)}(j) \stackrel{\text{def}}{=} p(i,j)$. $p^{(i)}$ is a degree d polynomial in j. By hypothesis (and the definition of $f(i,j)$) we have, $p^{(i)}(j) \in \{0, y_i\}$ for k values of $j \in [2k]$. Hence $p^{(i)}(j) = 0$ for $k/2$ values of j or $p^{(i)}(j) = y_i$ for $k/2$ values of j. In either case we have that $p^{(i)}$, a degree d polynomial, equals a constant polynomial for $k/2 = d+1$ points implying that $p^{(i)}$ is a constant. That $p^{(i)}(j) = c_i \in \{0, y_i\}$ follows from the hypothesis and definition of f. ▮

From the claim above it follows immediately that for any $i \in [d+1]$, $|\{j | f(i,j) = p(i,j)\}| \leq k$. Now using the fact that f and p agree on $2k + k(d+1)$ inputs it follows that for every $i \in [d+1]$, $f(i,j) = p(i,j)$ for exactly k values of j; and $f(0,j) = p(0,j) = B$

for all values of j. Using the above claim again we conclude that we can define a function $p'(i) \stackrel{\text{def}}{=} c_i \in \{0, y_i\}$ if $i \in [d+1]$ and $p'(0) = B$ such that $p(i,j) = p'(i)$ for every $(i,j) \in H$. Furthermore $p'(i)$ is a degree d polynomial, since p is a degree d polynomial; and hence $\sum_{i=0}^{d+1} \alpha_i p'(i) = 0$. Letting $S = \{i \in [d+1] | y_i \neq 0\}$, we get $-B + \sum_{i \in S} \alpha_i y_i = 0$ which in turns implies $B = \sum_{i \in S} a_i$. Thus the instance of the subset sum problem is satisfiable. This concludes the proof. ▮

Acknowledgments

We thank Mike Kearns, Eyal Kushilevitz, Yishay Mansour, Dana Ron and Salil Vadhan for helpful discussions and comments.

References

[1] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[2] D. Angluin and M. Kriķis. Learning with malicious membership queries and exceptions. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 57–66, New Brunswick, New Jersey, 12-15 July 1994. ACM Press.

[3] S. Ar, R. Lipton, R. Rubinfeld, M. Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing* 28(2): 487–510, April 1999.

[4] S. Arora and M. Sudan. Improved low degree testing and its applications. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 485–495, El Paso, Texas, 4-6 May 1997.

[5] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of Lecture Notes in Computer Science, pages 37–48, Rouen, France, 22-24 February 1990. Springer.

[6] A. Blum and P. Chalasani. Learning switching concepts. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 231-242, Pittsburgh, Pennsylvania, 27-29 July 1992. ACM Press.

[7] A. Blum, M. Furst, M. Kearns, R. Lipton. Cryptographic primitives based on hard learning problems. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of Lecture Notes in Computer Science, pages 278-291, 22-26 August 1993. Springer-Verlag.

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

[9] R. Cleve, M. Luby. A Note on Self-Testing/Correcting Methods for Trigonometric Functions. *International Computer Science Institute Technical Report*, TR-90-032, July, 1990.

[10] R. DeMillo and R. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, June 1978.

[11] P. Elias. List Decoding for noisy channels. *Technical Report 335*, Research Laboratory of Electronics, MIT, 1957.

[12] Y. Freund, D. Ron. Learning to model sequences generated by switching distributions. *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, pages 41-50, Santa Cruz, California, 5-8 July 1995. ACM Press.

[13] S. A. Goldman, M. J. Kearns, and R. E. Schapire. Exact identification of read-once formulas using fixed points of amplification functions. *SIAM Journal on Computing*, 22(4): 705–726, August 1993.

[14] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-testing/correcting for polynomials and for approximate functions. *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 32–42, New Orleans, Louisiana, 6-8 May 1991.

[15] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4): 169–174, September 1992.

[16] O. Goldreich. Foundations of Cryptography – Fragments of a Book. Weizmann Institute of Science, February 1995. Available from the ECCC, `http://www.eccc.uni-trier.de/eccc/`.

[17] O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25-32, Seattle, Washington, 15-17 May 1989.

[18] O. Goldreich, R. Rubinfeld and M. Sudan. Learning polynomials with queries: The highly noisy case. *36th Annual Symposium on Foundations of Computer Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 October 1995. IEEE.

[19] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. To appear *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, Palo Alto, California, 8-11 November, 1998. IEEE.

[20] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 392-401, San Diego, California, 16-18 May 1993.

[21] M. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4): 807–837, August 1993.

[22] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire and L. Sellie. On the learnability of discrete distributions (extended abstract). *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 273-282, Montreal, Quebec, Canada, 23-25 May 1994.

[23] M. Kearns, R. Schapire and L. Sellie, Towards efficient agnostic learning. *COLT*, 1992. Toward efficient agnostic learning (extended abstract). *Proceedings of the Fifth Annual*

ACM Workshop on Computational Learning Theory, pages 341-352, Pittsburgh, Pennsylvania, 27-29 July 1992. ACM Press.

[24] P. Koiran. Efficient learning of continuous neural networks. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 348-355, New Brunswick, New Jersey, 12-15 July 1994. ACM Press.

[25] E. Kushilevitz, Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331-1348, December 1993.

[26] P. Laird. *Learning From Good Data and Bad.* Kluwer Academic Publishers, Boston, 1988.

[27] L.A. Levin, Randomness and non-determinism. *J. of Symb. Logic*, 58(3):1102-1103, 1993. Also in *International Congress of Mathematicians: Abstracts of Invited Lectures.* p.155, Zurich, August 4, 1994.

[28] R. Lipton. New directions in testing. *Distributed Computing and Cryptography*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 191–202, v. 2, 1991.

[29] W. Maass. Efficient agnostic PAC-learning with simple hypotheses. *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 67-75, New Brunswick, New Jersey, 12-15 July 1994. ACM Press.

[30] W. Maass. Agnostic PAC-learning of functions on analog neural networks. *Proceedings of the 7th IEEE Conference on Neural Information Processing Systems.*

[31] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1981.

[32] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM Journal on Computing*, 24(2): 357–368, April 1995.

[33] D. Ron and R. Rubinfeld. Learning fallible finite state automata. *Machine Learning* 27, pp.69-96 (1997).

[34] R. Rubinfeld. On the robustness of functional equations. *35th Annual Symposium on Foundations of Computer Science*, pages 288-299, Santa Fe, New Mexico, 20-22 November 1994. IEEE. To appear in *SIAM Journal on Computing.*

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

[36] Y. Sakakibara. On learning from queries and counterexamples in the presence of noise. *Information Processing Letters*, 37(5):279–284, March 1991.

[37] Y. Sakakibara and Rani Siromoney. A noise model on learning sets of strings. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 295-302, Pittsburgh, Pennsylvania, 27-29 July 1992. ACM Press.

[38] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[39] R. H. Sloan. Types of noise in data for concept learning (extended abstract). *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 91–96, MIT, 3-5 August 1988. ACM Press.

[40] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1): 180–193, March 1997.

[41] L. G. Valiant. Learning disjunctions of conjunctions. *Proceedings of the 9th International Joint Conference on Artificial Intelligence* (pp. 560–566), 1985.

[42] H. Wasserman. Reconstructing randomly sampled multivariate polynomials from highly noisy data. *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 59–67, San Francisco, California, 25-27 January 1998.

[43] L. Welch and E. Berlekamp. Error correction of algebraic block codes. *US Patent*, Number 4,633,470, 1986.

[44] R. Zippel. Probabilistic algorithms for sparse polynomials. *EUROSAM '79, Lecture Notes in Computer Science*, 72:216–226, 1979.

[45] R. E. Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375–403, March 1990.

[46] R.E. Zippel. *Effective Polynomial Computation.* Kluwer Academic Publishers, Boston, 1993.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'