Learning polynomials with queries: The highly noisy case

Oded Goldreich[*] Ronitt Rubinfeld[†] Madhu Sudan[‡]

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 $\epsilon \stackrel{\text{def}}{=} \delta - \frac{1}{|F|} > 0$. In this case the running time of our algorithm is bounded by a polynomial in $\frac{1}{\epsilon}$, n and exponential in d. 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 $d = 1$).

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.

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, 37, 18, 36] (i.e., that the function is modified only at a small fraction of all possible inputs) or assume that the noise is random [1, 23, 17, 22, 30, 12, 33] (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. [20] (see also [26, 27, 21]). 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 looks very irregular, providing an explanation which fits only part of it is better than nothing. Interestingly, 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[1] 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".

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 $d \geq 1$. 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 $\mathrm{poly}(n/\epsilon)$ and outputs a list including all linear functions (degree $d = 1$ polynomials) which agree with

[*] Weizmann Institute of 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.

[‡] IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (`madhu@watson.ibm.com`).

[1] Actually, in some sense the following is not a generalization but rather falls within the original definition of agnostic learning. The reason being that the sequence of functions which together covers much of f can be viewed as one explanation.

f on a $\delta \stackrel{\text{def}}{=} \frac{1}{|F|} + \epsilon$ fraction of the inputs.

We remark that any algorithm for the reconstruction problem 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 $\binom{n}{d}$. Secondly, 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 Appendix A). Finally, 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" [17] which is commonly believed to be hard when one is only allowed uniformly and independently chosen examples [17, 7, 19]. (Actually, learning parity with noise is considered hard even for random noise, whereas here the noise is adversarial.) In the full version we give evidence that the problem may be hard with respect to d even in the case where $n = 1$.

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.[2] In this case, with high probability, only one polynomial (i.e., p) agrees with f on a δ fraction of the inputs. Thus, in this case, the above algorithm will output only the polynomial p.

A different perspective: Maximum-likelihood decoding of error-correcting codes Maximum likelihood decoding is the term applied to the task of computing the "nearest codeword" from a specified error-correcting code to a given word (cf., [28]). Consider the error-correcting code which encodes $\binom{n+d}{d}$ elements of F by first computing the polynomial obtained by using these elements as the list of coefficients and then evaluating the polynomial at all points in the field. Such codes are fairly well-known — for instance, the Hadamard code is one such code with $F = \mathrm{GF}(2)$ and $d = 1$, and the Reed-Solomon code lies at the other extreme with $n = 1$, and $|F| = O(d)$. One way to interpret our results is as providing a code (over the alphabet F) and a corresponding maximum-likelihood-decoder which works when the error-rate approaches 1. We are not aware of any other case where an approach other than brute force can be used to perform maximum-likelihood 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 (table lookup) for list-decoding.

1.1 Other Related 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 [40], 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 [25]. 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, including such functions as multiplication, division, polynomial multiplication, modular exponentiation, determinant and algebraic functions of trigonometric functions [8, 9, 25, 31]. Self-correctors for f that are polynomial functions over a finite field were found by [4, 25]. The fraction of errors they could correct was improved to almost $1/4$ by [13] and then to almost $1/2$ by [14] (using a solution for the univariate case implicit in [5]). However, when the error is larger than $\frac{1}{2}$ (or, alternatively $\delta < 1/2$), the utility of the standard self-correction approach seems to disappear, since there could be more than one polynomial that agrees with the program on an $\delta < 1/2$ fraction of the inputs.

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. An alternative way of viewing the work of Ar et. al. [3] is as reconstructing the list of polynomials that agree with the f on $\geq \delta$ fraction of the inputs, provided that the input-output relation satisfies some (unknown) algebraic identities. No other polynomial reconstructor seem to be known in the situation where the error of the program is arbitrary and the error rate approaches 1 (or, alternatively, δ approaches 0).

1.2 Rest of this paper

The rest of the paper is organized as follows. In Sections 2 and 3 we construct the algorithm for solving the polynomial reconstruction problem. In Section 4 we analyze the running time of this algorithm, modulo a lemma which bounds the number of polynomials which can agree with a given function at δ fraction of the inputs. In Section 5 we provide two upper bounds, one for the case of general degree d and another (tighter) one for the case of $d = 1$. In Section 6 we consider the case where the output of the black box either agrees with a fixed polynomial or is random. We conclude with some open issues in Section 7.

2 Motivation to the algorithm

We are given oracle access to a function $f : \mathrm{GF}(q)^n \rightarrow \mathrm{GF}(q)$ 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.

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

Linear Polynomials. 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 [16][3]. (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 (degree 1) monomials in which only the first i variables appear. 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. The 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. We pick uniformly $m = \mathrm{poly}(n/\epsilon)$ sequences in $\mathrm{GF}(q)^{n-i}$, and for each such sequence, $(s_{i+1}, ..., s_n)$, we estimate the probabilities

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

(where $\bar{r}, \bar{s}$ denotes the vector $(r_1, \ldots, r_i, s_{i+1}, \ldots, s_n)$. for all $\sigma \in \mathrm{GF}(q)$ (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 $(r_1, ..., r_i)$. 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 $\frac{1}{q} + \frac{\epsilon}{2}$. As shown in [16], the correct candidate passes the test with overwhelming probability. On the other hand, as shown in Section 5, at most $O(1/\epsilon^2)$ candidates (of a certain length) may pass the test.

The above yields a $\mathrm{poly}(nq/\epsilon)$-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 q possible ways, we do the following in order to efficiently find σ's for which $q(\sigma)$ is large enough. We pick uniformly $\overline{s} \stackrel{\text{def}}{=} (s_{i+1}, ..., s_n) \in \mathrm{GF}(g)^{n-i}$, $\overline{r} \stackrel{\text{def}}{=} (r_1, ..., r_{i-1}) \in \mathrm{GF}(q)^{i-1}$ and $r', r'' \in \mathrm{GF}(q)$. Note that for at least an $\epsilon/2$ fraction of the sequences $(\overline{r}, \overline{s})$ there is a polynomial $p(x_1, ..., x_n)$ so that $p(\overline{r}, r, \overline{s}) = f(\overline{r}, r, \overline{s})$ 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 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). Then, with probability $\Omega(\epsilon^3)$, the following two equations hold:

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

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

where $r' \neq r''$. 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 $\mathrm{poly}(n/\epsilon)$ times for each $(i-1)$-prefix. We have:

Theorem 1 *Given oracle access to a function f and parameters ϵ, k, our algorithm runs in $\mathrm{poly}(\frac{k \cdot n}{\epsilon})$-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 $\frac{1}{q} + \frac{\epsilon}{2}$ fraction of the inputs.*

Higher Degree Polynomials. Dealing with polynomials of degree $d > 1$ is more involved. Our plan is to first "isolate" the terms/monomials of degree exactly d and find (candidates for) their coefficients. We need the following definition, which is a generalization of an i-prefix: the (d, i)-prefix of a degree d polynomial $p(x_1, ..., x_n)$ is the polynomial which results by summing up all the degree d monomials of p in which only the variables $x_1, ..., x_i$ appear. Subtracting such a (d, i)-prefix for f leaves us with a problem of degree $d - 1$. Thus, the main part of the algorithm is finding the list of (d, i)-prefixes. This list is built analogously to the above and so we need to show how to extend a list of $(d, i-1)$-prefixes into a list of (d, i)-prefixes. Suppose we get the $(d, i-1)$-prefix p which we want to extend. We select uniformly a sequence $(s_{i+1}, ..., s_n) \in \mathrm{GF}(q)^{n-i}$, and $d + 1$ elements $r^{(1)}, ..., r^{(d+1)} \in \mathrm{GF}(q)$. Now consider the functions

$$f^{(j)}(x_1, ..., x_{i-1}) \stackrel{\text{def}}{=} f(x_1, ..., x_{i-1}, r^{(j)}, s_{i+1}, ..., s_n) - p(x_1, ..., x_{i-1}).$$

Now, suppose that f equals some degree d polynomial and that p is indeed the $(d, i-1)$-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 (d, i)-prefix. The last assertion deserves some elaboration. Consider the (d, 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

[3] We refer to the original algorithm as in [16], not to a simpler algorithm which appears in later versions (cf., [24, 15]).

the different monomials (in p'). (Actually, we obtain a degree d polynomial in variables $x_1, \ldots, x_i$ 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 effected by the instantiation of $x_{i+1}, \ldots, x_n$.) 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 an oracle access to the $f^{(j)}$'s, which in turn may be only relatively close to degree $d-1$ polynomials, 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.

The complete description of this algorithm can be found in Section 3.

3 Algorithm for degree $d > 1$ polynomials

The following algorithm is given oracle access to a function $f : \mathrm{GF}(q)^n \longrightarrow \mathrm{GF}(q)$ and finds the list of all polynomials of degree d in the variables $x_1, \ldots, x_n$, which agree with f on at least a δ fraction of the inputs. The algorithm utilizes two subroutines.

The first subroutine, called Constants, merely returns the list of all constants which agree with the function f on at least δ fraction of the inputs (it may also return other constants but not ones which agree with f on less than $\delta/2$ of the inputs). The main subroutine, called Extend, is given a polynomial p consisting only of monomials of degree d in $x_1, \ldots, x_{i-1}$ and returns a list of polynomials which "extend" it. Each such polynomial consists only of monomials of degree d in $x_1, \ldots, x_i$, and furthermore the monomials of degree d containing only the variables $x_1, \ldots, x_{i-1}$ are exactly those in p. In other words, if p' is in the returned list then

$$p'(x_1, \ldots, x_i) = p(x_1, \ldots, x_{i-1}) + \sum_{j=0}^{d-1} p_j(x_1, \ldots, x_{i-1}) \cdot x_i^{d-j}$$

where p_j consists only of monomials of degree j in $x_1, \ldots, x_{i-1}$.

We now turn to the main part of the algorithm, namely the Extend procedure. Clearly, given a polynomial p, Extend does not return an arbitrary list of admissible polynomials. Instead, it returns a list of polynomials which seem to be good with respect to f. Extend is invoked with a $(d, i-1)$-prefix and returns a list of (d, i)-prefixes all consistent with the input prefix. Furthermore, when invoked with a $(d, i-1)$-prefix of a polynomial p which agrees with f on at least a δ fraction of the inputs, Extend returns a list containing the (d, i)-prefix of p. Finally, the list returned by Extend is never too long; its length is bounded above by $N_{i,d,\delta} = \mathrm{poly}(\frac{n}{\delta-(d/q)})$ (see below).

NOTATIONS AND CONVENTIONS: We write $p_1 \equiv p_2$ if p_1 and p_2 are identical polynomials. Below, we use the term a (d, i)-strict-prefix to mean the difference between the (d, i)-prefix and the $(d, i-1)$-prefix. The notation $f|_{a_i,\ldots,a_n}$ represents the n-variate function $f(x_1, \ldots, x_n)$ restricted by $x_j = a_j$ (i.e., $f|_{a_i,\ldots,a_n}(x_1, \ldots, x_{i-1}) \stackrel{\mathrm{def}}{=} f(x_1, \ldots, x_{i-1}, a_i, \ldots, a_n)$). The parameter τ (approximately the logarithm of the allowed error probability of the algorithm) will be determined later. We also postpone the determination of the parameter α (which governs the "loss in agreement" introduced by the recursive calls).

```
Find-all-poly(f, δ; n, d, q);

  if d = 0 then return Constants(f, δ; n, q).
  L_0 ← {λ}.
  for i = 1 to n do
    L_i ← {} /* List of (d,i)-prefixes */
    for every polynomial p ∈ L_{i-1} do
      L_i = L_i ∪ Extend(f, δ, p, i; n, d, q)

  L ← {}.
  for every polynomial p ∈ L_n do
    L' ← Find-all-poly(f − p, δ; n, d − 1, q);
    L ← L ∪ {p + p' : p' ∈ L'}.

  return(L);
```

The procedure Extend uses two procedures to be described below. The first procedure, Comp-coeff, gets as (main) inputs a sequence of $d+1$ polynomials, $p^{(1)}, \ldots, p^{(d+1)}$, each of degree $d-1$ in the first $i-1$ variables and a corresponding list of $d+1$ values, $r^{(1)}, \ldots, r^{(d+1)}$. Using interpolation, it returns a polynomial, p, of degree d in the variables $x_1, \ldots, x_i$ such that $p|_{r^{(j)}} \equiv p^{(j)}$ for all $j = 1, \ldots, d+1$. (For details see Lemma 7.)

The second procedure, Test-valid, is given oracle access to a function $f'(x_1, \ldots, x_i)$ (= $f|_{s_{i+1},\ldots,s_n} - p$) and a degree d polynomial $p'(x_1, \ldots, x_i)$. It tests by the obvious sampling method whether p' agrees with f' on $\delta' = \frac{\delta}{2}$ fraction of inputs. The correctness of the algorithm is shown in the following lemma.

Lemma 2 *Suppose that* Constants, Test-valid *and* Comp-coeff *are perfectly correct; that is,*

(H1) Constants$(f, \delta; n, q)$ *contains each field element, e, so that f assumes the value e on at least a δ fraction of the inputs;*

(H2) Test-valid$(f', p', \delta'; i, d, q)$ *answers* yes *if and only if f' agrees with p' on at least a δ' fraction of the inputs; and*

(H3) Comp-coeff$(p^{(1)}, \ldots, p^{(d+1)}, r^{(1)}, \ldots, r^{(d+1)}; i, d, q)$ *returns a polynomial, p, of degree d in the variables $x_1, \ldots, x_i$ so that $p|_{r^{(j)}} \equiv p^{(j)}$, for all $j = 1, \ldots, d+1$.*

Suppose that p^* is a degree d polynomial which agrees with f on at least an $\delta > \frac{(d+1)\tau}{(1-\alpha)\alpha^{2d-1}\cdot q}$ fraction of the inputs. Then, with probability at least $1 - 2dn \cdot 2^{-\tau}$, the list output by Find-all-poly$(f, \delta;\ n, d, q)$ *contains the polynomial p^*.*

```
Extend(f, δ, p, i;  n, d, q).  (additional parameters τ and α)

  L' ← {}.
  repeat τ/((1−α)δ) times /* main loop */
    Pick s_{i+1}, ..., s_n uniformly in GF(q).
    Pick m =def τ/((1−α)αδ) · (d+1) distinct elements r^(1), ..., r^(m), uniformly from GF(q).
    for j = 1 to m do
      f^(j) ← f|_{r^(j), s_{i+1}, ..., s_n} − p.
      L^(j) ← Find-all-poly(f^(j), α^2 · δ;  i, d−1, q).

    for every set {j_1, ..., j_{d+1}} ⊂ {1, ..., m}
      for every (d+1)-tuple (p^(j_1), ..., p^(j_{d+1})) with p^(j_k) ∈ L^(j_k)
        p' ← Comp-coeff(p^(j_1), ..., p^(j_{d+1}), r^(j_1), ..., r^(j_{d+1});  i, d, q).
        if Test-valid(f|_{s_{i+1}, ..., s_n} − p, p', α · δ;  i, d, q) then
          L' ← L' ∪ {p + (d, i)-strict-prefix of p'};
    /* end of main loop */
  return(L').
```

The lower bound on δ is used for guaranteeing that in each invocation of Extend (regardless of the depth of recursion) the parameter m is not larger than the field size. This is required in order to allow the selection of m *different* field elements (see footnotes in the proof below).

Proof: The proof proceeds by induction on d. The case $d = 0$ follows by the hypothesis H1. The induction step uses the other two hypotheses (i.e., H2 and H3). First, observe that (for the next d) it suffices to show that the (d, n)-prefix of p^* (i.e., all degree d monomials) appears in the list $\mathcal{L}_n$ (since the induction hypothesis will be used to reconstruct all lower degree monomials of p^*).

We show by induction on i that the (d, i)-prefix of p^* appears in the list $\mathcal{L}_i$. Here the base case ($i = 0$) holds vacuously and so we consider the invocation of Extend$(f, \delta, p, i; n, d, q)$, where p is the $(d, i-1)$-prefix of p^* (i.e., by the induction hypothesis $p \in \mathcal{L}_{i-1}$). Call a sequence $\overline{s} = (s_{i+1}, ..., s_n) \in GF(q)^{n-i-1}$ good if the function $f|_{\overline{s}}$ and the polynomial $p^*|_{\overline{s}}$ agree on at least an $\alpha \cdot \delta$ fraction of the inputs. Clearly, at least a $(1-\alpha) \cdot \delta$ fraction of the sequences are good and so with probability at least $1 - 2^{-\tau}$ a good sequence $\overline{s}$ is selected in one of the iterations of the main loop. Let us consider such an iteration and fix the good sequence $\overline{s}$ for the rest of the proof.

We say that $r \in \mathrm{GF}(q)$ is $\overline{s}$-good if the function $f|_{r,\overline{s}}$ and the polynomial $p^*|_{r,\overline{s}}$ agree on at least an $\alpha \cdot \alpha\delta$ fraction of the inputs. Clearly, at least $(1-\alpha) \cdot \alpha\delta$ of the r's are $\overline{s}$-good and so, with probability at leat $1 - 2^{-\tau}$, at least $d+1$ of the m "uniformly"[4] chosen $r^{(j)}$'s are $\overline{s}$-good. Let us assume, without loss of generality, that $r^{(1)}, ..., r^{(d+1)}$ are all $\overline{s}$-good and fix them too for the rest of the proof.

Let us denote the polynomial $(p^*|_{r^{(j)},\overline{s}}) - p$ by $p^{(j)}$. We first observe that $p^{(j)}$ is a degree $d-1$ polynomial in the variables $x_1, ..., x_{i-1}$. (Note that the only monomials of degree d in $p^*|_{r^{(j)},\overline{s}}$ are those which have all variables in $\{x_1, ..., x_{i-1}\}$ and that all these monomials are cancelled out by p.) Next, we observe that $f^{(j)} \stackrel{\rm def}{=} (f|_{r^{(j)},\overline{s}}) - p$ agrees with the polynomial $p^{(j)}$ on at least an $\alpha^2\delta$ fraction of the inputs. By the induction hypothesis (for $d-1$), the polynomial $p^{(j)}$ must be in the list returned by Find-all-poly$(f^{(j)}, \alpha^2\delta;\ i, d-1, q)$ and so $p^{(j)} \in \mathcal{L}^{(j)}$. It remains to examine what happens when Comp-coeff is invoked with the polynomials $p^{(1)}, ..., p^{(d+1)}$ and the values $r^{(1)}, ..., r^{(d+1)}$. Let p' denote the polynomial returned in this invocation and recall that (by the hypothesis H3) $p'|_{r^{(j)}} \equiv p^{(j)}$ for $j = 1, ..., d+1$. Using the definition of $p^{(j)}$ we get $p'(x_1, ..., x_{i-1}, r^{(j)}) \equiv [p^*|_{\overline{s}}(x_1, ..., x_{i-1}, r^{(j)}) - p(x_1, ..., x_{i-1})]$ for $j = 1, ..., d+1$. Namely, these two degree d polynomials on i variables are identical for $d+1$ distinct[5] instantiations of the last variable and thus these two polynomials are identical; namely

$$p'(x_1, ..., x_{i-1}, x_i) \equiv p^*|_{\overline{s}}(x_1, ..., x_{i-1}, x_i) - p(x_1, ..., x_{i-1})$$

Next, we observe that the (d, i)-strict-prefix of p^* equals the (d, i)-strict-prefix of $p^*|_{\overline{s}}$ (actually, the corresponding (d, i)-prefixes are identical). Thus, it follows that the (d, i)-strict-prefix of p' equals the (d, i)-strict-prefix of p^* (equiv., of $p^* - p$) and that p' agrees with $(f|_{\overline{s}}) - p$ on at least an $\alpha\delta$ fraction of

[4] Actually, the $r^{(j)}$'s are selected uniformly among all m-tuples of distinct elements.

[5] Here is where we use the fact that the $r^{(j)}$'s are distinct. Clearly, the conclusion would not have hold otherwise.

the inputs. By the hypothesis H2 it follows that p' will pass Test-valid$((f|_{\bar{s}})-p, p', \alpha\delta; i, d, q)$ and so the (d,i)-prefix of p^* (i.e., $p+p''$ where p'' is the (d,i)-strict-prefix of p^*) will be in the list returned by Extend. The lemma follows. ▮

4 Bounding the running time

In all our complexity estimates we assume that it is possible to obtain the value of a function (be it a function given as oracle or a polynomial represented explicitly) at a given point at unit cost. Likewise, all standard field operations and algorithmic conventions are implemented at unit cost. In addition, we need to set the additional parameters for Extend: we set $\tau = k + O(d\log(knd/\delta))$ and $\alpha = 1-(1/d)$. We stress that here d is the degree-parameter in the initial/main invocation of algorithm Find-all-Poly (rather than in the recursive calls).

Theorem 3 *Given oracle access to a function f and parameters δ, k and d, so that $\delta = \Omega(\max\{(d+1)^3\cdot\log(knd)\cdot\frac{1}{q}, \sqrt{d/q}\})$, algorithm* Find-all-Poly *runs in $\mathrm{poly}((k\cdot nd/\delta)^{d+1})$-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.*

Using additional ideas it is possible to relax the condition on δ. Specifically, it suffices to have $\delta = \Omega(\sqrt{d/q})$. Furthermore, it is also possible to improve the running-time. One important idea is to randomize the problem so that most sequences and elements are good in the sense used in the proof of Lemma 2 (above). This is done by a uniformly selected linear transformation of the original variables x_i's into new variables y_i's (i.e., each y_i is a linear combination of x_i's and vice versa). The theorem follows from Lemma 2 (above) and the following four lemmas.

Lemma 4 Constants$(f,\delta;n,q)$ *can be implemented in time $\mathrm{poly}(k/\delta)$ so that with probability at least $1-2^{-k}$ the list output by the subroutine contains all field elements which agree with f on at least δ fraction of the inputs and no field elements which agree with f on less than $\delta/2$ fraction of the inputs.*

We assume, without loss of generality, that Constants$(f,\delta;n,q)$ never returns more than $2/\delta$ field elements (as otherwise, in the rare case this does not hold, we can halt with an error message).

Lemma 5 Test-valid$(f', p', \delta';\ i, d, q)$ *can be implemented in time $\mathrm{poly}(k/\delta')$ so that the following holds with probability at least $1-2^{-k}$: If p' agrees with f' on at least an δ' fraction of the inputs then the test accepts and if p' agrees with f' on at most an $\frac{\delta'}{2}$ fraction then the test rejects.*

Let us denote by $N_{i,d,\delta}$ an upper bound on the number of i-variate degree d polynomials which agree with a particular function on an δ fraction of the inputs. (See a specific bound in Lemma 6.) We assume, without loss of generality, that the number of polynomials, p', passing Test-valid$((f|_{\bar{s}})-p, p', \delta/2;\ i, d, q)$ in a single iteration of the main loop of Extend$(f,\delta,p,i;\ n,d,q)$ is bounded above by $N_{i,d,\delta/2}$ (since, again, in the rare case this does not hold the algorithm can be made to halt with an error message).

Lemma 6 *For $\delta > \frac{1}{q} + \sqrt{\frac{d-1}{q}}$ we may set $N_{i,d,\delta} = \frac{i}{\delta^2-(d/q)}$.*

Proof: Immediate by Theorem 10 Part (2). ▮

Lemma 7 Comp-coeff$(p^{(1)},...,p^{(d+1)},r^{(1)},...,r^{(d+1)}; m, d, q)$ *can be implemented in time $\binom{m+d}{d}\cdot\mathrm{poly}(d)$.*

Proof: Let $p^{(j)}(x_1,...,x_{m-1}) = \sum_I c_I^{(j)}\prod_{i\in I}x_i$ be degree $d-1$ polynomials, for $j=1,...,d+1$; that is, the I's are multisets of size at most $d-1$ (of $\{1,...,m-1\}$). We need to find a polynomial $p(x_1,...,x_m)$ of degree d so that $p(x_1,...,x_{m-1},r^{(j)}) = p^{(j)}(x_1,...,x_{m-1})$, for $j = 1,...,d+1$. Let $p(x_1,...,x_{m-1},x_m) = \sum_K c_K\prod_{i\in K}x_i$, where the K's are multisets of size at most d (of $\{1,...,m\}$). It follows that for every multiset $I\subseteq\{1,...,m-1\}$ we have, for every $j=1,...,d+1$,

$$c_I^{(j)} = \sum_{t=0}^{|I|-d} c_{I\cup\{m\}^t}(r^{(j)})^t$$

where $I\cup\{m\}^t$ denotes the multiset consisting of I and t occurrences of m. Note that all $c_I^{(j)}$'s and $r^{(j)}$'s are known to us and we are looking for the $c_{I\cup\{m\}^t}$'s. For each I we obtain an ordinary interpolation problem (which is solvable by linear algebra) and so the lemma follows. ▮

Lemma 8 *Set $\alpha = 1-\frac{1}{2d}$ and suppose that $\delta > 2\sqrt{d/q}$. Then the running-time of* Find-all-Poly$(f,\delta;\ n,d,q)$ *is at most*

$$\left(\frac{nd\tau}{\delta}\right)^{O(d)}$$

Proof omitted (see full version).

5 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. We then present examples indicating that our bound in the linear case is essentially tight. We have evidence as to why the bound in the general degree case may not have a simple improvement. Details of the latter will be included in the final version.

Theorem 9 *Let $\epsilon > 0$. For a function $f: \mathrm{GF}(q)^l\to\mathrm{GF}(q)$, if $f_1,\ldots,f_m:\mathrm{GF}(q)^l\to\mathrm{GF}(q)$ are distinct linear functions which satisfy*

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

$$\mathit{thenm} \leq \left(1-\frac{1}{q}\right)^2 \cdot \frac{1}{\epsilon^2}$$

Proof: We first fix some notation. We use α to denote $q/(q-1)$ and Q to denote q^l. For $i \in \{1,\ldots,m\}$ and $x \in \mathrm{GF}(q)^l$ let $\chi_i(x) \equiv 1$ if $f_i(x) \neq f(x)$ and 0 otherwise. For $i \in \{1,\ldots,m\}$, $t \in \mathrm{GF}(q)$ and $x \in \mathrm{GF}(q)^l$ let $\chi_i^{(t)}(x) \equiv 1$ if $f_i(x) - f(x) = t$ and 0 otherwise. Let $w_i \equiv |\{x : f_i(x) \neq f(x)\}|$ and let $\overline{w} = \frac{\sum_{i=1}^m w_i}{m}$.

The fact that the f_i's are close to f implies that for all i, $w_i \leq (1-\epsilon-\frac{1}{q}) \cdot Q$.

Our proof generalizes a proof due to S. Johnson (c.f., MacWilliams and Sloane [28]) for the case $q = 2$. The central quantity used to bound m in their case can be generalized in one of the two following ways:

$$S \equiv \sum_{i,j,x} \chi_i(x)\chi_j(x).$$

$$S' \equiv \sum_{i,j,x}\sum_{t\neq 0} \chi_i^{(t)}(x)\chi_j^{(t)}(x).$$

The first sums over all i,j, the number of inputs for which f_i and f_j both differ from f. The second sums over all i,j, the number of inputs for which f_i and f_j both differ by the same amount from f. (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 $S + S'$**:** Let $N_x = |\{i | f_i(x) \neq f(x)\}|$ and let $N_x^{(t)} = |\{i | f_i(x) - f(x) = t\}|$. Then we can lower bound S as follows:

$$S = \sum_{i,j,x} \chi_i(x)\chi_j(x) = \sum_x N_x^2 \geq \frac{(m\overline{w})^2}{Q}.$$

The last inequality above follows from the fact that subject to the condition $\sum_x N_x = m\overline{w}$, the sum of N_x's squared is minimized when all the N_x's are equal.

Similarly we lower bound S' as follows:

$$S' = \sum_{i,j,x}\sum_{t\neq 0} \chi_i^{(t)}(x)\chi_j^{(t)}(x) = \sum_x \sum_{t \neq 0} (N_x^{(t)})^2 \geq \frac{(m\overline{w})^2}{(q-1)Q}$$

Thus by adding the two lower bounds above we obtain:

$$S + S' \geq \frac{(m\overline{w})^2}{Q} + \frac{(m\overline{w})^2}{(q-1)Q} = \frac{\alpha m^2 \overline{w}^2}{Q}. \qquad (2)$$

Upper bound on $S + S'$**:** For the upper bound we define the following quantities: For distinct $i, j \in \{1,\ldots,m\}$ and $t_1, t_2 \in \mathrm{GF}(q)$, let

$$M_{t_1 t_2}^{(ij)} \equiv |\{x | \chi_i^{(t_1)}(x) = \chi_j^{(t_2)}(x) = 1\}|.$$

Then we can express S and S' as:

$$S = \sum_{i,j}\sum_{t_1\neq 0}\sum_{t_2 \neq 0} M_{t_1t_2}^{(ij)} \quad \text{and} \quad S' = \sum_{i,j}\sum_{t\neq 0} M_{tt}^{(ij)}.$$

We start by upper bounding the internal sum above for fixed distinct pair i and j. By the fact that $f_i(x) = f_j(x)$ for (at most) Q/q values of x, we have

$$\sum_{t\neq 0} M_{tt}^{(ij)} \leq Q/q - M_{00}^{(ij)}.$$

To bound the other term in the summation above we use inclusion-exclusion as follows:

$$\begin{aligned}
&\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(ij)} \\
&= \sum_{t_1}\sum_{t_2} M_{t_1t_2}^{(ij)} - \sum_{t_1} M_{t_1 0}^{(ij)} - \sum_{t_2} M_{0 t_2}^{(ij)} + M_{00}^{(ij)} \\
&= Q - (Q - w_i) - (Q - w_j) + M_{00}^{(ij)} \\
&= w_i + w_j - Q + M_{00}^{(ij)}.
\end{aligned}$$

Combining the bounds above we have (for $i \neq j$)

$$\begin{aligned}
&\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(ij)} + \sum_{t\neq 0} M_{tt}^{(ij)} \\
&\leq Q/q - M_{00}^{(ij)} + w_i + w_j - Q + M_{00}^{(ij)} \\
&= w_i + w_j - \frac{Q}{\alpha}.
\end{aligned}$$

Also observe that if $i = j$, then the quantity $\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(ii)} = \sum_{t\neq 0} M_{tt}^{(ii)} = w_i$.

We now combine the bounds above as follows:

$$\begin{aligned}
S + S' &= \sum_i \left(\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(ii)} + \sum_{t\neq 0} M_{tt}^{(ij)}\right) \\
&\quad + \sum_{i\neq j} \left(\sum_{t_1\neq 0}\sum_{t_2\neq 0} M_{t_1t_2}^{(ij)} + \sum_{t\neq 0} M_{tt}^{(ij)}\right) \\
&\leq \sum_i 2w_i + \sum_{i\neq j}(w_i + w_j - \frac{Q}{\alpha})
\end{aligned}$$

Simplifying the right hand side above, we get:

$$S + S' \leq 2m^2\overline{w} - m(m-1)Q/\alpha. \qquad (3)$$

Putting it together: By comparing the bounds (2) and (3), we have $2m^2\overline{w} - m(m-1)Q/\alpha \geq \frac{m^2\overline{w}^2}{Q}\alpha$ so $m\alpha/Q(\overline{w} - Q/\alpha)^2 \leq Q/\alpha$. Substituting $\overline{w} \leq Q(1-\frac{1}{q}) - \epsilon Q = Q/\alpha - \epsilon Q$, we get

$$m \leq \left(\frac{\alpha}{\epsilon}\right)^2 = \left(\frac{1}{\epsilon}\right)^2 \cdot \left(1 - \frac{1}{q}\right)^2.$$

▮

For general d, we have the following theorem.

Theorem 10 *Let $\delta > 0$ and $f : \mathrm{GF}(q)^l \to \mathrm{GF}(q)$. Suppose $f_1, \ldots, f_m : \mathrm{GF}(q)^l \to \mathrm{GF}(q)$ are distinct polynomials of degree at most d which satisfy*

$$\forall j \in \{1, \ldots, m\}, \quad \Pr_{x \in \mathrm{GF}(q)^l} [f(x) = f_i(x)] \geq \delta.$$

Then the following bounds hold:

1. *If $\delta \geq \sqrt{2 + \frac{d}{4q}} \cdot \sqrt{\frac{d}{q}} - \frac{d}{2q}$ then $m < \frac{2}{\delta + d/(2q)}$. (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 $m \leq \frac{(q-d)(q-1)}{q^2} \cdot \frac{1}{(\delta-(1/q))^2-(q-1)(d-1)/q^2}$. (For small $\frac{d}{q}$ the above expressions are approximated by $\delta > \frac{1}{q} + \sqrt{\frac{d-1}{q}}$ and $m < \frac{1}{\delta^2-((d+2\delta-1)/q)}$, respectively.)*

Remark The above bounds apply in different situations and yield different bounds on m. The first applies for larger values of δ and yields a bound which is $O(\frac{1}{\delta})$. The second bound applies 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 $m' \leq m$ and let $Q = q^l$. We count the number of points $x \in \mathrm{GF}(q)^l$ that satisfy the property that one of the first m' polynomial agree with f on x. Namely, let $\chi_i(x) = 1$ if $f_i(x) = f(x)$ and $\chi_i(x) = 0$ otherwise. Then, by inclusion-exclusion we get

$$\begin{aligned} Q &\geq |\{x : \exists i \ \chi_i(x) = 1\}| \\ &\geq \sum_{i=1}^{m'} \sum_x \chi_i(x) - \sum_{1 \leq i < j \leq m'} \sum_x \chi_i(x)\chi_j(x) \\ &\geq m' \cdot \delta Q - \binom{m'}{2} \cdot |\{x : f_1(x) = f_2(x)\}| \end{aligned}$$

Since two degree d polynomials f_1 and f_2 can agree on at most $\frac{d}{q} \cdot Q$ points [10, 35, 38], we get:

$$m'\delta Q - \frac{m'(m'-1)}{2} \cdot \frac{dQ}{q} \leq Q.$$

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

Let $\beta = \frac{d}{2q}$. Then $g(y) = \beta m^2 - (\beta + \delta)m + 1$. The roots, α_1 and α_2 are real, provided that $\Delta \stackrel{\text{def}}{=} (\beta + \epsilon)^2 - 4\beta$ is positive which follows from a stronger requirement (see below). Let α_1 be the smaller root. To guarantee $\alpha_2 - \alpha_1 \geq 1$ we require $2\frac{\sqrt{\Delta}}{2\beta} \geq 1$ which translates to $\Delta \geq \beta^2$ (and hence $\Delta > 0$ as required above). We need to show

$$\begin{aligned} &(\beta + \delta)^2 - 4\beta \geq \beta^2 \\ &\quad \text{implied by} \quad (\beta + \delta)^2 \geq \beta^2 + 4\beta \\ &\quad \text{implied by} \quad \delta \geq \sqrt{\beta^2 + 4\beta} - \beta. \end{aligned}$$

Plugging in the values of δ and β we find that the last inequality is exactly as guaranteed in the theorem statement. Lastly observe that:

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

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

We sketch the proof of part (2). The proof is an extension of the proof of Theorem 9. We define S and S' as in Theorem 9. Here $\overline{w} \leq (1 - \delta)Q = (1 - \epsilon - d/q)Q$. The lower bound found for $S + S'$ still applies i.e.,

$$S + S' \geq \frac{m^2 \overline{w}^2 q}{Q \cdot (q-1)}. \tag{4}$$

The upper bound gets modified due to the fact that two polynomials agree on at most $\frac{d}{q}$ fraction of the points (and not $1/q$ fraction). This yields:

$$S + S' \leq 2m^2\overline{w} - m(m-1)\frac{q-d}{q}Q \tag{5}$$

Thus we find that

$$m \leq \frac{q-d}{q} \cdot \frac{1}{(\frac{\overline{w}}{Q})^2 \frac{q}{q-1} + \frac{q-d}{q} - 2\frac{\overline{w}}{Q}},$$

provided $(\frac{\overline{w}}{Q})^2 \frac{q}{q-1} + \frac{q-d}{q} - 2\frac{\overline{w}}{Q} \geq 0$. Let $g(x) \stackrel{\text{def}}{=} \frac{q}{q-1}x^2 - 2x + \frac{q-d}{q}$. We need to bound δ so that $g(1-\delta) > 0$. (For $d = 1$ we have $g(x) > 0$ for all $x \neq 1 - \frac{1}{q}$ indicating that the bound holds for all $\delta > 1/q$.) For $d > 1$ the function $g(x)$ is positive provided that $x < \frac{1-\sqrt{1-(q-d)/(q-1)}}{q/(q-1)}$. Thus, $\delta > \frac{1}{q} + \frac{q-1}{q} \cdot \sqrt{\frac{d-1}{q-1}}$ suffices (which is the condition in Part (2)). The corresponding bound is $m \leq \frac{q-d}{q} \cdot \frac{1}{g(1-\delta)}$ which, using $\frac{q-1}{q} \cdot g(x) = (x - \frac{q-1}{q})^2 - \frac{q-1}{q^2} \cdot (d-1)$, yields the bound claimed in Part (2). ∎

The following result shows that Theorem 9 is tight for $\delta = O(1/q)$ (and $d = 1$), whereas Part (1) of Theorem 10 is tight for $\delta = \Theta(1/\sqrt{q})$ and $d = 1$.

Proposition 11 *Given a prime p, and integers d and $k > 1$, let $\delta = k/p$. Then, there exists a function $f : \mathrm{GF}(p) \to \mathrm{GF}(p)$ and at least $m \stackrel{\text{def}}{=} \frac{1}{5(k-1)\delta^2}$ 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 g_i so that R becomes a function which still agrees with each g_i on k inputs.

Let $l = \lfloor p/k \rfloor$. 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 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 $p(l+1)/2$ distinct pairs (a, b) s.t. $a(k-1)+b < l$. Fix $a < l = \lfloor p/k \rfloor$. Then there are at least $l - (k-1)a$ possible values for b, so that the total number of pairs becomes $\sum_{a=0}^{\frac{l}{k-1}-1} l - (k-1)a = \frac{l^2}{k-1} - (k-1) \cdot \binom{\frac{l}{k-1}-1}{2} > \frac{l^2}{2(k-1)}$.

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 $(i_1, j_1), (i_1, j_2) \in R''$, $i_1 \neq i_2$ implying that R'' can be extended to a function $f : \mathrm{GF}(p) \to \mathrm{GF}(p)$, 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 $f : \mathrm{GF}(p) \to \mathrm{GF}(p)$ 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 $(1/p) + \delta < \frac{3}{2} \cdot \delta$). ▮

6 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 is chosen randomly on other points. As a consequence we find that for functions constructed in this manner the output of our reconstruction algorithm will be a single polynomial — namely, p.

Theorem 12 *Let $\delta > \frac{2(d+1)}{q}$. 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 $1 - \exp\{(n^d \log_2 q) - (\delta/4)^2 q^{n-1}\}$, the polynomial p is the only degree d polynomial which agrees with f on at least an $\delta/2$ fraction of the inputs.

7 Conclusions

The main feature controlling the running time of the reconstruction algorithm described in this paper is the bound on the number of polynomials which can agree with a given function at δ fraction of the places. Thus by improving any of the bounds given here (or presenting similar bounds in other situations) one can improve the running time of the algorithm presented (or extend it to other cases). The case of degree d polynomials with $\delta \leq \sqrt{d/q}$ seems to be a prime candidate for analysis here. We seem to have some evidence that this bound may grow as $(1/\delta)^{d+1}$ for small enough δ.

Lastly we speculate on the need for the exponential dependence on d. In the full version we point out the NP-hardness of a related univariate question which asks for the best polynomial fitting a relation specified on $O(d)$ points. The fact that this evidence applies only to learning *relations* (rather than functions) without *queries* makes this relatively weak. However when specialized to the univariate case, there is no known separation of the "reconstruction" problem between the case on learning with or without queries. (i.e., both are solvable, in time poly(d), if error is bounded away from half.) Also we do not know of instances where learning relations is harder than learning functions. Thus all this accumulates to some feeling that maybe this exponential dependence may be inherent.

Acknowledgments

We are most grateful to Mike Kearns and Dana Ron for discussion regarding agnostic learning. We also wish to thank Eyal Kushilevitz and Yishay Mansour for comments regarding an early version of this paper.

References

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

[2] D. Angluin, M. Krikis. Learning with Malicious Membership Queries and Exceptions. *COLT*, 1994.

[3] S. Ar, R. Lipton, R. Rubinfeld, M. Sudan. Reconstructing Algebraic Functions from Mixed Data. *FOCS*, 1992.

[4] D. Beaver and J. Feigenbaum. Hiding Instance in Multioracle Queries. *STACS*, 1990.

[5] E. Berlekamp and L. Welch. Error Correction of Algebraic Block Codes. *US Patent*, Number 4,633,470, 1986.

[6] A. Blum and P. Chalasani. Learning Switching Concepts. *COLT*, 1992.

[7] A. Blum, M. Furst, M. Kearns, R. Lipton. Cryptographic Primitives Based on Hard Learning Problems. *CRYPTO*, 1993.

[8] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *STOC*, 1990.

[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] Y. Freund, D. Ron. Learning to Model Sequences Generated by Switching Distributions. *COLT*, 1995.

[12] 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 (1993), 705–726.

[13] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. *STOC*, 1991.

[14] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Info. Proc. Letters*, 43 (1992), 169–174.

[15] O. Goldreich. Foundations of Cryptography – Fragments of a Book. Weizmann Institute of Science, February 1995. Available from the ECCC, email `ftpmail@ftp.eccc.uni-trier.de` (use subject 'help eccc' for initial instructions).

[16] O. Goldreich and L.A. Levin. A Hard-Core Predicate for any One-Way Function. *STOC*, 1989.

[17] M. Kearns. Efficient noise-tolerant learning from statistical queries. *STOC*, 1993.

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

[19] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire and L. Sellie. On the Learnability of Discrete Distributions, *STOC*, 1994.

[20] M. Kearns, R. Schapire and L. Sellie, Towards efficient agnostic learning. *COLT*, 1992.

[21] P. Koiran. Efficient Learning of Continuous Neural Nets. *COLT*, 1994.

[22] E. Kushilevitz, Y. Mansour. Learning decision trees using the Fourier spectrum. *STOC*, 1991.

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

[24] 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.

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

[26] Wolfgang Maass. Efficient Agnostic PAC-Learning with Simple Hypotheses. *COLT*, 1994.

[27] Wolfgang Maass. Agnostic PAC-Learning of Functions on Analog Neural Nets. *Proc. 7th IEEE Conf. on Neural Information Processing Systems.*

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

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

[30] Dana Ron and Ronitt Rubinfeld. Learning fallible finite state automata. *COLT*, 1993. To appear in *Machine Learning*, COLT '93 special issue.

[31] R. Rubinfeld. Robust Functional Equations and their Applications to Program Testing. *FOCS*, 1994.

[32] R. Rubinfeld and M. Sudan. Robust Characterizations of Polynomials and their Applications to Program Testing. To appear in *SIAM J. of Computing*. Also available as IBM Research Report RC 19156 (83446) 9/9/93 and Cornell CS Tech. Report 93–1387, September 1993.

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

[34] Y. Sakakibara and Rani Siromoney. A noise model on learning sets of strings. *COLT*, 1992.

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

[36] R. H. Sloan. Types of noise in data for concept learning. *COLT*, 1988.

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

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

[39] R.E. Zippel. Interpolating Polynomials from their Values. *J. Symbolic Computation* 9, pages 375-403, 1990.

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

A Justifying the $\frac{d}{|F|}$ Threshold

Proposition 13 *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 $c_2, \ldots, c_n \in \mathrm{GF}(q)$. 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 $(a_1, \ldots, a_n)$ satisfying either $a_1 \in \{1, \ldots, (d-1)\}$ 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. ∎