Reconstructing Algebraic Functions from Mixed Data

Sigal Ar* Richard J. Lipton† Ronitt Rubinfeld‡ Madhu Sudan§

Abstract

We consider the task of reconstructing algebraic functions given by black boxes. Unlike traditional settings, we are interested in black boxes which represent several algebraic functions - $f_1, \ldots, f_k$, where at each input x, the box arbitrarily chooses a subset of $f_1(x), \ldots, f_k(x)$ to output. We show how to reconstruct the functions $f_1, \ldots, f_k$ from the black box. This allows us to *group* the sample points into sets, such that for each set, all outputs to points in the set are from the same algebraic function. Our methods are robust in the presence of errors in the black box. Our model and techniques can be applied in the areas of *computer vision, machine learning, curve fitting and polynomial approximation, self-correcting programs and bivariate polynomial factorization.*

1 Introduction

Suppose you are given a large set of points in the plane and you are told that an overwhelming majority of these points lie on one of k different algebraic curves of degree bound d (but you are not told anything else about the curves). Given the parameters k and d, your task is to determine or "reconstruct" these algebraic curves, or alternatively, to *group* the points into sets, each of which is on the same degree d curve. Related versions of this problem are also of interest, such as extensions to higher dimensions, and a setting where instead of the points being given in advance, one is allowed to make queries of the form "what is the value of one of the curves at point x?"[1]. This problem seems to be fundamental and its solution has a variety of applications:

Computer Vision Consider a computer vision system for a robot that picks parts out of a bin. The input to the system contains an intensity map of the scene. The robot can distinguish between the parts by extracting edges from the image. Current edge detection algorithms use discretised differential operators to extract edges (e.g. [23][9]). These algorithms produce output consisting of a bit map, where for every image point (x, y), the bit value of the point, $e(x, y)$, is set to 1 if and only if this point lies on an edge. For many vision applications it is then desired to connect between neighboring points to achieve a more compact representation of the edge map. This problem, known as "the grouping problem", is complicated by the fact that the parts are cluttered, they may be nonconvex, they may contain holes, and the edges are distributed along the image creating noncontinuous sequences of curves. No polynomial time algorithm has been found for this problem.

Under the assumption that the edges of the parts are given by piecewise algebraic curves, and that the edge detection process produces results which are free of precision error, our algorithm transforms edge maps into piecewise polynomial curves in polynomial time. The second assumption is unrealistic in real computer vision applications. However, we feel that it may be possible to make a simple modification to our algorithm so that it works in a real setting.

Learning Our mechanism can be used to extend some well-known results on learning boolean functions to the setting of learning real-valued functions. Here, we present two specific instances of such situations:

- In the study of economics, the price-demand curve is often considered to be well-described by an algebraic function (e.g. $f(x) = c/x$ or $f(x) = -a \cdot x + b$). However, it is also the case that this curve may change [17]. In particular, there may be several unknown price-demand curves which apply in various situations - one may correspond to the behavior found

*Princeton University, Supported by Dept. of Navy Grant #N00014-85-C-0456, by NSF PYI grant CCR-9057486 and a grant from MITL.

†Princeton University. Part of this research was done while at Matsushita Information Technology Labs.

‡Cornell University. Part of this research was done while at Hebrew University and Princeton University and supported by DIMACS, NSF-STC88-09648.

§U.C. Berkeley. Supported in part by NSF Grant No. CCR 88-96202.

[1] The answer to such a query will not specify which of the k curves was used to compute the value.

when the country is at war, another may correspond to typical behavior on a Sunday, a third may apply after a stock market crash, and yet another behavior may be found after a change in the tax structure. Some of the factors that determine which curve applies may be obvious, however others (such as a change in behavior due to the day of the week) may occur because of more subtle reasons. The task of learning the price-demand relationship may be decomposed into the two subtasks of first determining the unknown curves, and then learning what determines the move from one curve to another. Our algorithm gives a solution for the first task.

- A *polynomial-valued decision list* given by a list of terms (conjuncts of literals), $(D_i)_{i=1}^{k}$ over boolean variables $y_1, \ldots, y_n$, and a list of univariate polynomials, $(f_i)_{i=1}^{k+1}$ in a real variable x, represents a real valued function f as follows:

  $$f(x, y_1, \ldots, y_n) = f_i(x)$$

  (where i is the least index such that $D_i(y_1, \ldots, y_n)$ is true)

  If the terms are restricted to being conjuncts of at most c literals, we call it a *polynomial-valued c-decision list*. This is an extension of the *boolean decision list* model defined by Rivest in [24], where the polynomials f_i are restricted to being the constants 0 or 1.

  In [24], Rivest shows that the class of boolean c-decision lists are learnable in polynomial time under the Valiant model of PAC learning. Using our techniques, in combination with Rivest's algorithm, we can extend this result to show that the class of polynomial-valued c-decision lists can be learned in polynomial time.

In general, our results imply that any function on input x and boolean attributes $(y_1, \ldots, y_m)$ which uses $(y_1, \ldots, y_m)$ to select f_i from a set of polynomial functions $f_1, \ldots, f_k$ and then computes and outputs $f_i(x)$, can be learned, as long as the selector function can be learned.

Independent of our work, Blum and Chalasani [6] also consider a model of learning from examples where the examples may be classified according to one of several different concepts. In their model an adversary controls the decision of which concept would be used to classify the next example. Under this model they study the task of learning boolean-valued concepts such as k-term DNF's and probabilistic decision lists.

Curve Fitting A typical curve fitting problem takes the following form: Given a set of points $\{(x_i, y_i)\}_{i=1}^{t}$ on the plane, give a simple curve that "fits" the given points. Depending on the exact specification of the "fit" the problem takes on different flavors: for instance, if the curve is to pass close to all the points, then this becomes a *uniform approximation* problem [25], while if the curve is supposed to pass through *most* of the points, then it resembles problems from coding theory. Here, we consider a problem that unifies the above two instances over *discrete domains*. For example, given a set of t points, with integral coordinates, we consider the task of finding a polynomial with integer coefficients, so that it is Δ-close to all but an ϵ fraction of the points (if such a polynomial exists), where ϵ need only be less than $1/2$.

Self-Correcting Programs One motivation for this work comes from the area of self-correcting programs introduced independently in [7][21]. For many functions, one can take programs that are known to be correct on *most* inputs and apply a simple transformation to produce a program that is correct with high probability on *each* input. But, how bad can a program be, and still allow for such a transformation? There is previous work addressing this question when the functions in question are polynomials (see for example [21],[10],[11]). The results we achieve apply to polynomials as well as to more general forms of algebraic functions, and subsume the best previously known results.

One particular situation where this is useful is in the computation of the permanent of a matrix, over a finite field. Results of Cai and Hemachandra ([8]), when used in combination with our results, imply that if there is an efficient program which computes the permanent correctly on a non-negligible fraction of the input and computes one of a small number of other algebraic functions on the rest of the inputs, then the permanent can be computed efficiently everywhere.

Factoring Bivariate Polynomials In [4] Berlekamp gave a randomized polynomial time algorithm for factoring univariate polynomials over finite fields. In [18] and [13] it is shown that the problem of bivariate factoring can also be solved in polynomial time by reduction to univariate factoring, using fairly deep methods from algebra. Our techniques give a simple method to reduce the problem of factoring bivariate polynomials to that of factoring univariate polynomials over finite fields in the special case when the bivariate polynomial splits into factors which are monic and of constant degree in one of the variables. Though the results are not new, nor as strong as existing results, the methods are much simpler than those used to obtain the previously known results.

The k-Algebraic Black Box Model

In most of the above situations, the difficulty is in *grouping* the sample points, to determine which sample points are related - i.e. come from the same algebraic function. In order to abstract this, we introduce a black box "reconstruction" problem. The problem may be stated as follows.

> Given a black box B, such that the input/output pairs (x, y) of the black box always satisfy a relationship of the form $Q(x, y) = 0$, where Q is a bivariate polynomial of total degree k, find an algebraic function[2] which describes a relationship between the input and the output on a non-trivial fraction of the points.

We call this new model of a black box a **k-algebraic black box**. This model captures situations where on any given input, the black box may output one or more of the several algebraic functions it represents. As a particular example, $Q(x, y) = \Pi_{1 \leq i \leq l}(y - f_i(x))$ would mean that the black box chooses to output one (or more) of l different functions $f_1(x), \ldots, f_l(x)$ at every input x (though it is not specified which ones). This is a generalization of the black box model used in [3],[15],[16],[28] and [29], where on input x, the black box outputs $f(x)$ for f polynomial or rational function. While the target we set for our analysis is that we recover at least one function f_i such that many of the y's satisfy $y = f_i(x)$, we can iterate this process, after stripping off points from f_i, to extract the remaining functions as well.

The notion naturally extends to multivariate functions, and we describe it in terms of polynomials and rational functions. In the case of multivariate functions the description of the target polynomial or rational function might be very large. In order to prevent this from requiring too much running time (to produce the output function), we instead reconstruct the function implicitly by giving a *mechanism* to compute it: we give an efficient algorithm, which is allowed to make queries to the black box. Thus our problem here is defined as follows:

> Given a black box B, such that the input $x_1, \ldots, x_n$ and the output y of the black box always satisfy a relationship of the form $Q(x_1, \ldots, x_n, y) = 0$, where Q is an $(n+1)$-variate polynomial of total degree k, find, or provide a mechanism to compute a polynomial or rational function which describes a relationship between the input and the output on a non-trivial fraction of the points.

In addition, for both the univariate and the multivariate cases, we consider extensions to noisy situations: **ϵ-noisy k-algebraic black boxes**, where the black box is allowed to output arbitrary answers on an ϵ fraction of the inputs.

In the case that *all* the l functions, $f_1, \ldots, f_l$, represented by the black box are polynomials of degree at most d, we call the black boxes (l, d)-**polynomial black boxes**, or **ϵ-noisy (l, d)-polynomial black boxes**.

Previous Work and Our Results

The setting where the black box represents a **single** polynomial or rational function, without noise, is the classic interpolation problem and is well studied. Efficient algorithms for sparse multivariate polynomial interpolation are given in [28], [16], [3] and [29], and for sparse rational functions in [15].

The case where the black box represents a single function with some noise has also been studied previously. In [5] it is shown how to reconstruct a univariate polynomial from a $(\frac{1}{2} - \delta)$-noisy 1-polynomial black box and in [10] and [11] it shown how to do the same for multivariate polynomials. Reconstructing functions from a black box representing more than one function seems to be a previously unexplored subject.

Our algorithmic results include the following: We give an efficient algorithm for explicitly reconstructing univariate polynomials and rational functions, over finite fields, $\mathcal{Z}$ and $\mathcal{Q}$, from an ϵ-noisy k-algebraic black box. The only constraint imposed on ϵ is that our algorithm is only guaranteed to find an f_i if the fraction of inputs for which the black box answers according to f_i is more than ϵ. This result is described in Theorem 2 for the case of reconstructing a polynomial, but applies essentially as it is to the case of extracting a rational function. The result can also be extended to the case of other algebraic functions (e.g. $y = \sqrt{f/g}$, with f and g polynomials), while still tolerating a non-negligible fraction of noise[3].

[2] We use the words "algebraic function" to describe either polynomials or rational functions. Both the notion and our techniques of reconstructing extend to the more general case of algebraic relations where the (x, y) pairs satisfy an *irreducible* polynomial identity.

[3] The fraction of points from the good curve must outweigh the fraction of noise by a multiplicative factor of d where d is the

To reconstruct a polynomial, we first construct a bivariate polynomial $Q'(x, y)$ which is zero at **all** the sample points and then use bivariate polynomial factorization to find a factor of the form $(y - f(x))$. If it exists, $f(x)$ then becomes our candidate for output. We then show that if the number of points on f is large in the sample we see, then $y - f(x)$ has to be a factor of any Q' which all the sample points satisfy.

For the general problem of implicitly reconstructing multivariate polynomials, we show how to reconstruct multivariate polynomials (or rational functions) over sufficiently large finite fields, under the same restrictions on noise as in the univariate case. Our technique here is to reduce the multivariate reconstruction problem into a univariate reconstruction problem using a careful sampling technique. We then use the univariate technique described earlier to solve the new univariate problem.

A note about the specific representation of the output in the case of multivariate polynomials: The fact that we only reconstruct the output implicitly should not be considered a weakness, but rather a strength. In the particular case that the target function is a sparse multivariate polynomial, this allows for the reconstruction of a small set of explicitly represented multivariate polynomials, by the interpolation mechanisms of [16],[3],[28] or [15]. On the other hand, by using the techniques of [19] we can also continue to manipulate the black boxes as they are for whatever purposes[4].

Organization The rest of this paper is organized as follows. In Section 2, we describe our results for univariate polynomials. These results, although presented in terms of polynomials, can be applied directly to rational functions and other algebraic functions. In Section 3, we show how to extend our results to multivariate functions. Here too, the results are presented in terms of polynomials. Finally, in Section 4, we describe in more detail some of the applications of our work.

maximum degree of y in the algebraic relation relating x and y - $Q(x, y)$.

[4]The mechanism of manipulating multivariate polynomials and rational functions represented by black boxes was proposed by Kaltofen and Trager in [19]. They show that it is possible to factor and compute g.c.d.'s for polynomials given by such a representation and to separate the numerator from the denominator of rational functions given by such a representation.

2 Univariate Black Boxes

In this section, we consider the problem of explicitly reconstructing the univariate polynomials $f_1, \ldots, f_k$, each of degree at most d, from a (k, d)-polynomial black box for them. As mentioned earlier, these results apply to rational functions and extend to algebraic functions.

2.1 An Intermediate Model

As an intermediate step towards solution, we assume that the black box outputs *all* of $f_1(x), \ldots, f_k(x)$ on any input x. These are output in arbitrary order, which is not necessarily the same for each x. We further assume that there are no errors in the output. We reduce the problem of extracting the polynomials to that of bivariate polynomial factorization.

If on x the black box outputs $\{y_1, \ldots, y_k\}$, we know that $\forall i,\ 1 \leq i \leq k,\ \exists j,\ 1 \leq j \leq k$ such that $y_i = f_j(x)$. Therefore, one relation satisfied by the input/output of the black box is:

$$\forall (x, y_1, \ldots, y_k)\ \ \Sigma_i \Pi_j\ (y_i - f_j(x)) = 0$$

We can construct a related polynomial which will enable us to recover the f_j's.

Consider the functions $\sigma_i : F \mapsto F, i \in [k]$ defined as

$$\sigma_i(x) \equiv \sum_{S \subset [k], |S| = i} \prod_{j \in S} f_j(x)$$

(these are the *primitive symmetric* functions of $f_1, \ldots, f_k$).

Observe that $\sigma_i(x)$ can be evaluated at any input x using the given (k, d)-polynomial black box (in $O(k \log k)$ time). Observe further that σ_i is a polynomial of degree at most id. Hence evaluating it at $id + 1$ points suffices to find all the coefficients of this polynomial (if the black box outputs every $f_i(x)$ for every x).

Now to find the functions f_j from the functions σ_i, we construct the following polynomial, in x and a new indeterminate y:

$$Q(x, y) = \prod_{j=1}^{k} (y + f_i(x))$$

Observe that Q can also be written as

$$Q(x, y) = y^k + \sigma_1(x) y^{k-1} + \cdots + \sigma_k(x)$$

Thus we can describe Q explicitly (in terms of its coefficients). To recover the f_j's now all we have to do is find the factors of the bivariate polynomial Q. Bivariate factorization can be done efficiently over the rationals ([13], [18], [22]), and can be done efficiently (probabilistically) over finite fields [12],[18].

We have shown the following:

Theorem 1 *Let $f_1, \ldots, f_k$ be degree d polynomials over $\mathcal{Q}$, $\mathcal{Z}$ or a finite field. Given a black box which on input x outputs $\{f_1(x), \ldots, f_k(x)\}$ in arbitrary order, we can reconstruct all the polynomials that it computes with $O(kd+1)$ queries.*

2.2 Noisy (k,d)-Polynomial Black Boxes

We build on the methods of the previous section to reconstruct information from a (k,d)-polynomial black box which outputs the value of *one* of k univariate polynomials $f_1, \ldots, f_k$, on every input. Our method extends immediately to the case where the black box sometimes outputs a value corresponding to *none* of the f_i's, an ϵ-noisy (k,d)-polynomial black box, and we present this result directly. The technique used to achieve this also builds on a procedure of Berlekamp and Welch [5], which (in our notation) reconstructs a polynomial from a $(\frac{1}{2}-\delta)$-noisy $(1,d)$-polynomial black box, where δ can be arbitrarily small.

Given an ϵ-noisy (k,d)-polynomial black box B for the polynomials $f_1, \ldots, f_k$, each of degree at most d, over a finite field F, let p_i be defined as

$$p_i \equiv \Pr_{x \in F}[B \text{ outputs } f_i(x) \text{ on input } x].$$

We show that with high probability, we can reconstruct a polynomial f_i from the ϵ-noisy black box B, in time polynomial in k, d and $\frac{1}{p_i - \epsilon}$ provided $p_i > \epsilon$.

Theorem 2 *Let B be an ϵ-noisy (k,d)-polynomial black box representing the polynomials $f_1, \ldots, f_k$ in x over a finite field F ($|F| > O(\max\{k^2d^2, \frac{1}{(p_1-\epsilon)^2}\})$). If $p_1 > \epsilon$ then (with high probability) a set of at most k polynomials of degree d can be reconstructed from B, such that f_1 is amongst them, with $O(\max\{k^2d^2, \frac{1}{(p_1-\epsilon)^2}\})$ queries.*

Our problem of reconstructing $f_1, \ldots, f_k$ reduces to the following problem:

Problem 1
Given: *t pairs of points $\{(x_1,y_1), \ldots, (x_t,y_t)\}$, such that there exist polynomials $f_1, \ldots, f_k$ satisfying*
$\forall\, i$ $deg(f_i) \le d$, and for all but l values of $j \in [t]$, $\exists\, i \in [k]$ s.t. $f_i(x_j) = y_j$.
(l is the number of noise points in our sample.)

Problem: *Find $f_1, \ldots, f_k$.*

Let S_i be the subset of the indices $j \in [t]$ such that $y_j = f_i(x_j)$ and let E be the set of indices of the error points, i.e., $E = [t] \setminus (\cup_{i=1}^{k} S_i)$. (By our definition $|E| = l$.)

For every index $j \in S_i$, we have $y_j - f_i(x_j) = 0$. Thus for indices $j \notin E$ we have

$$(y_j - f_1(x_j)) * \cdots * (y_j - f_k(x_j)) = 0$$

We can construct a polynomial W, of degree at most l, such that $W \not\equiv 0$, but for all indices $j \in E$, $W(x_j) = 0$. W is simply the polynomial $W(x) = \prod_{j \in E}(x - x_j)$.

Thus we get

$$\forall j \in [t], \quad W(x_j) * (y_j - f_1(x_j)) * \cdots * (y_j - f_k(x_j)) = 0$$

Expanding the product above to represent it as a polynomial in y, we see that there are polynomials in x, $A_0, \ldots, A_k$, $deg(A_m) \le l + m * d$, $A_0 \not\equiv 0$, s.t.

$$\forall\, j \in [t], \quad A_0(x_j) * y_j^k + A_1(x_j) * y_j^{k-1} + \cdots + A_k(x_j) = 0$$

We now consider the following problem:

Problem 2
Given: *t pairs of points $\{(x_1,y_1), \ldots, (x_t,y_t)\}$, and given d, and l, such that there exist polynomials $A_0, \ldots, A_k$, $A_0 \not\equiv 0$, $deg(A_m) \le l + m * d$, and*

$$\forall\, j \in [t], \quad \sum_{m=0}^{k} A_m(x_j) * y_j^{k-m} = 0$$

Problem: *Find $A_0, \ldots, A_k$.*

We can obtain a solution to Problem 2 by substituting unknowns for the coefficients of the A_m's and solving the linear system of equations that is obtained by the constraints of the problem statement above. If the solution to Problem 2 were known to be unique, then by factoring the bivariate polynomial $Q(x,y)$, defined as

$$Q(x,y) \equiv \sum_{m=0}^{k} A_m(x) * y^{k-m} \tag{1}$$

we could get a solution to problem 1. (Notice that $Q(x,y)$ is effectively the same here as in Section 2.1 except that it is multiplied by a factor of $W(x)$ here.)

Unfortunately, the solution to Problem 2 might not be unique and the particular solution to Problem 2 that we may end up finding could be one that will not yield a solution to Problem 1. The following lemma guarantees that under certain constraints on the sizes of the sets $S_1, \ldots, S_k$, the solution to Problem 2 satisfies invariants which guarantee a unique solution to Problem 1.

Lemma 1 *If* $|S_i| > kd + l$ *then* $(y - f_i(x)) \mid Q(x, y)$

Proof: Consider the univariate polynomial $Q_i(x) \equiv Q(x, f_i(x))$.

For all indices $j \in S_i$ we have that $Q_i(x_j) = 0$. But $Q_i(x)$ is a polynomial of degree at most $kd + l$ in x and hence if it is zero at $|S_i| > kd + l$ places, then it must be identically zero, implying that $(y - f_i(x)) | Q(x, y)$ □

The lemma above guarantees that under certain circumstances, the factors of $Q(x, y)$ do give us useful information about the f_i's, leading us to the proof of the theorem:

Proof: [of Theorem 2] We sample from the black box $t = O(\max\{k^2 d^2, \frac{1}{(p_1 - \epsilon)^2}\})$ times and then solve for the polynomial Q as defined in Equation 1. We find all the factors of Q that are linear and monic factors in y and output the set of polynomials $g(x)$ such that $(y - g(x)) | Q(x, y)$.

By Lemma 1 the output set will include f_1 if $|S_1| > kd + |E|$. By using Chernoff bounds we can show that, if the x_j's are picked independently and at random then

$$\Pr\left[(|S_1| < p_1 t - c\sqrt{t}) \text{ or } |E| > \epsilon t + c\sqrt{t}\right] < 2e^{-c^2/2}$$

Thus if $t = \Theta(\max\{k^2 d^2, \frac{1}{(p_1 - \epsilon)^2}\})$, we have that with high probability $|S_1| > kd + |E|$ and hence our algorithm performs correctly. □

3 Multivariate Black Boxes

In this section, we extend Theorem 2 to multivariate polynomial black boxes over finite fields. Once again, we note that this extends to more general cases. The methods of Section 2 do not seem to extend directly to the general multivariate case. This is due to the possibly large *explicit* representation of the function extracted from the black box, which makes it inefficient to work with. Instead, we use techniques of pairwise independent sampling to reduce the problem to a univariate situation and then apply Theorem 2 to the new univariate problem.

Theorem 3 *Given an ϵ-noisy (k, d)-polynomial black box in n variables $x_1, \ldots, x_n$ over a sufficiently large finite field F ($|F| > \max\{4k^2 d^2, \frac{1}{(p-\epsilon)^2}\}$) a black box representation of a polynomial f, which describes the output of the black box on a fraction p ($p > \epsilon$) of the input space, can be constructed in time polynomial in k, d, n and $\frac{1}{p-\epsilon}$, if such a polynomial exists.*

Proof (sketch): Fix any candidate polynomial f which describes the output of the black box on a fraction p of the input space. We show how to reconstruct a small set of polynomials which includes f.

We use the techniques of [1] and [11] to probabilistically construct a small subdomain D of F^n, parameterized by x, such that the following properties hold:

1. D contains any two given points $\hat{a}$ and $\hat{b}$. (In particular, we can ensure $D(0) = \hat{a}$ and $D(1) = \hat{b}$.)
2. Over the domain D, the polynomial $Q(x_1, \ldots, x_n, y)$ can be expressed as a bivariate polynomial in x, y i.e., the function $Q^D(x, y) = Q(D(x), y)$ is a bivariate polynomial.
3. D resembles a randomly and independently chosen sample of F^n of size $|D|$. In particular, with high probability, the fraction of points **from** D where the black box responds with $f(x_1, \ldots, x_n)$, is very close to the fraction of points **from** F^n where the black box responds with f.

D can be constructed by substituting random cubic equations in x for each variable, x_i. The construction works for large finite fields. The details of such constructions are standard and omitted here. A description of a similar construction appears for example in [11].

Claim 2 *For any two points $\hat{a}, \hat{b} \in F^n$, given a domain D containing $\hat{a}$ and $\hat{b}$ with properties (2) and (3) listed above, a set $\{(f_1(\hat{a}), f_1(\hat{b})), \ldots, (f_l(\hat{a}), f_l(\hat{b}))\}$ of values can be reconstructed efficiently such that for some i, $1 \leq i \leq l$, $f_i(\hat{a}) = f(\hat{a})$ and $f_i(\hat{b}) = f(\hat{b})$.*

Proof: Reconstruct l ($l \leq k$) univariate polynomials $f_1^D, \ldots, f_l^D$, in x, that represent all the candidates for f restricted to the domain D. Due to Property (3) and Theorem 2, this is possible in time polynomial in k, d and $\frac{1}{p-\epsilon}$. Evaluating these l polynomials at $D(0)$ and $D(1)$ gives the desired set of values. □

We are not quite done yet because we wish to somehow select the correct value from this set so as to always output the value of the polynomial f. We would like a method to consistently order the values $\{(f_1(\hat{a}), f_1(\hat{b})), \ldots, (f_l(\hat{a}), f_l(\hat{b}))\}$ so that the ith value in

the ordering always comes from f_i. In order to achieve a global ordering of this form we pick a reference point $\hat{r}$, such that $f_i(\hat{r}) \neq f_j(\hat{r})$ if $i \neq j$. The following definition and claim show that such a point exists.

DEFINITION 3.1 *For a multivariate polynomial $Q(x_1, \ldots, x_n, y)$ of degree k in y, and total degree kd, with no repeated factors, a* **reference point** *is a point $\hat{r} =< r_1, \ldots, r_n >$ such that the polynomial $f_{\hat{r}}(y)$ given by Q restricted to $\hat{x} = \hat{r}$ has no repeated factors*[5].

Claim 3 *A random point in F^n is a reference point with probability $\geq 1/2$.*

Proof (sketch): Let $\Delta(x_1, \ldots, x_n)$ be the discriminant (see [26], for instance, for a definition of discriminant) of the polynomial $Q(x_1, \ldots, x_n, y)$ when viewed as a polynomial in y with coefficients that are degree kd polynomials in $x_1, \ldots, x_n$. Observe that the discriminant satisfies the following properties:

- Δ is not identically zero, because Q has no repeated factors.
- Δ is a polynomial in $x_1, \ldots, x_n$ of total degree at most $2k^2d$ (since it is a polynomial in the coefficients of y of total degree at most $2k$).

Thus if $|F| \geq 4k^2d$, then

$$Pr_{\hat{r} \in_R F^n}[\Delta(\hat{r}) \neq 0] \geq \frac{1}{2}$$

But this implies that the polynomial $f_{\hat{r}}(y) = Q(\hat{r}, y)$ has a non-zero discriminant, implying $f_{\hat{r}}(y)$ has no repeated factors, and $\hat{r}$ is a reference point. □

The values of the k polynomials on $\hat{r}$, $\{s_1, \ldots, s_k\}$, can now be used to fix a "global" ordering of the candidate polynomials $f_1, \ldots, f_k$ as follows: Fix an arbitrary ordering $s_1, \ldots, s_k$ and let f_i be the (unique) polynomial that evaluates to s_i at $\hat{r}$. Now, on input $\hat{b} \in F^n$, the black box for f_i does the following:

- Construct a domain D, such that $D(0) = \hat{r}$ and $D(1) = \hat{b}$ and properties (2) and (3) hold.
- Use the univariate reconstruction technique to reconstruct the univariate polynomials $f_1^D, \ldots, f_k^D$.
- Find j such that $f_j^D(0) = s_i$.
- Output $f_j^D(1)$.

□

[5] Note that if $Q(\hat{x}, y) = \Pi_{i=1}^{k}(y - f_i(x))$, this is equivalent to the condition that $f_i(\hat{r}) \neq f_j(\hat{r})$ when $i \neq j$.

4 Applications

In this section, we describe the application of our techniques to some of the areas mentioned in the introduction. The other applications are more straightforward, and we will give the complete details in a later version.

4.1 (ϵ, Δ) Interpolation Problems over Discrete Domains

Consider the task of fitting a low degree polynomial on a given set of points, so that the majority of the points are "close" to the polynomial, while some fraction of the points can be very far away. We look at such computations over discrete domains. For example, over Z_p (or over the integers) the problem can be formulated as:

Problem 3
Given: *t pairs of points, $\{(x_1, y_1), \ldots, (x_t, y_t)\}$, and ϵ, such that there exists a polynomial f, of degree at most d, so that*

for all but ϵt values of j in $[t]$

$$\exists i \in [-\Delta, \Delta] \text{ s.t. } y_j = f(x_j) + i$$

Problem: *Find f.*

This is exactly a $(2\Delta + 1, d)$-polynomial black box reconstruction problem and the reconstruction procedure of Section 2.2 can be used to find one of the polynomials $f_i(x) \equiv f(x) + i$ (or a small set of polynomials which includes f).

We know that there exists a bivariate polynomial $Q(x, y)$ such that $Q(x_i, y_i) = 0$ for all $i = 1, \ldots, t$. This is the polynomial: $Q(x, y) = W(x) \cdot \Pi_i (y - f_i(x))$, where f_i is the polynomial given by $f_i(x) = f(x) + i$, $i \in [-\Delta, \Delta]$, and $W(x)$ is defined by all the points where the y coordinate is more than Δ away from $f(x)$ (as in Section 2.2), so that $deg(W) \leq \epsilon t$.

From Theorem 2 we get the following claim:

Claim 4 *If there exists an i such that the fraction of points for which $y = f_i(x)$ is bigger than ϵ, then we can find f_i.*

A small set of polynomials that is guaranteed to contain f, is $\{f_i + j \ : \ j \in [-\Delta, \Delta]\}$ where the f_i's are the outputs of the reconstruction algorithm.

We want to be able to handle situations where ϵ is arbitrarily close to $1/2$. Using our mechanism directly means that the ϵ that our algorithm can handle depends on Δ. This is because there must be an $i \in [-\Delta, \Delta]$ such that more than ϵt of the points are on $f(x) + i$. An important point to note is that we can artificially decrease the influence of the *bad* points. This is due to the role of these points in defining $Q(x, y)$.

To do this, we look at the following set of points: $\{(x_j^i, y_j^i)\}_{j=1,i=0}^{t,\ 2\Delta}$, where $x_j^i = x_j$ and $y_j^i = y_j - \Delta + i$. (From each point in the original sample, we generate 2Δ additional points, by adding and subtracting up to Δ to the y coordinate of each point.)

Observe that the following conditions hold for the $(2\Delta+1)t$ points so constructed:

- There exists a polynomial $Q(x, y)$ of degree at most $\epsilon t + (4\Delta+1)d$ such that $Q(x, y) = 0$ for all the points. This is the polynomial: $Q(x, y) = W(x) \cdot \Pi_{i=0}^{4\Delta} (y - f_i(x))$, where $W(x)$ is as before, and $f_i(x) = f(x) + i - 2\Delta$.
- At least $(1 - \epsilon)t$ of the points satisfy $y = f(x)$. This is because for every point in the original (x_j, y_j) such that y_j is within Δ of $f(x_j)$ (and there were $(1 - \epsilon)t$ such points), one of the new (x_j^i, y_j^i) pairs satisfies $y_j^i = f(x_j^i)$.

Claim 5 *If $\epsilon < \frac{1}{2}$ then we can find all functions f such that f is Δ-close to all but an ϵ fraction of the points, provided $t > \frac{(4\Delta+1)d}{1-2\epsilon}$.*

Proof: Find a polynomial $Q'(x, y)$ such that $Q'(x, y) = 0$ for all the points and degree of Q' is at most $\epsilon t + (4\Delta+1)d$. If $\epsilon t + (4\Delta+1)d < (1-\epsilon)t$ then by Lemma 1 we know that for every candidate function f which forms an (ϵ, Δ) fit on the given points, $(y - f(x))$ divides Q'. Thus factoring Q' will give us all the candidates. □

4.2 Reducing Bivariate Factoring to Univariate Factoring

In Section 2.1, we saw how to reduce the problem of reconstructing polynomial black boxes to the problem of factoring bivariate polynomials. In the specific case of univariate polynomial black boxes, we can also reduce the reconstruction problem to that of factoring univariate polynomials to their irreducible factors. As an interesting consequence, we find a simple way of showing that over finite fields, factoring special bivariate polynomials is reducible to factoring univariate polynomials. There are known methods to reduce factoring of bivariate polynomials to that of univariate polynomials [18], however this result is interesting in the sense that the reduction, as well as the proof of its correctness, are extremely simple and use very basic algebraic tools.

Suppose, as in Section 2.1, we want to know $f_1, \ldots, f_k$, univariate polynomials, each of degree at most d. However, all we have is a black box - B, that at any given point - x - outputs the **unordered** set $\{f_i(x)\}$. Sampling from the black box, and interpolating, we can find the polynomial $t(x) = \prod_{i=1}^{k} f_i(x)$ explicitly (in terms of its coefficients). If somehow we could guarantee that at least one of the f_i's is irreducible, we could factor t to find f_i. Such a guarantee is not available, but we simulate it via randomization.

Let $\alpha(x) \in F[x]$ be a random degree d polynomial. We can convert the given set of sample points so that on each input x we have the (still unordered) set $\{g_1(x), \ldots, g_k(x) \ : \ g_i(x) = f_i(x) + \alpha(x)\}$. Each of the polynomials g_i is a random degree d polynomial (but they are not necessarily independent). We then use the fact that random polynomials over finite fields have a reasonable chance of being irreducible.

Lemma 6 ([20], p.84) *The probability $P_q(d)$ that a random polynomial, of degree d, is irreducible over F_q, is at least $\frac{1}{d}(1 - \frac{1}{q-1})$.*

We can thus interpolate (after sampling at enough - $kd + 1$ - points) and explicitly compute $t'(x) = \prod_{i=1}^{k} g_i(x)$. We factor t' into irreducible factors $r_1 \cdot \cdots \cdot r_l$. For each factor r_j of t', we verify whether or not $r_j - \alpha$ is a candidate for one of the f_i's by checking that it evaluates to one of the outputs of the black box B on all the sampled points. By Lemma 6 we know that with non-negligible probability g_i is irreducible and if this happens, we find g_i as one of the factors of t' (one of the r_j's). Subtracting α from g_i gives us f_i, which will pass the candidacy verification.

Lemma 7 *If a degree d polynomial p agrees with one of the outputs of the black box on $kd + 1$ different x's, then p agrees with one of the outputs of the black box on all x's.*

Proof: If p agrees with one of the outputs of the black box on $kd + 1$ different x's, then by the pigeonhole principle there is a polynomial f_i which agrees with p on at least $d + 1$ different x's. Thus $p \equiv f_i$. □

Thus, no r_j which is not equal to one of the g_i's will pass the candidacy verification. By repeating this procedure enough

times and outputting all the candidates, we can reconstruct all the polynomials $\{f_1, \ldots, f_k\}$. Straightforward analysis shows that the expected number of times that we need to repeat the process (choose random α) is $O(\ k/P_q(d)\ )$ to be able to reconstruct all of the f_i's. Refining the analysis, we can show that $O(\ \ln k/P_q(d)\ )$ times suffice.

From the above, we get the following algorithm for finding the monic linear factors of a bivariate polynomial $Q(x, y)$.

```
program Simple Factor
repeat O( ln k/P_q(d) ) times
   pick a random degree d polynomial
         α(x) over F
   factor Q(x, α(x))
   for every factor g(x) of Q(x, α(x))
      if (y + g(x) − α(x)) divides Q(x, y)
         output (y + g(x) − α(x))
end
```

Claim 8 *Given a bivariate polynomial $Q(x, y)$, over a finite field F, of total degree at most $k\,d$, the algorithm* **Simple Factor** *finds all the linear and monic factors of $Q(x, y)$.*

We next extend this mechanism, and apply the reconstruction mechanism of Section 2.2 to the problem of finding the factors of $Q(x, y)$ which are monic and of constant degree in y. Our mechanism tries to isolate some factor $A(x, y)$ of $Q(x, y)$ of the form

$$A(x, y) = y^c + a_1(x)y^{c-1} + \cdots + a_c(x)$$

where the a_i's are polynomials in x of degree at most d (and c is a constant).

For each $i \in [c]$ we construct a program P_i for a_i, such that the output of this program on any x is from a small set of polynomials, and is guaranteed to contain $a_i(x)$. This program can be thought of as a black box for a_i. We then use our reconstruction procedure (Theorem 2) to produce, for each $i \in [c]$, a small list of polynomials which contains a_i. This, in turn, gives a small set of polynomials in x and y which contains $A(x, y)$. $A(x, y)$ can be isolated from this set by exhaustive search.

The program P_i for a_i works as follows on input x_1:

- P_i constructs the polynomial $Q_{x_1}(y) \equiv Q(x_1, y)$ (which is a polynomial in y) and factors Q_{x_1}.
- Let S be the set of factors of Q_{x_1}. (S contains polynomials in y.)
- Let S^c be the set of polynomials of degree c obtained by taking products of polynomials in S.
- P_i picks a random polynomial f in S^c and outputs the coefficient of y^i in f.

To see that P_i computes a_i on some inputs, observe that $A_{x_1}(y) = A(x_1, y)$ divides Q_{x_1} and hence appears in the set S^c (which is the set of all degree c factors of Q_{x_1}). Thus with non-negligible probability, when P_i picks a random element of S^c, it is likely to be $A_{x_1}(y)$ and hence when P_i outputs the coefficient of y^i from this polynomial, it outputs $a_i(x_1)$ correctly.

On the remaining inputs, P_i's output is in error, but is the value of some polynomial, so that the value output in this case is obtained from an algebraic combination of elements y_i which satisfy $Q(x_1, y_i) = 0$, and this corresponds to situations for which we can solve.

Acknowledgments

We are very grateful to Avi Wigderson for asking a question that started us down this line of research and for helpful discussions. We are very grateful to Umesh Vazirani for his enthusiasm, his valuable opinion and suggestions and the time that he spent with us discussing this work. We thank Ronen Basri, Oded Goldreich and Mike Kearns for their comments on the writeup of this paper. We would like to thank Joel Friedman for technical discussions about questions related to the topics of this paper.

References

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

[2] M. Ben-Or. Probabilistic Algorithms in Finite Fields In *Proc. 22nd IEEE Symposium on Foundations of Computer Science*, pages 394–398, 1981.

[3] M. Ben-Or and P. Tiwari. A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation. In *Proc. 20th STOC*, pages 301-309, 1988.

[4] E. Berlekamp. Factoring Polynomials over Large Finite Fields. *Mathematics of Computation*, page 713, vol. 24, no. 111, 1970.

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

[6] A. Blum and P. Chalasani. Learning Switching Concepts. In *Proc. 5th Annual Workshop on Computational Learning Theory*, pages 231–242, 1992.

[7] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. In *Proc. 22nd ACM Symposium on Theory of Computing*, 1990.

[8] J.Y. Cai and L. Hemachandra. A note on enumerative counting. Information Processing Letters 38 (1991) pp. 215-219.

[9] Canny J., 1983. Finding edges and lines in images. *M.I.T., Arificial Intelligence Laboratory Report*, AI-TR-720.

[10] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan and A. Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In *Proc. 23rd ACM Symposium on Theory of Computing*, 1991.

[11] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. To appear in *Information Processing Letters*.

[12] D. Grigoriev. Factorization of Polynomials over a Finite Field and the Solution of Systems of Algebraic Equations. Translated from Zapiski Nauchnykh Seminarov Lenningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR, Vol. 137, pp. 20-79, 1984.

[13] D. Grigoriev and A. Chistov. Fast decomposition of polynomials into irreducible ones and the solution of systems of algebraic equations. Soviet Math. Doklady , 29 (1984) 380–383.

[14] D. Grigoriev and M. Karpinski. Algorithms for Sparse Rational Interpolation. ICSI Tech. Report, TR-91-011, January 1991.

[15] D. Grigoriev, M. Karpinski, M.F. Singer. Interpolation of Sparse Rational Functions Without Knowing Bounds on Exponents. Institut für Informatik der Universität Bonn, Report No. 8539-CS, Nov. 1989.

[16] D. Grigoriev, M. Karpinski, M.F. Singer. Fast Parallel Algorithms for Sparse Multivariate Polynomial Interpolation over Finite Fields. SIAM J. Comput., Vol. 19, No. 6, pp 1059-1063, Dec. 1990.

[17] J. Henderson and R. Quandt. **Microeconomic Theory**, McGraw Hill Book Company, 1958, 1971.

[18] E. Kaltofen. A Polynomial-Time Reduction from Bivariate to Univariate Integral Polynomial Factorization. In *23rd Annual Symposium on Foundations of Computer Science*, pages 57-64, 1982.

[19] E. Kaltofen and B. Trager. Computing with Polynomials Given by Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. In *29th Annual Symposium on Foundations of Computer Science*, pages 296–305, 1988.

[20] R. Lidl and H. Niederreiter. Introduction to finite fields and their applications. Cambridge University Press, 1986

[21] 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.

[22] A. K. Lenstra, H. W. Lenstra and L. Lovasz. Factoring polynomials with rational coefficients. Math. Ann., 261 (1982), 515–534.

[23] Marr D. & Hildreth E., 1980. "Theory of Edge Detection". *Proceeding of the Royal Society London, B207*, pp. 187-217.

[24] R. Rivest. Learning Decision Lists. *Machine Learning*, 2(3), pages 229-246, 1987.

[25] T.J. Rivlin. An Introduction of the Approximation of Functions. Dover Publications, 1969.

[26] Van der Waerden. Algebra, Volume 1. Frederick Ungar Publishing Co., Inc., page 82.

[27] R. J. Walker. Algebraic Curves. Dover Publications, 1962

[28] R.E. Zippel. Probabilistic Algorithms for Sparse Polynomials. In *Proc. EUROSAM '79*, Springer Verlag LNCS, pages 216-226, v. 72, 1979.

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