Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computer and System Sciences 72 (2006) 786–812

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

Extractors from Reed–Muller codes

Amnon Ta-Shma [a,*,1], David Zuckerman [b,2], Shmuel Safra [a]

[a] *Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel*
[b] *Department of Computer Science, University of Texas, Austin, TX 78712, USA*

Received 26 February 2002; received in revised form 6 May 2004

Available online 20 February 2006

Abstract

Finding explicit extractors is an important derandomization goal that has received a lot of attention in the past decade. Previous research has focused on two approaches, one related to hashing and the other to pseudorandom generators. A third view, regarding extractors as good error correcting codes, was noticed before. Yet, researchers had failed to build extractors directly from a good code without using other tools from pseudorandomness. We succeed in constructing an extractor directly from a Reed–Muller code. To do this, we develop a novel proof technique.

Furthermore, our construction is the first to achieve degree close to linear. In contrast, the best previous constructions brought the log of the degree within a constant of optimal, which gives polynomial degree. This improvement is important for certain applications. For example, it was used [E. Mossel, C. Umans, On the complexity of approximating the VC dimension, J. Comput. System Sci. 65 (2002) 660–671] to show that approximating VC dimension to within a factor of $N^{1-\delta}$ is AM-hard for any positive δ.
© 2006 Elsevier Inc. All rights reserved.

* Corresponding author.
E-mail addresses: amnon@post.tau.ac.il (A. Ta-Shma), diz@cs.utexas.edu (D. Zuckerman), safra@post.tau.ac.il (S. Safra).
URLs: http://www.cs.tau.ac.il/~amnon (A. Ta-Shma), http://www.cs.utexas.edu/~diz (D. Zuckerman), http://www.cs.tau.ac.il/~safra (S. Safra).

[1] Some of this work was done while the author was at the University of California at Berkeley, and supported in part by a David and Lucile Packard Fellowship for Science and Engineering and NSF NYI Grant CCR-9457799.

[2] Some of this work was done while the author was on leave at the University of California at Berkeley. Supported in part by a David and Lucile Packard Fellowship for Science and Engineering, NSF Grants CCR-0310960, CCR-9912428, and CCR-9457799, and an Alfred P. Sloan Research Fellowship.

0022-0000/$ – see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2005.05.010

Keywords: Extractors; Reed–Muller codes; Expanders; Pseudorandomness; Derandomization; Inapproximability

1. Introduction

1.1. History and background

Sipser [2] and Santha [3] were the first to realize that extractor-like structures can be used to save on randomness. Sipser and Santha showed the existence of such objects, and left open the problem of explicitly constructing them. True extractors were first defined in [4]:

Definition 1. [4] $E:\{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an *ϵ-extractor* for a class of distributions $\mathbb{X}$ over $\{0,1\}^n$, if for every distribution $X \in \mathbb{X}$ the distribution $E(X,U_t)$ is within statistical distance ϵ from U_m.[3] E is called strong if $U_t \circ E(X,U_t)$ is within ϵ of uniform (here $\circ$ denotes concatenation and U_t refers to the same uniform distribution, i.e., not two independent copies). E is explicit if $E(x,y)$ can be computed in time polynomial in the input length $n+t$. E is a *(k,ϵ)-extractor* if E is an extractor for all distributions with min-entropy k.[4]

Thus, extractors extract the entropy from a defective random source $X \in \mathbb{X}$ using a small number t of additional truly random bits. The goal is to construct extractors for any min-entropy k with t as small as possible and m, the number of output bits, as large as possible.

Building on earlier work of Zuckerman [5,6], Nisan and Zuckerman [4] built an extractor with $t = O(\log^2 n)$ when the entropy of the source k was high, $k = \Omega(n)$. Srinivasan and Zuckerman [7] extended this solution to the case $k = n^{1/2+\epsilon}$ and Ta-Shma [8] further extended it to any entropy k. Also, Ta-Shma was the first to extract all the entropy from the source. Zuckerman [9] showed a construction with $t = O(\log n)$ working for high entropies $k = \Omega(n)$. All of this work used hashing and k-wise independence in various forms.

Departing from previous techniques, Trevisan [10] showed a connection between pseudorandom generators for small circuits and extractors. Trevisan used the Nisan–Wigderson pseudorandom generator [11] to construct a simple and elegant extractor that achieves $t = O(\log n)$ when $k = n^{\Omega(1)}$ (and $t = O(\log^2 n)$ for the general case). Trevisan's work was extended in [12–15] to work for every k with only $t = O(\log n)$ truly random bits. These extensions also made the construction more involved and added to the conceptual complexity of the extractor.

Thus, in the current state of the art, there are two techniques that are used in various forms and combinations and different degrees of complexity. Even after all that work, all

[3] U_t denotes the uniform distribution on t bits, and $E(X,U_t)$ denotes the distribution obtained by evaluating $E(x,y)$ for x chosen according to X and y according to U_t. Also, see Section 2 for the definition of statistical distance, also known as variation distance.

[4] See Section 2 for the definition of min-entropy.

Table 1
Milestones in building explicit extractors

k	t	m	Ref.
$\Omega(n)$	$O(\log n)$	$\Omega(k)$	[9]
any k	$O(\frac{\log^2 n}{\log k})$	$k^{1-\alpha}$	[10]
any k	$O(\log n)$	$k/\log n$	[14]
$k \geqslant \sqrt{n}m \log^2 n$	$\log n + O(\log(\log^* m))$	m	This paper
$k \geqslant n^{1/c}m$	$\log n + O(c^2 \log m)$	m	This paper
$\Omega(n)$	$\log n + O(\log\log n)$	$\Omega(k)$	This paper
any k	$\log n + \Theta(1)$	$k + t - \Theta(1)$	Optimal. [16]

The error ϵ is a constant.

known constructions use $t \geqslant 2\log n$ (and often much more) while the lower bound is only $t = \log n + O(1)$. This progress is summarized in Table 1 for the case of constant error ϵ.

1.2. *The significance of the extractor degree*

Besides their straightforward applications to simulating randomized algorithms using weak sources, extractors have had applications to many areas in derandomization that are seemingly unrelated to weak sources. These include constructing expanders that beat the second eigenvalue method [17], superconcentrators and non-blocking networks [17], sorting and selecting in rounds [17], pseudorandom generators for space-bounded computation [4], unapproximability of CLIQUE [6] and certain Σ_2^P minimization problems [18], time versus space complexities [2], leader election [9,19], another proof that BPP $\subseteq$ PH [20], random sampling using few random bits [9], and error-correcting codes with strong list decoding properties [21]. Håstad [22] uses non-explicit dispersers[5] in his result that CLIQUE is unapproximable to within $n^{1-\alpha}$ for any $\alpha > 0$. The use of non-explicit dispersers make the result depend on the assumption that NP $\neq$ ZPP; a derandomized version would assume that NP $\neq$ P.

In many of these applications, extractors are viewed as highly unbalanced strong expanders. In this view an extractor is a bipartite graph $G = (V, W, E)$ with $V = \{0,1\}^n$, $W = \{0,1\}^m$, and an edge (x, z) exists iff there is some $y \in \{0,1\}^t$ such that $E(x, y) = z$. Thus, the degree of each vertex of V is $T = 2^t$, and the extractor hashes the input $x \in V$ to a random neighbor among its T neighbors in W.

Often this degree T is of more interest than $t = \log T$. For example, in the samplers of [9] the degree is the number of samples; in the simulation of BPP using weak sources [6] the degree is the number of calls to the BPP algorithm; in the extractor codes of [21] T is the length of the code; and in the unapproximability of CLIQUE, the size of the graph is closely related to T.

As stated before, all previous constructions have degree T which is at least poly(n) = poly$(\log|V|)$ while the lower bound (that matches non-explicit constructions) is only $T = \Omega(n) = \Omega(\log|V|)$. Our construction breaks the polynomial degree bound and is the first explicit construction with degree close to linear.

[5] A disperser is a one-sided version of extractor.

Our construction was used by Mossel and Umans [1] to show that it is AM-hard to approximate VC dimension to within a large factor. In fact, Mossel and Umans only need that for all $\delta > 0$ there are explicit dispersers with degree $n^{1+\delta}$ for sources with min-entropy n^{δ}, and we supply an even smaller degree.

Recently, Guruswami [23] used our construction to explicitly build error correcting codes with $1-\epsilon$ list-decoding and close to $\Omega(\epsilon)$ rate. The list decoding algorithm returns sub-exponentially many possible solutions. Guruswami also showed [23, Corollary 3] that extractors with small degree that work for low entropies, lead to close to optimal list-decodable error correcting codes.

Another inapproximability result that requires dispersers of very low degree is Håstad's result [22] that CLIQUE cannot be approximated to within $n^{1-\alpha}$ for any constant $\alpha > 0$, assuming NP $\neq$ ZPP. The unproven assumption could be relaxed to NP $\neq$ P if for any positive constant γ the following disperser could be efficiently constructed. For sources with min-entropy $k = \gamma n$, the output length should be linear in k (and hence n), and the degree should be $O(n/\log(1-\epsilon)^{-1})$. Note that as the error ϵ approaches 1, the degree becomes less than n. This dependence on ϵ is known non-explicitly and matches lower bounds. Very recently such a disperser was announced [24].

1.3. Our construction

Our construction uses error-correcting codes. Codes are known to be related to extractors. Trevisan's extractor can be viewed as first encoding the weak source input $x \in \{0,1\}^n$ with any good binary error correcting code (good here means minimum distance close to half) and then using the truly random string y to select bits from the encoded string using designs. A natural question is whether one can use a specific good code to allow y to be used in a more efficient way. Our extractor construction is the first to do this.

Our good code is a Reed–Muller code concatenated with a good binary code. Specifically, we view the input x from the weak source as defining a degree h multivariate polynomial $f_x : \mathbb{F}^D \to \mathbb{F}$ over some large field $\mathbb{F}$ (h is about $n^{1/D}$, the size of $\mathbb{F}$ is slightly larger than h). We further choose a good binary error correcting code $\mathbf{C}$ to encode field elements. Our code maps x to the sequence of values $\mathbf{C}(f_x(a))$, where a goes over all the elements of $\mathbb{F}^D$.

We now use the truly random string y to choose output bits from the encoding of x. We think of the random string y as indexing an element $a \in \mathbb{F}^D$ and a random position j of the binary code. The ith output bit of the extractor, $E(x;y)_i$, is

$$E(x;y)_i = \mathbf{C}\big(f_x\big(a + (i,0,\ldots,0)\big)\big)_j,$$

i.e., the jth bit of $\mathbf{C}(f_x(a + (i,0,\ldots,0)))$. Thus, the construction can be viewed as using a Reed–Muller concatenated with a good binary code for encoding, and selecting the code evaluation on m consecutive points

$$a + (1,0,\ldots,0),\ a + (2,0,\ldots,0),\ \ldots,\ a + (m,0,\ldots,0)$$

for the output.

Notice the simple way in which we use the random string y to select the output bits. Indeed, this gives an extractor using only $t = \log n + O(\log \frac{m}{\epsilon})$ truly random bits. For simplicity we will focus on the bivariate case, though the multivariate case works as well, and gives different parameters. We prove:

Theorem 2. *For every $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n) \leqslant 1/2$ such that $3m\sqrt{n}\log(\frac{n}{\epsilon}) \leqslant k \leqslant n$, there exists an explicit family of (k, ϵ) strong extractors $E_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ with $t = \log n + O(\log m) + O(\log \frac{1}{\epsilon})$.*

Furthermore, we can reduce the $O(\log m)$ term above to $O(\log(\log^* m))$. To do that we notice that our construction (and also Trevisan's) can be viewed as an efficient reduction from the problem of constructing extractors for general sources, to the problem of constructing extractors for almost block sources (defined in Section 3.1). We then show an efficient construction for almost block sources. We get:

Theorem 3. *For every $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n)$ such that $3m\sqrt{n}\log(n/\epsilon) \leqslant k \leqslant n$ and $m \geqslant \Omega(\log^2(\frac{1}{\epsilon}))$, there exists an explicit family of (k, ϵ) strong extractors $E_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^{\Omega(m)}$ with $t = \log n + O(\log \epsilon^{-1}) + O(\log(\log^* m))$.*

Notice that while we dramatically improve the random bit complexity, both constructions work only for entropies $k = n^{1/2+\gamma}$ and the number of output bits is only k^δ for some $\delta < \gamma$. We can make the construction work for smaller entropies by using multivariate polynomials instead of bivariate polynomials, but then we pay in the number of the output bits and the error. Namely, we prove:

Theorem 4. *For every constant D and $m = m(n) \leqslant (n/\log n)^{1/D}$, there exists an explicit family of (k, ϵ) strong extractors $E_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ with $k = \Omega(m^{D-1} n^{1/D} \log^2 n)$, $t \leqslant \log n + O(D^2 \log m)$, and $\epsilon = 1 - \frac{1}{8D}$.*

We can also extract more output bits in the case where the entropy is large, namely, $k = \Omega(n)$. Formally,

Theorem 5. *For any constant $\delta > 0$, there exists a constant γ and an explicit family of $(\delta n, 1/\log n)$ strong extractors $E : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ with $t = \log n + O(\log\log n)$ and $m = \gamma n$.*

The ideas of this paper were generalized in [25] and later on in [26] to give the best known explicit extractors and pseudo-random generators. We note, however, that both [25] and [26] use $O(\log n)$ truly random bits, rather than $(1 + o(1))\log(n)$ truly random bits as in our construction.

1.4. Our proof technique

At the highest level our proof plan resembles Trevisan's. That is, we assume that the distribution $E(X, U_t)$ is not close to uniform. Also, w.l.o.g., we assume the distribution X is uniformly distributed on its support (see Fact 6). It is convenient to overload notation and use X to denote both the distribution and its support. Yao's lemma gives us a next element

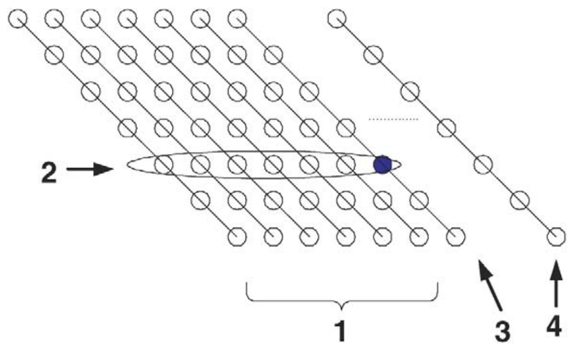

Fig. 1. Learning f_x from $m-1$ lines. We have the following stages: (1) We query the value of f_x on $h-1$ parallel lines. Notice that f_x restricted to a line is a univariate degree h polynomial, so this amounts to $(m-1)h$ point queries. (2) For each point on the next parallel line, we use the next element predictor to predict the value of f_x on it. (3) We find the unique polynomial that has large agreement with our prediction. (4) We keep learning consecutive lines until we recover f_x, hence also x itself.

predictor that on average can learn the value $f_x(a_1+i, a_2)$ from the values $f_x(a_1+1, a_2)$, $\ldots, f_x(a_1+i-1, a_2)$, where $f_x : \mathbb{F}^2 \to \mathbb{F}$ is the bivariate polynomial representing x. We then use the predictor to give a small description of the elements in X. We conclude that if $E(X, U_t)$ is not close to uniform, then X is small. Equivalently, the contra-positive is that if the set X is large (so the distribution X has large min-entropy), then $E(X, U_t)$ is close to uniform.

We now describe how we use the predictor to give a small description of the elements in X. We play a mental game. We pick a random line L and assume someone gives us the correct values of f_x on $m-1$ consecutive parallel left-shifts of L, i.e., $L-(1,0)$ through $L-(m-1,0)$. In other words, each point on L is preceded by $m-1$ points for which we already know the correct value of f_x. Hence we can use the predictor to predict that point, with some moderately good success probability. Overall, the predictor is correct for many points on L.

We now wish to find the value of f_x on the line L. Note that f_x restricted to L is a low-degree polynomial, i.e., a codeword of a Reed–Solomon code. Assume for the moment that our predictor is correct on, say, 90% of the points on L. Then we could find the unique low-degree polynomial which agrees with the predicted values 90% of the time. This is the usual decoding of Reed–Solomon codes.

Proceeding in the same way, we then learn f_x restricted to $L_1 = L + (1,0)$, $L_2 = L + (2,0)$, etc., until we learn enough lines to reconstruct f_x itself. This is illustrated in Fig. 1.

In reality, our predictor is only guaranteed to be correct with probability smaller than half. Hence unique decoding is impossible, and we will have to use "list-decoding" of Reed–Solomon codes to narrow our choices. We will then have to make additional "line queries" to determine the correct value of f_x restricted to the lines.

Playing this mental game, we can prove that for every set X for which $E(X, U_t)$ is not close to uniform, there exists a set of about mh queries (and recall that h is about $\sqrt{n}$ and m is the number of output bits) such that almost every $x \in X$ can be reconstructed given the answers to these queries. Note that h^2 queries can always reconstruct X; our gain is that we can reconstruct the elements of X using only mh instead of h^2 queries. This shows that $|X| \leqslant |\mathbb{F}|^{mh}$, or equivalently that the distribution X has min-entropy at most $mh \log q$. We conclude that if the distribution X has larger min-entropy, than the distribution $E(X, U)$ is close to uniform.

The technique was inspired by work done on list decoding of Reed–Muller codes, and its application to hardness amplification in [27]. Aside from applying an error-correcting technique to a different information theoretic setting, our proof technique has additional ideas. For example, we obtain our savings by learning a line using previously learned lines, and this whole notion of recycling queries makes sense only in our setting and does not appear in previous constructions.

Our construction is the first purely algebraic extractor construction, and the first to rely solely on error-correcting codes. We believe this is a clean and elegant way of constructing extractors.

1.5. Organization of the paper

In Section 2 we give formal definitions, and state the results we use from coding theory. In Section 3 we give a top-down overview of the construction and the proof. The proof is given in two parts: first (in Section 3.2), a reduction to almost block sources (that are defined in Section 3.1), then an extractor for such sources (stated in Section 3.3). In Section 3.4 we put everything together to derive Theorems 2 and 3. In Section 3.5 we explain how to get more output bits and derive Theorem 5. In the later sections we fill the theorems stated in the top-down overview. In Section 4 we show the reduction to almost block sources and in Section 5 we present explicit extractors for almost block sources. Finally, in Section 6 we show the higher-dimensional version of our construction and prove Theorem 4.

2. Preliminaries

Notation. Throughout, $\mathbb{F} = \mathbb{F}_q$ denotes a field of prime size q. As usual, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. For $S \subseteq \mathbb{F}$ and $a \in \mathbb{F}$, $S + a$ denotes the set $\{s + a\colon s \in S\}$. For two sets $S_1, S_2 \subseteq \mathbb{F}$, $S_1 + S_2 = \{s_1 + s_2\colon s_1 \in S_1,\ s_2 \in S_2\}$. All logarithms are to the base 2. We will assume, when needed, that various quantities are integers. It is not hard to check that this has only a negligible effect on our analysis.

We will constantly be discussing probability distributions, the distances between them, and the randomness hidden in them. In this subsection we give the basic definitions used to allow such a discussion.

A probability distribution X over a (finite) space Λ simply assigns to each $a \in \Lambda$ a positive real $X(a) > 0$, with the property that $\sum_{a \in \Lambda} X(a) = 1$. For a subset $S \subseteq \Lambda$ we denote $X(S) = \sum_{a \in S} X(a)$. A distribution is *flat* if it is uniformly distributed over its support. The uniform distribution U on Λ is defined as $U(a) = 1/|\Lambda|$ for all $a \in \Lambda$. U_k denotes the uniform distribution over $\{0, 1\}^k$.

We identify a random variable with the probability distribution it induces. Thus, if X is a random variable, then $x \in X$ denotes picking an element x according to the distribution the random variable X induces. We denote random variables and distributions by capital letters, and use small letters $a, x, z, \ldots$ to denote elements in the probability space.

The *statistical distance* (or *variation distance*) between two distributions D_1 and D_2 on the same space S is $\max_{T \subseteq S} |D_1(T) - D_2(T)| = \frac{1}{2} \sum_{s \in S} |D_1(s) - D_2(s)|$. We say two distributions are ϵ-*close* if their statistical distance is at most ϵ. The *min-entropy* of a distribution D on a probability space S is $\min_{s \in S}\{-\log_2 D(s)\}$.

An element $x \in \Lambda_1 \times \cdots \times \Lambda_m$, for some domains Λ_i, is an m-tuple, $x = x_1 \circ \cdots \circ x_m$. Similarly, a distribution X over Λ^m defines m correlated distributions $X = X_1 \circ \cdots \circ X_m$. For $1 \leqslant i \leqslant j \leqslant m$ let $X_{[i,j]}$ denote the random variable $X_i \circ \cdots \circ X_j$ and similarly for x. Thus, $\Pr(X_{[i,j]} = x_{[i,j]})$ is a shorthand for $\Pr(X_i = x_i \wedge \cdots \wedge X_j = x_j)$.

Extractors. We gave the definition of an extractor in Section 1.1. As we said there, $E:\{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a (k,ϵ) extractor iff it is an ϵ extractor for all distributions with k min-entropy. It is well known that every distribution with k min-entropy, for an integer k, can be expressed as a convex combination of *flat* distributions with k min-entropy. In particular,

Fact 6. E is a (k,ϵ) extractor iff E is an ϵ extractor against all *flat* sources with k min-entropy.

This means that to show that $E:\{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a (k,ϵ) extractor, it suffices to concentrate on *flat* distributions X that are uniformly distributed over some large set X of cardinality at least 2^k. For each such X we need to show that $E(X, U_t)$ is ϵ-close to uniform.

Polynomials. We will use the following lemma due to Sudan.

Lemma 7. [28] *Given m distinct pairs $(x_i, y_i) \in \mathbb{F}^2$, there are less than $2m/a$ degree h polynomials p such that $p(x_i) = y_i$ for at least a values of $i \in [m]$, provided that $a \geqslant \sqrt{2hm}$.*

2.1. Binary codes

Definition 8. A binary code has combinatorial list decoding property α if every Hamming ball of relative radius $1/2 - \alpha$ has $O(1/\alpha^2)$ codewords.

We will use codes from the following code construction due to [29].

Fact 9. [29] There is a polynomial-time (in fact, Logspace) constructible $[n,k]$ code with combinatorial list decoding property α, where $n = O(k/\alpha^4)$.

Simpler and more efficient constructions can be achieved with somewhat worse parameters, e.g., [30,31].

Reed–Muller codes. In an (h, D) Reed–Muller code over $\mathbb{F}_q$ the message specifies a polynomial f in D variables over $\mathbb{F}_q$ of total degree at most h, and the output is all the values of f over $\mathbb{F}_q^D$. Every polynomial in D variables of total degree $\leqslant h$ can be represented by the coefficients of the different monomials $x_1^{i_1} \cdots x_D^{i_D}$ with $i_1 + \cdots + i_D \leqslant h$, and there are exactly $\binom{h+D}{D}$ such monomials. It follows that such a code has length q^D and dimension $\binom{h+D}{D}$.

3. Top–down overview

3.1. Almost block sources

Block sources were defined and studied in many early papers, most notably [4,7,32,33]. We extend this definition to a β-almost block source:

Definition 10. A distribution $W = W_1 \circ \cdots \circ W_b$ is a β-almost $((n_1, k_1), \ldots, (n_b, k_b))$ block source if for every $i = 1, \ldots, b$, W_i is distributed over $\{0, 1\}^{n_i}$ and

$$\Pr_{w_{[1,i-1]} \in W_{[1,i-1]}} \left[H_\infty(W_i \mid W_{[1,i-1]} = w_{[1,i-1]}) < k_i \right] \leqslant \beta.$$

If $\beta = 0$ we say W is an $((n_1, k_1), \ldots, (n_b, k_b))$ block source.

The following lemma shows that a β-almost block source with b blocks is $b\beta$ close to a block source. Notice, that the penalty $b\beta$ depends linearly on the number of blocks, a dependence that is sometimes too costly for us.

Lemma 11. *A β-almost $((n_1, k_1), \ldots, (n_b, k_b))$ block source $Z = Z_1 \circ \cdots \circ Z_b$ is $b\beta$ close to an $((n_1, k_1), \ldots, (n_b, k_b))$ block source Z'.*

Proof. We build a sequence $Z^{b+1}, \ldots, Z^1$ of distributions. We set $Z^{b+1} = Z$ and we define Z' to be Z^1. Suppose we built $Z^{b+1}, \ldots, Z^{i+1}$ so far, we explain how to build Z^i. We call $w_{[1,i-1]}$ a *bad* prefix if $H_\infty(Z_i \mid Z_{[1,i-1]} = w_{[1,i-1]}) < k_i$. The idea is to redistribute the weight of every bad prefix $w_{[1,i-1]}$ uniformly over $\{0, 1\}^{k_i}$. Formally,

- We let Z^i have the same distribution on the first $i - 1$ blocks as Z^{i+1}, i.e., $Z^i_{[1,i-1]} = Z^{i+1}_{[1,i-1]} = Z^{b+1}_{[1,i-1]} = Z_{[1,i-1]}$.
- Now, for every *bad* prefix $w_{[1,i-1]}$ we let $(Z^i_i \mid Z^i_{[1,i-1]} = w_{[1,i-1]})$ be the uniform distribution, and for all other prefixes we let $(Z^i_i \mid Z^i_{[1,i-1]} = w_{[1,i-1]}) = (Z^{i+1}_i \mid Z^{i+1}_{[1,i-1]} = w_{[1,i-1]}) = (Z_i \mid Z_{[1,i-1]} = w_{[1,i-1]})$.
- Finally, for $j > i$, if $\Pr(Z^{i+1}_{[1,j-1]} = w_{[1,j-1]}) > 0$ we let $(Z^i_j \mid Z^i_{[1,j-1]} = w_{[1,j-1]}) = (Z^{i+1}_j \mid Z^{i+1}_{[1,j-1]} = w_{[1,j-1]})$, otherwise we let $(Z^i_j \mid Z^i_{[1,j-1]} = w_{[1,j-1]})$ be the uniform distribution.

By definition, at each stage in the process at most β fraction of the prefixes are bad, and so at most $b\beta$ of the weight is redistributed. Thus, $Z' = Z^1$ is $b\beta$ close to Z.

Also, for any i and any prefix $w_{[1,i]}$, $(Z^1_{i+1} \mid Z^1_{[1,i]} = w_{[1,i]})$ is either uniform or $(Z^{i+1}_{i+1} \mid Z^{i+1}_{[1,i]} = w_{[1,i]})$. In the later case, if the prefix $w_{[1,i]}$ is bad this is the uniform distribution, and otherwise it is $(Z_{i+1} \mid Z_{[1,i]} = w_{[1,i]})$ and has at least k_{i+1} min-entropy (because the prefix $w_{[1,i]}$ is not bad). In either case, $H_\infty(Z'_{i+1} \mid Z'_{[1,i]} = w_{[1,i]}) \geqslant k_{i+1}$.

It follows that Z' is $b\beta$ close to Z and is an $((n_1,k_1),\dots,(n_b,k_b))$ source as desired. □

Nisan and Zuckerman [4] showed a simple technique for constructing efficient extractors for block sources. Suppose we are given an $((n_1,k_1),\dots,(n_b,k_b))$ block source, and b strong extractors $E_1,\dots,E_b$ with $E_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{r_{i-1}-r_i}$. We denote $E'_i(x;y) = y \circ E_i(x;y)$, so $E'_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{r_{i-1}}$. Define $F' : \{0,1\}^{n_1+\dots+n_b} \times \{0,1\}^{r_b} \to \{0,1\}^{r_1}$ by

$$F'(z_1,\dots,z_b;y) \overset{\text{def}}{=} E'_1\bigl(z_1;\dots E'_{b-1}\bigl(z_{b-1};E'_b(z_b;y)\bigr)\dots\bigr).$$

Notice that $F'(x;y) = y \circ F(x;y)$ for some function $F : \{0,1\}^{n_1+\dots+n_b} \times \{0,1\}^{r_b} \to \{0,1\}^{r_1-r_b}$.

The following lemma is implicit in [4] and explicit in [7], though without the mention of strong extractors.

Lemma 12. [4,7] *If each E_i is a (k_i,ϵ_i) strong extractor than F is a strong $\epsilon = \sum_{i=1}^b \epsilon_i$ extractor for $((n_1,k_1),\dots,(n_b,k_b))$ block sources.*

We finally observe that any $((1,k_1),\dots,(1,k_1))$ block source with $2^{-k_1} = 1/2+\alpha$ is $b\alpha$ close to uniform:

Lemma 13. *A $((1,k_1),\dots,(1,k_1))$ block source $Z = Z_1 \circ \dots \circ Z_b$, with $2^{-k_1} = 1/2+\alpha$, is $b\alpha$ close to the uniform distribution.*

Proof. We notice that for every prefix the probability of the next bit being 0 or 1 is at most $1/2+\alpha$. We can again induct from $i=b$ to $i=1$ and redistribute the weight such that it is perfectly uniform, in much the same way as is done in the proof of Lemma 11. The resulting difference is again at most $b\alpha$. □

3.2. A reduction to almost block sources

We now present our reduction from general sources to block sources. Let $\mathbb{F} = \mathbb{F}_q$ be a field of size $q \gg \sqrt{n}$, q prime. The reduction begins by viewing the n-bit input string $x \in \{0,1\}^n$ as a bivariate polynomial $f_x : \mathbb{F}^2 \to \mathbb{F}$ of total degree at most $h-1$. We would like different inputs to map into different bivariate polynomials. Every degree $h-1$ bivariate polynomial f can be specified using $\binom{h+1}{2}$ coefficients from $\mathbb{F}_q$ and so we need $\binom{h+1}{2}\log q \geqslant n$ which sets $h = \Theta(\sqrt{n/\log n})$.

We now present a function $\mathbf{Z} : \{0,1\}^n \times \mathbb{F}^2 \times [\bar{\ell}] \to \{0,1\}^m$. We will later show $\mathbf{Z}$ reduces high entropy sources to almost block sources. We define:

The function $\mathbf{Z}$

Input: $x \in \{0,1\}^n$.
Parameters: $\alpha, \beta > 0$ error parameters. m is a parameter specifying output length.
Setting: Set $h = \lceil 3\sqrt{n/\log n}\rceil$, q the first prime with $q \geqslant \Omega(\frac{h}{\alpha^4\beta^4})$ and $\mathbb{F}$ a field of size q.
Binary code: $\mathbf{C}$ is a linear binary code of dimension $\ell = \log q$, combinatorial list decoding property $\frac{\alpha\beta}{4}$ (see Definition 8) and length $\bar{\ell} = \mathrm{poly}(\alpha^{-1}, \beta^{-1})$ (see Fact 9).
Random coins: $a = (a_1, a_2) \in \mathbb{F}^2$, $j \in [\bar{\ell}]$.
Output: We associate with $x \in \{0,1\}^n$ a function $f_x : \mathbb{F}^2 \to \mathbb{F}$ of total degree at most $h-1$. We define

$$\mathbf{Z}(x; a, j)_i = \mathbf{C}\big(f_x(a_1 + i, a_2)\big)_j.$$

For a subset $X \subseteq \{0,1\}^n$ let $U_t \circ \mathbf{Z}(X, U_t)$ denote the random variable obtained by picking x uniformly at random from X, y uniformly at random from $\mathbb{F}^2 \times [\bar{\ell}]$ and computing $y \circ \mathbf{Z}(x; y)$. We claim:

Theorem 14. *Let n, $\alpha, \beta > 0$ be arbitrary and set h, q and a field $\mathbb{F}$ of size q as above. Let $m \leqslant \sqrt{n}$. For every subset $X \subseteq \{0,1\}^n$ of cardinality at least q^{mh} the distribution*

$$U_t \circ \mathbf{Z}(X, U_t) = U_t \circ \mathbf{Z}_1 \circ \mathbf{Z}_2 \circ \cdots \circ \mathbf{Z}_m$$

is a β-almost $((t,t), (1, f(\alpha)), \ldots, (1, f(\alpha)))$ block source with $2^{-f(\alpha)} = 1/2 + \alpha$. Furthermore $t \leqslant \log n + O(\log(\frac{1}{\alpha\beta}))$ and $\mathbf{Z}$ can be computed in $O(\log q)$ space and $\mathrm{poly}(q) = \mathrm{poly}(n, \frac{1}{\alpha\beta})$ *time.*

We first verify parameters. Since

$$\binom{h+1}{2}\log q \geqslant \frac{h^2}{2}\log h \geqslant \frac{9n}{2\log n}\frac{1}{2}(\log n - \log\log n) \geqslant n$$

two different inputs give rise to two different bivariate polynomials. Next, note that the number of truly random bits $t = \log(q^2) + \log\bar{\ell}$. Using Fact 9, $\bar{\ell} = \frac{\log q}{\mathrm{poly}(\alpha,\beta)}$, so $t = \log(q^2\log q) + O(\log((\alpha\beta)^{-1})) = \log n + O(\log((\alpha\beta)^{-1}))$. The running time of $\mathbf{Z}$ is dominated by the complexity of evaluating $f_x(a)$, which can be done with $O(\log q)$ space and $\mathrm{poly}(q)$ time. We prove the reduction correctness in Section 4.

3.3. An ϵ-almost block extractor

An extractor working on a general distribution over n input bits requires at least $t \geqslant \log n - O(1)$ truly random bits, even when the allowed error ϵ is a constant [16]. In contrast, the following theorem shows that extractors for block sources require only $t = O(1)$ truly random bits and explicitly construct such an extractor.

Theorem 15. *Let $k_1 \leqslant 1$ be a constant. For every $\epsilon = \epsilon(n) \geqslant 0$, there exists a strong extractor $\mathbf{F} : \{0,1\}^m \times \{0,1\}^r \to \{0,1\}^{m'}$ for β-almost $((1,k_1), \ldots, (1,k_1))$ block sources with*

$r = O(\log \epsilon^{-1})$, $m' = \frac{k_1}{2}m - O(\log^4 \frac{1}{\epsilon} + \log^* m \cdot \log \epsilon^{-1})$ *output bits, and* $\epsilon + O(\beta \cdot \log^* m)$ *error.*

We prove Theorem 15 in Section 5.

3.4. Putting it together

Proof of Theorem 2. Suppose $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n)$ are such that $3m\sqrt{n}\log(\frac{n}{\epsilon}) \leqslant k \leqslant n$. Let us set $\alpha = \beta = \frac{\epsilon}{2(m+1)}$ and q and h as in Theorem 14. We claim $E(x; y) = \mathbf{Z}(x; y)$ is the desired (k, ϵ) strong extractor.

By Fact 6 we can concentrate on high min-entropy flat distributions. Let X be a flat distribution over a subset $X \subseteq \{0,1\}^n$ of cardinality at least $K = 2^k$. Then $k \geqslant mh\log q$ and so $K = 2^k \geqslant q^{mh}$. By Theorem 14, $U_t \circ \mathbf{Z}(X; U_t)$ is a β-almost $((t,t),(1,f(\alpha)),\ldots,(1,f(\alpha)))$ block source, with $2^{-f(\alpha)} = 1/2 + \alpha$.

By Lemma 11 every almost block source is close to a block source, and so $U_t \circ \mathbf{Z}(X; U_t)$ is $(m+1)\beta$ close to a $((t,t),(1,f(\alpha)),\ldots,(1,f(\alpha)))$ block source. Also, by Lemma 13, a block-source with one-bit blocks is close to uniform, and so $U_t \circ \mathbf{Z}(X; U_t)$ is $(m+1) \times (\alpha + \beta) = \epsilon$ close to uniform, as desired. □

To reduce the $O(\log m)$ additive penalty in the number of truly random bits we can use the extractor for almost block sources:

Proof of Theorem 3. Suppose $m = m(n)$, $k = k(n)$ and $\epsilon = \epsilon(n)$ are such that $3m\sqrt{n}\log(\frac{n}{\epsilon}) \leqslant k \leqslant n$ and $m \geqslant 2\log^2 \frac{1}{\epsilon}$. Let us set $\alpha = 0.1$, $\beta = \Theta(\frac{\epsilon}{\log^* m})$. This determines q to be $\Theta(\frac{h}{\alpha^4\beta^4}) = \Theta(\sqrt{\frac{n}{\log n}}\frac{1}{\beta^4})$. Let $\mathbf{F}(X; U)$ be the strong extractor for β-almost block sources, of Theorem 15. Our extractor is

$$E(x; y_1, y_2) = \mathbf{F}\big(\mathbf{Z}(x; y_1); y_2\big).$$

That is, we first apply $\mathbf{Z}$ on the input x using the truly random string y_1, and then we apply the extractor $\mathbf{F}$ for β-almost block sources using a fresh truly random string y_2.

Correctness: Let X be a flat distribution over a subset $X \subseteq \{0,1\}^n$ of cardinality at least $K = 2^k$. As before, $|X| \geqslant 2^k \geqslant q^{mh}$. By Theorem 14, $U_t \circ \mathbf{Z}(X; U_t)$ is a β-almost $((t,t),(1,f(\alpha)),\ldots,(1,f(\alpha)))$ block source, with $2^{-f(\alpha)} = 1/2 + \alpha$. As $\mathbf{F}$ is a strong extractor for almost block sources, by Theorem 15, $U_t \circ E(X; U_t)$ is $O(\epsilon)$ close to uniform.

Parameters: By Theorem 14, the length of y_1 is $\log n + O(\log(1/\beta)) + O(1) = \log n + O(\log(\log^* m)) + O(\log \epsilon^{-1})$. By Theorem 15 the length of y_2 is $O(\log \epsilon^{-1})$. Thus the number of truly random bits used is as required. The output length of $\mathbf{Z}$ is m, and thus the output length of $\mathbf{F}$ is at least $\frac{f(\alpha)}{2}m - O((\log^* m)^2 \log \epsilon^{-1}) \geqslant \frac{f(\alpha)}{4}m$ provided that $m \geqslant \Omega(\log^2 \frac{1}{\epsilon})$. □

3.5. Increasing the output length

Proof of Theorem 5. We combine our extractor with a known extractor and block extractor, whose properties we describe.

- The strong block extractor of Reingold et al. [14] is a family of functions
  $$\mathbf{RSW} = \mathbf{RSW}_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$$
  with the property that for every δ there exists $\gamma = \gamma(\delta) > 0$ such that for all flat distributions over subsets X of cardinality at least $|X| \geqslant 2^{\delta n}$, the distribution of $U_t \circ B(X; U_t) \circ X$ is within $\frac{1}{4\log n}$ of a
  $$\left((t,t), \left(\frac{\delta}{2}n, 2\gamma n\right), \left(n, \frac{\delta}{4}n\right)\right)$$
  block source. A remarkable property of the construction is that the seed length t is very small, $t \leqslant O(\log\log n)$.
- We also take any explicit strong extractor family with
  $$\mathbf{NZ} = \mathbf{NZ}_n : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$$
  which is a $(2\gamma n, 1/n)$ strong extractor with $m = \gamma n$ and $t = \text{polylog}(n)$, e.g., the extractor of [4].

We first use the block extractor **RSW** with $O(\log\log n)$ truly random bits y_1 to output a block source $W_1 \circ W_2 \circ W_3$. Say $\mathbf{RSW}(x; y_1) = (w_1, w_2, w_3)$. Our extractor then uses a fresh truly random string y_2 and outputs

$$w_1 \circ \mathbf{NZ}\big(w_2; E(w_3; y_2)\big),$$

where E is the extractor from Theorem 2 (or the more complicated Theorem 3) with polylog(n) output bits and $\epsilon = \frac{1}{4\log n}$ error.

To prove correctness we apply Lemma 12 with three extractors E_1, E_2 and E_3 where E_1 is the identity function $E_1(x, y) = x$, $E_2 = \mathbf{NZ}$ and $E_3 = E$ the extractor from Theorem 2. The number of truly random bits is the sum of those used for the block extractor **RSW** and the extractor $E_3 = E$, which is $\log n + O(\log\log n)$. The total error is the sum of the errors from the block extractor and the two extractors, which is less than $1/\log n$. $\square$

4. The reduction to almost block sources

To prove Theorem 14, we show that if $\mathbf{Z}$ is not as required, then there exists a large subset X'' of X such that the answers to a small number of queries distinguish elements of X''. This shows that X'' is small, and therefore X itself is small. The reader may want to review the outline given in Section 1.4. We begin with some definitions.

Let $f : \mathbb{F}^2 \to \mathbb{F}$. A *query* to f is either

- a *point query* of the form, "what is $f(v)$?" for some point $v \in \mathbb{F}^2$, or,

- a *line query* of the form "which polynomial among $p_1, \dots, p_B$ is equal to f restricted to the line L, denoted $f|_L$?" If none of the polynomials in $p_1, \dots, p_B$ equals $f|_L$ the answer is "quit."

We will be interested in the number of possibilities a query distinguishes among. This is q for the point queries and B for the line queries. The number of possibilities a set of queries distinguishes is the product of the possibilities for each query in the set.

A *predictor* P for bivariate polynomials is a probabilistic function which on input $a \in \mathbb{F}^2$ makes queries $Q(a)$ about a bivariate polynomial $f : \mathbb{F}^2 \to \mathbb{F}$. The set $Q(a)$ may be chosen at random and may or may not depend on a. On receiving the answers to the queries from an oracle, P computes a subset $P(a) \subseteq \mathbb{F}$. P has *preprocessed queries* if $Q(a)$ does not depend on a, and P is *deterministic* if it does not use random coins (neither to choose Q nor to compute its answer).

P has *A possible answers* if for every $a \in \mathbb{F}^2$, every possible set of queries and every set of answers, the size of $P(a)$ is at most A. *P predicts $f : \mathbb{F}^2 \to \mathbb{F}$ with success p* if, when the oracle answers the queries according to f,

$$\Pr_{a \in \mathbb{F}^2,\ \text{coins of } P} \big[f(a) \in P(a) \big] \geqslant p,$$

P predicts $S \subseteq \{0,1\}^n$ with success p if, for every $x \in S$, P predicts f_x with success p.

Suppose P is a predictor for bivariate polynomials and $f : \mathbb{F}^2 \to \mathbb{F}$ is a bivariate polynomial. $P_Q^f(a)$ denotes the subset $P(a)$ when the queries are $Q(a)$ and the oracle answers according to f. Whenever the set of queries Q is clear from the context we denote it by $P^f(a)$. If P is deterministic, the set of queries Q is completely determined by a and we always denote it by $P^f(a)$.

Theorem 14 follows from the following proposition.

Proposition 16. *If $U_t \circ \mathbf{Z}(X; U_t)$ is not β-almost $((t,t), (1, f(\alpha)), \dots, (1, f(\alpha)))$ block source, $2^{-f(\alpha)} = 1/2 + \alpha$, then there exists a deterministic predictor for a set X'' of size at least $|X''| \geqslant \frac{\alpha\beta}{4}|X|$, with 1 possible answer and success 1. There are at most $(m-1)h$ point queries and h line queries, all preprocessed, and each line query distinguishes at most $O(1/\alpha^3\beta^3) = o(q)$ possibilities. Hence $|X| = o(q^{mh})$.*

Notice that while in general we need about $h^2/2$ values to determine an arbitrary degree $h-1$ bivariate polynomial, here only about mh queries suffice. This immediately implies X is small.

We prove Proposition 16 in the next three subsections.

4.1. Evaluating a point

Suppose X is such that $U_t \circ \mathbf{Z}(X; U_t)$ is not a β-almost $((t,t), (1, f(\alpha)), \dots, (1, f(\alpha)))$ block source and let us fix X. We define a predictor, which depends on the subset X.

$\mathbf{EP} = \mathbf{EP}_{i_0}$: *Evaluate point*

Input: $a = (a_1, a_2) \in \mathbb{F}^2$.

Parameter: $1 \leqslant i_0 \leqslant m$.

Queries: The query points are $Q(a_1, a_2) = \{(a_1 - (i_0 - 1), a_2), \ldots, (a_1 - 1, a_2)\}$. Let b_i denote the answer to the query $(a_1 - i, a_2)$.

Algorithm:

- For every $j \in [\bar{\ell}]$ and $b_1, \ldots, b_{i_0-1}$ define the set $X_{j,b_1,\ldots,b_{i_0-1}}$ to be all $x \in X$ for which $\mathbf{C}(f_x(a_1 - i, a_2))_j = \mathbf{C}(b_i)_j$ for every $i = 1, \ldots, i_0 - 1$. Let

$$g_j(a) = \text{MAJORITY}_{x \in X_{j,b_1,\ldots,b_{i_0-1}}}\big(\mathbf{C}\big(f_x(a)\big)_j\big).$$

  (Ties are broken arbitrarily.)
- Set $g(w) = g_1(w) \ldots g_{\bar{\ell}}(w)$.

Output: $\mathbf{EP}(w)$ is the set of all codewords of $\mathbf{C}$ that have at least $\frac{1}{2} + \frac{\alpha\beta}{4}$ relative agreement with $g(w)$.

Note that $\mathbf{EP}$ is deterministic and that the queries depend on the input a. Also, since $\mathbf{C}$ has combinatorial list decoding property $\frac{\alpha\beta}{4}$ for any a, $|\mathbf{EP}^{f_x}(a)| \leqslant O(\frac{1}{\alpha^2\beta^2})$.

Lemma 17. *There exists a subset $X' \subset X$ of cardinality $|X'| \geqslant \frac{\alpha\beta}{2}|X|$ and $1 \leqslant i_0 \leqslant m$ such that $\mathbf{EP} = \mathbf{EP}_{i_0}$ predicts X' with at most $m - 1$ point queries, $A = O(\frac{1}{\alpha^2\beta^2})$ possible answers and $p = \frac{\alpha\beta}{4}$ success.*

Proof. Recall that the random string $y \in U_t$ indexes a point $a \in \mathbb{F}^2$ and a value $j \in [\bar{\ell}]$. Let $A_{z,a,j}$ be a Boolean random variable that is one when $H_\infty(\mathbf{Z}_{i_0} \mid \mathbf{Z}_{[1,i_0-1]} = z_{[1,i_0-1]}, a, j) < f(\alpha)$ and zero otherwise. As $U_t \circ \mathbf{Z}(X; U_t)$ is not a β-almost $((t,t), (1, f(\alpha)), \ldots, (1, f(\alpha)))$ block source, by definition, there exists an $i_0 \in \{1, \ldots, m\}$ such that $\Pr_{a,j,z\in\mathbf{Z}}(A_{z,a,j} = 1) \geqslant \beta$. Now,

$$\begin{aligned}
&\Pr_{x\in X,a,j}\big[g_j(a) = \mathbf{C}\big(f_x(a)\big)_j\big] \\
&\quad= \Pr_{x\in X,a,j}[A_{z,a,j} = 1] \cdot \Pr\big[g_j(a) = \mathbf{C}\big(f_x(a)\big)_j \mid A_{z,a,j} = 1\big] \\
&\qquad+ \Pr_{x\in X,a,j}[A_{z,a,j} = 0] \cdot \Pr\big[g_j(a) = \mathbf{C}\big(f_x(a)\big)_j \mid A_{z,a,j} = 0\big] \\
&\quad\geqslant \beta\left(\frac{1}{2} + \alpha\right) + (1 - \beta)\frac{1}{2} = \frac{1}{2} + \alpha\beta.
\end{aligned}$$

An averaging argument shows that there exists a subset $X' \subset X$ of cardinality at least $\frac{\alpha\beta}{2}|X|$ such that for every $x \in X'$,

$$\Pr_{a,j}\big[g_j(a) = \mathbf{C}\big(f_x(a)\big)_j\big] \geqslant \frac{1}{2} + \frac{\alpha\beta}{2}.$$

Another averaging argument yields for every $x \in X'$

$$\Pr_a\left[\Pr_j\big[g_j(a) = \mathbf{C}\big(f_x(a)\big)_j\big] > \frac{1}{2} + \frac{\alpha\beta}{4}\right] \geqslant \frac{\alpha\beta}{4}.$$

For every a such that $\Pr_j[g_j(a) = \mathbf{C}(f_x(a))_j] > \frac{1}{2} + \frac{\alpha\beta}{4}$ it holds that $f_x(a) \in \mathbf{EP}(a)$. It follows for every $x \in X'$ we have $\Pr_a[f_x(a) \in \mathbf{EP}(a)] \geqslant \frac{\alpha\beta}{4}$ as desired. □

4.2. Evaluating a line

We use procedure **EP** to build procedure **EL** (for "evaluate line") that given a line L makes some specific queries and outputs a single polynomial over $\mathbb{F}$.

EL: *Evaluate line*

```
Input:     A line $L : \mathbb{F} \to \mathbb{F}^2$ not parallel to the $x$-axis.
Algorithm:
           • For every $i = 1, \ldots, q$:
             – Evaluate $\mathbf{EP}(L(i))$ by making the necessary point queries.
           • Form the set $S = \{(i, w) \mid i \in [1, q],\ w \in \mathbf{EP}(L(i))\}$.
           • Compute the list $G$ of all univariate polynomials $g : \mathbb{F} \to \mathbb{F}$ of degree at most
             $h - 1$ with agreement at least $\frac{\alpha\beta}{8}q$ with $S$. We will soon prove that $G$ is always
             a small set, $|G| \leqslant O(\frac{1}{\alpha^2\beta^2})$.
Output:    Query which one of the polynomials in $G$ is equal to $f|_L$, and output this poly-
           nomial. If no polynomial in $G$ equals to $f|_L$ output "quit."
```

We now show that **EL** does well on random lines.

Lemma 18. *For every $x \in X'$:*

$$\Pr_L\left[\mathbf{EL}^{f_x}(L) \neq f_x(L)\right] \leqslant \eta = O\left(\frac{1}{\alpha\beta q}\right).$$

Once G contains the right polynomial, the answer of $\mathbf{EL}^{f_x}(L)$ is correct by definition. The lemma therefore follows from the following claim, which shows that G almost always contains the right polynomial.

Claim 19. *For every $x \in X'$, $\Pr_L[f_x(L) \notin G] \leqslant O(\frac{1}{\alpha\beta q})$.*

Proof. Call $v \in \mathbb{F}^2$ *nice* for $p : \mathbb{F}^2 \to \mathbb{F}$ if $p(v) \in \mathbf{EP}^p(v)$. Fix any $x \in X'$. Let Y_i be the random variable indicating whether $L(i)$ is nice for f_x, and $Y = \sum_{i=1}^q Y_i$. We have $E(Y) \geqslant \frac{\alpha\beta}{4}q$, i.e., for every $x \in X'$ we expect to see many nice points on a random line L. We say L is bad for x if $Y \leqslant \frac{\alpha\beta}{8}q$. Since L is a random line, the points on L are pairwise independent. Therefore,

$$\Pr\left[Y \leqslant \frac{E(Y)}{2}\right] \leqslant \frac{4 \cdot \mathrm{VAR}(Y)}{(E(Y))^2} \leqslant O\left(\frac{\alpha\beta q}{(\alpha\beta q)^2}\right) = O\left(\frac{1}{\alpha\beta q}\right).$$

If the line L is bad for x (which happens with probability at most $O(\frac{1}{\alpha\beta q})$) we lose. Otherwise, $Y > \frac{E(Y)}{2}$ and the line L contains at least $\frac{\alpha\beta}{8}q$ nice points $v = L(i)$. Therefore, $f_x(L) \in G$. □

We now show that the list decoding process does not return many possible solutions. Since $|S| \leqslant O(\frac{q}{\alpha^2\beta^2})$ and $q = \Omega(\frac{h}{\alpha^4\beta^4})$ we have $\frac{\alpha\beta q}{8} \geqslant \sqrt{2h \cdot |S|}$ and therefore by Lemma 7, $|G| \leqslant O(\frac{|S|}{\alpha\beta q}) = O(\frac{1}{\alpha^2\beta^2} \cdot \frac{1}{\alpha\beta}) = O(\frac{1}{\alpha^3\beta^3})$.

We now take a closer look at the point queries done in $\mathbf{EL}(L)$. If L is not parallel to the x-axis then $\mathbf{EL}(L)$ makes point queries for each point on each of the $i_0 - 1$ lines $L - (j, 0)$, $j \in [i_0 - 1]$. However, since the values of f_x on a line can be determined by querying h points on that line, it suffices to query only h points from each line, and the number of point queries is at most $(i_0 - 1)h$. Denote this set of point queries $Q(L)$.

4.3. *Proof of Proposition 16*

Proof. We now give a procedure **EA** (for "evaluate all") that makes few queries and outputs, with good probability, the unique polynomial $f_x \in X''$ that agrees with the queries.

EA: *Evaluate all*

```
Input:     None.
Algorithm: Pick a random line $L$, and query the points in $Q(L)$.
             For $j = 0$ to $h - i_0$.
           • Evaluate $\mathbf{EL}(L + (j, 0))$. Note that all of the point queries needed to evaluate
             this have been made or deduced previously. Only the line queries need to be
             made.
             Output the unique degree $h - 1$ polynomial consistent with the answers on the
           $h \times h$ block.
```

To prove Proposition 16, we need a deterministic predictor, so we show that we can fix L suitably. Say that a line L is good for $f : \mathbb{F}^2 \to \mathbb{F}$ if $\mathbf{EA}_L^f$, the output of $\mathbf{EA}^f$ when picking the line L, is f, and bad otherwise. Recall that η, defined in Lemma 18, is the error probability for **EL**.

Claim 20. *For every $x \in X'$, $\Pr_L[L$ is bad for $f_x] \leqslant h\eta$.*

Proof. For each of the h lines we learn, the probability we fail (given the right answers to the line we use) is at most η. By the union bound (regardless of correlations) the claim follows. □

Now, $h\eta \leqslant O(\frac{h}{\alpha\beta q}) \leqslant 1/2$ and therefore for every $x \in X'$, $\Pr_L[L$ is good for $f_x] \geqslant 1/2$. Hence, there is at least one fixed choice of a line L and a subset $X'' \subseteq X'$ of cardinality at least $|X''| \geqslant \frac{1}{2}|X'|$ such that L is good for every $x \in X''$. Fix this L. $\mathbf{EA}_L$ is the required deterministic predictor.

Note that the number of point queries is at most $(i_0 - 1)h \leqslant (m - 1)h$, and the number of line queries is at most $h - i_0 + 1 \leqslant h$. Also, the line queries distinguish at most $O(1/(\alpha^3\beta^3)) \leqslant o(q)$ possibilities. Thus the predictor distinguishes at most

$q^{(m-1)h}(o(q))^h \leqslant o(q^{mh})$ possibilities. Since $1/(\alpha^3\beta^3) = O(\alpha\beta q)$, we also conclude that $|X| \leqslant \frac{4}{\alpha\beta}|X''| \leqslant o(q^{mh})$. Proposition 16 and hence Theorem 14 follow. □

5. Extractors for almost block sources

5.1. Reducing a β-almost block source to a block source with smaller penalty

We saw that a β-almost block source with m blocks is $m\beta$ close to a block source (Lemma 11). However, it is important for us to avoid this $m\beta$ penalty. The key observation is that at a price of losing at most half the entropy, if we group ℓ consecutive blocks, then instead of being $\ell\beta$ close to the behavior we expect from a block source we are 2β close to it.

Lemma 21. *Suppose $Z = Z_1 \circ \cdots \circ Z_m$ is a β-almost $((n_1, k), \dots, (n_b, k))$ block source. For every $1 \leqslant a \leqslant b \leqslant m$*

$$\Pr_{z_{[1,a-1]} \in Z_{[1,a-1]}}\left[H_\infty(Z_{[a,b]} \mid Z_{[1,a-1]} = z_{[1,a-1]}) \leqslant \frac{1}{2}\sum_{j=a}^{b} k\right] \leqslant 2\beta.$$

Proof. We say $i \in [m]$ is bad for $z_{[1,i-1]} \in \text{support}(Z_{[1,i-1]})$ if $H_\infty(Z_i \mid Z_{[1,i-1]} = z_{[1,i-1]}) < k$. We say i is bad for $z \in Z$ if i is bad for $z_{[1,i-1]}$. By assumption, for all i, $\Pr_{z\in Z}[i \text{ is bad for } z] \leqslant \beta$. Hence, using Markov,

$$\Pr_{z\in Z}\big[\text{at least half of } i \in [a,b] \text{ are bad for } z\big] \leqslant 2\beta.$$

Whenever at most half of $i \in [a,b]$ are bad for z, $\Pr[Z_{[a,b]} = z_{[a,b]} \mid Z_{[1,a-1]} = z_{[1,a-1]}] \leqslant \prod_{i:\, i \text{ is not bad for } z} 2^{-k} \leqslant 2^{-k(b-a)/2}$ and the lemma follows. □

Thus given, e.g., a β-almost $((1, f(\alpha)), \dots, (1, f(\alpha)))$ block source with m blocks and some constant α, we can group the m blocks into, say, b meta-blocks such that each meta-block contains at least half the desired amount of entropy. We can then redistribute the weight of the bad strings in the same way as is done in the proof of Lemma 11 (and by Lemma 21 these are at most $O(b\beta)$ fraction of all strings) and get a block source with b blocks and the desired entropy. The error term $O(b\beta)$ is significantly smaller than $m\beta$ if b is small. In the next subsection we follow this outline for a specific choice of parameters that suits our needs.

5.2. An extractor for a block source with very small seed

Lemma 22. *Let γ be a positive constant and $\epsilon = \epsilon(n) > 0$. There exists $b = b(n) = O(\log^* n)$ and $n_1, \dots, n_b$, $n_1 + \cdots + n_b = n$, and a strong ϵ-extractor for $((n_1, \gamma n_1), \dots, (n_b, \gamma n_b))$ block sources, using $t = O(\log \frac{1}{\epsilon})$ random bits, and outputting $\gamma n - O(\log^* n \times \log \epsilon^{-1} + \log^4 \frac{1}{\epsilon})$ output bits.*

Proof. We choose the integers n_i as follows:

- $n_b = O(\log^4 \frac{1}{\epsilon})$,
- $n_{b-1} = \frac{1}{\epsilon^2}$, and
- $n_{i-1} = 2^{n_i^{1/6}}$ for $i < b$.

It can be verified that $n_{i-2} \geqslant 2^{n_i}$ and hence it follows that $b = O(\log^* n)$. We will need two basic extractors:

- The RRV family of strong extractors [34, Theorem 4]:
$$E_n^{\mathrm{RRV}} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m,$$
which is a (k, ϵ) strong extractor, with $t \leqslant c_0 \log^3 n \log \epsilon^{-1}$ and $m = k - 2\log\epsilon^{-1} - O(1)$, for some constant c_0.
- The Zuckerman family of strong extractors [9]:
$$E_n^{Z} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m,$$
which is a $(k = \Omega(n), \epsilon)$ strong extractor, with $t \leqslant O(\log n + \log \epsilon^{-1})$ and $m = \Omega(k)$.

We want to apply Lemma 12. For that we choose $(k_i = \gamma n_i, \epsilon_i = \frac{\epsilon}{2 \cdot 2^{b-i}})$ strong extractors:
$$E_i : \{0,1\}^{n_i} \times \{0,1\}^{r_i} \to \{0,1\}^{r_{i-1} - r_i}.$$
We choose E_b to be $E_{n_b}^{Z}$ with $r_b = O(\log n_b + \log \epsilon^{-1}) = O(\log \epsilon^{-1})$, and we choose E_j to be $E_{n_j}^{\mathrm{RRV}}$ for $j < b$. We need to verify that the extractors E_i indeed support an output length $r_{i-1} - r_i$.

Indeed, for E_b we have a constant entropy rate and $r_b = O(\log n_b + \log \epsilon^{-1})$, hence we have $r_{b-1} = \Omega(n_b) = \Omega(\log^4 \frac{1}{\epsilon})$ output bits, which is larger than $c_0 \log^3(n_{b-1}) \log(\frac{1}{\epsilon_{b-1}}) = 8c_0 \log^3 \frac{1}{\epsilon} \log(\frac{1}{\epsilon_{b-1}}) \approx 8c_0 \log^4 \frac{1}{\epsilon}$ for the appropriate choice of constant in the choice of n_b. For E_j, $j < b$, it is easy to verify that $r_i \geqslant c_0 \cdot \log^3 n_i \cdot \log \frac{1}{\epsilon_i}$ as needed.

Having that, Lemma 12 assures us that we get a strong extractor $F : \{0,1\}^m \times \{0,1\}^{r_b} \to \{0,1\}^{r_1 - r_b}$ with error at most $\sum \epsilon_i \leqslant \epsilon$. The number of truly random bits is $r_b = O(\log \epsilon^{-1})$ as desired. To analyze the number of output bits, we notice that in applying E_b we lose at most $O(n_b) = O(\log^4 \frac{1}{\epsilon})$ entropy, and applying each E_j, $j < b$, we lose at most $2\log\epsilon^{-1} + O(1)$. Altogether, we lose at most $O(b \log \epsilon^{-1} + \log^4 \frac{1}{\epsilon})$ entropy. □

5.3. Putting it together

We now prove Theorem 15.

Proof of Theorem 15. Let $Z = Z_1 \circ \cdots \circ Z_m$ be a $((1, k_1), \ldots, (1, k_1))$ block source. We partition the m blocks into b meta-blocks, the ith block being of length n_i (as in Lemma 22) and $\sum_{i=1}^{b} n_i = m$. We look at Z as $W_1 \circ \cdots \circ W_b$ with each W_i distributed over $\{0,1\}^{n_i}$. By

Lemma 21, $W_1 \circ \cdots \circ W_b$ is $O(b\beta)$ close to a $((n_1, \frac{k_1}{2}n_1), \ldots, (n_b, \frac{k_1}{2}n_b))$ block source. We now apply the block source extractor of Lemma 22 with $\gamma = k_1/2$.

The number of truly random bits used is $r_b = O(\log \epsilon^{-1})$. The error is $O(b\beta)$ because of the reduction from an almost block source to a block source, plus ϵ from Lemma 22. The entropy loss is $\frac{k_1}{2}n$ because of the reduction from almost block sources to block sources (by far the most significant loss), plus the entropy loss from Lemma 22. Theorem 15 follows. □

6. The multivariate extractor

We now describe a generalization of the bivariate extractor to D dimensions, where $D \geqslant 3$. Let h be the smallest integer such that $\binom{h+D}{D}\log(h) \geqslant n$. Let $\mathbb{F} = \mathbb{F}_q$ be a field of prime size $q \gg h$. We view the n-bit input string $x \in \{0,1\}^n$ as a function $f_x : \mathbb{F}^D \to \mathbb{F}$ of total degree at most h. As it takes $\binom{h+D}{D}\log q \geqslant \binom{h+D}{D}\log h \geqslant n$ bits to specify a polynomial, different inputs map to different polynomials.

The function $\mathbf{ME}_D$

Input: $x \in \{0,1\}^n$.
Setting: $h = \lceil D(\frac{n}{\log h})^{1/D} - D\rceil$ (and so $\binom{h+D}{D}\log h \geqslant n$). $\mathbb{F}$ is a field of size q where q is the smallest prime larger than $\Omega(m^{\max(4,D-1)}h)$.
We identify $x \in \{0,1\}^n$ with a function $f_x : H^D \to \mathbb{F}$ of total degree h.
Binary code: $\mathbf{C}$ is a linear binary code with dimension $\ell = \log q$, length $\bar{\ell}$ and combinatorial list decoding property $\alpha = \frac{1}{8m}$ (see Section 2.1).
Random coins: $a \in \mathbb{F}^D$, $d \in [D-1]$, $j \in [\bar{\ell}]$.
Output: $\mathbf{ME}(x; a, d, j)_i = \mathbf{C}(f_x(a + ie_d))_j$ for $i \in [m]$. Here e_d denotes the basis vector in $\mathbb{F}^D$ with a 1 in the dth position and 0's elsewhere.

We now give a top-level proof of Theorem 4.

Proof of Theorem 4. We first check parameters:

$$\begin{aligned}
t &\leqslant D\log q + \log D + \log\left(\log q \cdot \left(\frac{m}{\epsilon}\right)^4\right) \\
&\leqslant D\log\bigl(m^{D-1}h\bigr) + \log D + \log\log q + O(\log m) \\
&\leqslant O\bigl(D^2\log m\bigr) + \log\bigl(h^D\bigr) + \log\log q \\
&\leqslant O\bigl(D^2\log m\bigr) + O(D\log D) + \log n - \log\log h + \log\log q \\
&= \log n + O\bigl(D^2\log m\bigr).
\end{aligned}$$

For the correctness proof, we suppose $U_t \circ \mathbf{ME}(X, U_t)$ is not ϵ close to uniform for some X with $|X| \geqslant 2^k$, and fix X. We first establish that the assumption that ϵ is close to 1 ($\epsilon = 1 - \frac{1}{8D}$) implies that there are next-bit predictors in each orthogonal direction except the last.

Lemma 23. *If $U_t \circ \mathbf{ME}(X, U_t)$ is not $\epsilon = 1 - \frac{1}{8D}$ close to uniform for some X with $|X| \geqslant 2^k$ then there are tests $T_1, \ldots, T_{D-1} : \{0,1\}^{m-1} \to \{0,1\}$ and a subset $X' \subseteq X$ of cardinality at least $\frac{1}{2}|X|$ such that for every $x \in X'$ and every $d \in [D-1]$:*

$$\Pr_{\substack{x \in X,\, a \in \mathbb{F}^D,\, j \in [\bar{\ell}] \\ y \in \mathbf{ME}(x;a,d,j)}} \big[T_d(a, d, j, y_1, \ldots, y_{m-1}) = y_m\big] \geqslant \frac{1}{2} + \frac{1}{4m}. \tag{1}$$

We use the next-bit predictors to define procedures $\mathbf{EP}(a, d)$ that try to predict $f_x(a)$ given its evaluation on the $m-1$ preceding points in the dth dimension. As in the bivariate case, $\mathbf{EP}(a, d)$ always returns a small set of possible answers of cardinality at most $A \leqslant O(1/\alpha^2) = O(m^2)$, and we prove that for every $x \in X'$ and $d \in [D-1]$, $\Pr_a[f_x(a) \in \mathbf{EP}^{f_x}(a, d)] \geqslant \alpha = O(1/m)$.

To evaluate everything, we pick a random line and call $\mathbf{Eval}(d, L)$. $\mathbf{Eval}(d, L)$ tries to compute f_x on the set $U_{d,L} = L + \mathrm{span}\{e_1, \ldots, e_d\}$ (for the definition of the $+$ operation on sets see Section 2). Thus, for $d = 0$ we have $U_{0,L} = L$ and we try to learn the values of f_x on the one-dimensional line L. For $d = 1$ we try to learn the values of f_x on a two-dimensional affine subspace $U_{1,L} = L + \mathrm{span}\{e_1\}$ and so forth, eventually learning f_x on the whole space $\mathbb{F}^D$, i.e., learning x itself. We conclude that if the extractor fails then the set X' has a small description. Hence the set X' (and therefore the set X) are small sets, and the random variable X has low min-entropy. The contrapositive of this is Theorem 4. □

6.1. Preliminaries

We record a generalization of Lemma 7.

Lemma 24. *Let $\mathbb{F}$ be a field of size q, $h < q$ and $\delta > 0$ such that $q \geqslant 16A^2h/\delta^2$. For each element $u \in \mathbb{F}^D$ assign a set $S_u \subseteq \mathbb{F}$ of size at most A. Then there are less than $4A/\delta$ D-variate polynomials p of total degree h such that $p(u) \in S_u$ for at least a δ fraction of points, provided that $\delta \geqslant 2\sqrt{2hA/|\mathbb{F}|}$.*

Proof. The case $D = 1$ is a weakening of Lemma 7; the main difference is that Lemma 7 is stated with absolute agreement size whereas this lemma is stated with relative agreement size.

We now prove for general D by reducing to Lemma 7. Suppose there was a set $\mathcal{P}$ of $4A/\delta$ such polynomials. Pick a random line L, and consider a polynomial $p \in \mathcal{P}$ restricted to L. By Chebychev, with probability at least $1 - \frac{2}{\delta q}$, at least a $\delta/2$ fraction of points $u \in L$ satisfy $p(u) \in S_u$. By the union bound, the probability there exists a polynomial $p \in \mathcal{P}$ that does not satisfy the above is at most $\frac{8A}{\delta^2 q}$, which by our choice of q is smaller than half.

Also, for any two different degree h polynomials, their restrictions to a random line are equal with probability at most h/q (because a random line in particular contains a random point).[6] The probability that there exists two such polynomials in $\mathcal{P}$ is at most $\binom{4A/\delta}{2}\frac{h}{q} < \frac{8A^2h}{\delta^2 q} \leqslant 1/2$.

[6] The bound can be obviously improved, but the improvement is not necessary for what we need.

We conclude that there exists a line L that splits $\mathcal{P}$ (i.e., the polynomials in $\mathcal{P}$ have different restrictions to L) and such that each $p \in \mathcal{P}$ has a $\delta/2$ fraction of agreement on the line. However, this contradicts Lemma 7. □

6.2. A predictor in each direction

Proof of Lemma 23. Assume there exists a test $T : \{0,1\}^m \to \{0,1\}$ that $\epsilon = 1 - \frac{1}{8D}$ distinguishes $U_t \circ \mathbf{ME}(X, U_t)$ from the uniform distribution, i.e.,

$$\Pr\bigl[T\bigl(U_t, \mathbf{ME}(X, U_t)\bigr) = 1\bigr] - \Pr\bigl[T(U_{t+m}) = 1\bigr] = \epsilon = 1 - \frac{1}{8D}.$$

Let U^d denote the uniform distribution on $a \in \mathbb{F}^D$, $j \in [\bar{\ell}]$ with $d \in [D-1]$ held fixed, and let δ_d be the advantage T has on U^d, i.e.,

$$\Pr\bigl[T\bigl(U^d, \mathbf{ME}\bigl(X, U^d\bigr)\bigr) = 1\bigr] - \Pr\bigl[T(U_{t+m}) = 1\bigr] = 1 - \delta_d$$

and notice that we must have $\delta_d > 0$ for every $d \in [D-1]$ (otherwise ϵ cannot be that close to 1). Then, $\frac{1}{D-1}\sum \delta_d \leqslant \frac{1}{8D}$ and therefore $\sum \delta_d < 1/8$. By a Markov argument, for each d, for at least a $1 - 4\delta_d$ fraction of $x \in X$,

$$\Pr\bigl[T\bigl(U^d, \mathbf{ME}\bigl(x, U^d\bigr)\bigr) = 1\bigr] - \Pr\bigl[T(U_{t+m}) = 1\bigr] \geqslant \frac{3}{4}. \tag{2}$$

Therefore, the fraction of $x \in X$ for which (2) holds for all j is at least $1 - 4\sum \delta_d > 1/2$. This set is X'. Now, for every $d \in [D-1]$ we use Yao's next element predictor argument to convert T into a predictor T_d with advantage $\frac{1}{4m} = 2\alpha$. By the symmetry of $\mathbf{ME}$, we can assume that the predictor predicts the last bit well. We obtain that for all $x \in X'$, Eq. (1) holds. □

6.3. Evaluating a point using a given direction

We introduce our point evaluator **EP**.

EP: *Evaluate point*

Input: $a \in \mathbb{F}^D$, $d \in [D-1]$.
Parameters: $\alpha = \frac{1}{8m}$.
Queries: The query points are $a - (m-1)e_d, a - (m-2)e_d, \ldots, a - e_d$; the answers are $b_1, \ldots, b_{m-1}$.
Algorithm: For all $j \in [\bar{\ell}]$ compute

$$g_j(a) = T_d\bigl(a, d, j, \mathbf{C}(b_1)_j, \ldots, \mathbf{C}(b_{m-1})_j\bigr)$$

and set $g(a) = g_1(a) \cdots g_{\bar{\ell}}(a)$.
Output: $\mathbf{EP}(a, d)$ is the set of all codewords of $\mathbf{C}$ that have at least $1/2 + \alpha$ agreement with $g(a)$.

We note that the best possible predictor T_d chooses the majority vote for $C(f_x(a))_j$ over all x that are consistent with $\mathbf{C}(b_1)_j, \dots, \mathbf{C}(b_{m-1})_j$, as in Section 4, and so we could have given an alternative definition of $g_j(a)$ using that best predictor rather than T_d.

The proof of the following lemma is identical to that of the second half of Lemma 17.

Lemma 25. *For every $x \in X'$ and $d \in [D-1]$, $\Pr_a[f_x(a) \in \mathbf{EP}^{f_x}(a,d)] \geqslant \alpha$.*

Also, as C has combinatorial list decoding property α, for every x and d we must have $|\mathbf{EP}(a,d)| \leqslant A = O(1/\alpha^2)$.

6.4. Evaluating all

To evaluate everything, we pick a random line and call $\mathbf{Eval}(d,L)$. $\mathbf{Eval}(d,L)$ tries to compute f_x on the set $U_{d,L} = L + \operatorname{span}\{e_1,\dots,e_d\}$. Thus, for $d=0$ we have $U_{0,L} = L$ and we try to learn the values of f_x on the one-dimensional line L. For $d=1$ we try to learn the values of f_x on the set $U_{1,L} = L + \operatorname{span}\{e_1\}$ and so forth.

Notice that as L itself is a one-dimensional affine subspace, $U_{d,L}$ is also an affine subspace. Also, as L is picked at random, with high probability, L does not belong to an affine translation of $\operatorname{span}\{e_1,\dots,e_{d-1}\}$, and so for every d, $\operatorname{span}\{e_1,\dots,e_{d-1},L\}$ will be d-dimensional. Thus, our goal is to learn $U_{D-1,L}$.

Also, if we define

$$U_{d,L,i} \stackrel{\text{def}}{=} L + \operatorname{span}\{e_1,\dots,e_{d-1}\} + ie_d$$

then it is an affine translation of $\mathbb{F}^d$, and so there is an affine translation map $\phi_i : \mathbb{F}^d \to U_i$. In the bivariate case, this corresponds to viewing the line as a one-dimensional (affine) subspace over $\mathbb{F}$.

We are now ready to present the reconstruction algorithm:

$\mathbf{Eval}^f(d,L)$

```
Input:     A line $L$ and a dimension $d$. The queries are answered by $f$.
Algorithm: If $\operatorname{span}\{e_1,\dots,e_{d-1},L\}$ is not $d$-dimensional we fail and output "don't know".
           Otherwise:

           If $d=0$: $U = L$. Query $U$ on $h$ points, interpolate the unique polynomial $p$ of
                     degree less than $h$, and deduce $f(x)$ for all $x \in L$.
           For $d = 1,\dots,D-1$:
               • For $i = 0,\dots,m-2$ perform $\mathbf{Eval}(d-1, L+ie_d)$.
                 After this we deduce a (hopefully correct) value $\widetilde{f(y)}$ for each $i = 0,\dots,m-2$ and $y \in L + ie_d + \operatorname{span}\{e_1,\dots,e_{d-1}\}$.
               • For $i = m-1$ to $h-1$:
                 – For every $u \in U_i \stackrel{\text{def}}{=} L + ie_d + \operatorname{span}\{e_1,\dots,e_{d-1}\}$ evaluate $\mathbf{EP}(u,d)$.
                   Note that the queries needed for $\mathbf{EP}(u,d)$ have been made or previously deduced.
```

– Define an affine map $\phi_i : \mathbb{F}^d \to U_i$ and form the set $S = \{(v, w) \mid v \in \mathbb{F}^d,\ w \in \mathbf{EP}(\phi_i(v), d)\}$.
Compute the list G of all d-variate polynomials $g : \mathbb{F}^d \to \mathbb{F}$ of degree at most h with agreement at least $\frac{\alpha}{2} q^d$ with S.
– Pick $\Delta = \Delta_d = \Theta(d \log q + \log \frac{1}{\alpha})$ random points $a_1, \ldots, a_\Delta \in \mathbb{F}^d$. Query the points $\phi_i(a_1), \ldots, \phi_i(a_\Delta)$, and let $b_1, \ldots, b_\Delta$ be the answers.
If there is a single polynomial $g \in G$ such that $g(a_i) = b_i$ for $i \in [\ell]$, then deduce that for all v, $f(\phi_i(v)) = g(v)$. Otherwise output "don't know".

Lemma 26. $\Pr_L[\mathbf{Eval}(d, L) \neq f_x(U_{d,L})] \leqslant O(\frac{m^{d-1}h}{\alpha q})$.

Proof. The probability that $\operatorname{span}\{e_1, \ldots, e_{D-1}, L\}$ is not D-dimensional is at most $1/q$. If $\operatorname{span}\{e_1, \ldots, e_{D-1}, L\}$ is D-dimensional then for every $d = 1, \ldots, D-1$ it must be that $\operatorname{span}\{e_1, \ldots, e_{d-1}, L\}$ is d-dimensional. From now on we assume $\operatorname{span}\{e_1, \ldots, e_{D-1}, L\}$ is D-dimensional.

Let us define error_d to be the probability (over choosing a random line L) that $\mathbf{Eval}(d, L) \neq f_x(U_{d,L})$, given that all queries are answered correctly. Clearly, $\text{error}_0 = 0$. Also, we will soon prove the recursion:

Claim 27. $\text{error}_d \leqslant (m-1)\,\text{error}_{d-1} + O(\frac{h}{\alpha q})$

and solving the recursion we get our result. □

Proof of Claim 27. The first term in the recursion comes from the stage where i runs from 0 to $m-2$ and we call $\mathbf{Eval}(d-1, \cdot)$, which has probability of error_{d-1} to fail.

Each i from $m-1$ to $h-1$ causes an additional error, which we analyze analogously to **EL**. Call $v \in \mathbb{F}^D$ *nice* for $p : \mathbb{F}^D \to \mathbb{F}$ if $p(v) \in \mathbf{EP}^p(v, d)$. We know that:

- for every $x \in X'$, $\Pr_{v \in \mathbb{F}^D}[v \text{ is nice for } f_x] \geqslant \alpha$ and
- for every $p : \mathbb{F}^D \to \mathbb{F}$ and v, $|\mathbf{EP}^p(v)| \leqslant A \leqslant O(1/\alpha^2)$.

Fix any $x \in X'$. Points in U_i are indexed by $s \in \mathbb{F}$ and $a \in \operatorname{span}\{e_1, \ldots, e_{d-1}\}$, the point corresponding to s, a being $U_i(s, a) = L(s) + ie_d + a$. For an index $v = (s, a)$, let Y_v be the indicator random variable that is 1 iff $U_i(v)$ is nice for f_x, and $Y = \sum_v Y_v$. We have $E(Y) \geqslant \alpha |U_i| = \alpha q^d$, i.e., for every $x \in X'$ we expect to see many nice points in U_i. We say U_i is bad for x if $Y \leqslant E(Y)/2$.

Every point in U_i is uniform over all possible values (over a random choice of L). We would like to claim that the points in U_i are pairwise independent. If we look at two points, one indexed by $v_1 = (s_1, a_1)$, the other by $v_2 = (s_2, a_2)$, and $s_1 \neq s_2$, then clearly the two points are independent (over a random choice of L). On the other hand, they may be dependent for $s_1 = s_2$. Let us denote $v_1 \sim v_2$ iff $s_1 = s_2$. We have:

$$\operatorname{VAR}(Y) = \sum_v \operatorname{VAR}(Y_v) + 2 \sum_{v1, v2} \operatorname{COVAR}(Y_{v_1}, Y_{v_2})$$

$$\leqslant \sum_v E(Y_v) + 2\sum_{v_1}\sum_{v_2\sim v_1} E(Y_{v_1}Y_{v_2})$$
$$\leqslant E(Y) + 2\sum_{v_1}\sum_{v_2\sim v_1} E(Y_{v_1})$$
$$= E(Y) + 2q^{d-1}E(Y) \leqslant \big(2q^{d-1}+1\big)E(Y).$$

Hence,

$$\Pr\left[Y \leqslant \frac{E(Y)}{2}\right] \leqslant \frac{4\mathrm{VAR}(Y)}{(E(Y))^2} \leqslant O\left(\frac{q^{d-1}E(Y)}{(E(Y))^2}\right) \leqslant O\left(\frac{q^{d-1}}{\alpha q^d}\right) = O\left(\frac{1}{\alpha q}\right).$$

If for some $i \in [h-1, m-1]$, U_i is bad for x we lose, and this accounts for the second error term of $O(\frac{h}{\alpha q})$. Otherwise, $Y > E(Y)/2$ and U_i contains at least $\frac{\alpha}{2}q^d$ nice points. Therefore, $f_x(L) \in G$.

We check and see that $\alpha/2 \geqslant 2\sqrt{2hA/|\mathbb{F}|}$ (because $A = O(1/\alpha^2) = O(m^2)$ and $|\mathbb{F}| \geqslant O(hm^4)$). Also $q \geqslant \Omega(A/\delta^2) = \Omega(m^4)$ and $q \geqslant \Omega(A^2h) = \Omega(nm^4)$. Thus, we can apply Lemma 24 to bound the number of polynomials in G, and we deduce that $|G| \leqslant \frac{4A}{\alpha/2} = 8A/\alpha$. Now, $a_1, \ldots, a_\Delta \in \mathbb{F}^d$ are taken uniformly from $\mathbb{F}^d$. The probability two different polynomials of degree at most h agree on Δ random points is at most $(h/q)^\Delta$. Therefore, the probability there are two or more solutions that agree with the query on $f_x(L(a_i))$ for all $i \in [\Delta]$ is at most

$$\binom{|G|}{2}\left(\frac{h}{q}\right)^\Delta < O\left(\frac{A^2}{\alpha^2}\left(\frac{h}{q}\right)^\Delta\right) \leqslant O\left(\frac{1}{\alpha^6}\left(\frac{1}{2}\right)^\Delta\right) \leqslant \frac{1}{q^d}.$$

This accounts to a third error term of order $O(h/q^d)$ which is swallowed by the second error term.

Otherwise, for every $i = m-1, \ldots, h-1$, U_i has enough nice points, we therefore have the correct answer in G, and the filtering leaves us with a single answer that must be the correct answer. Thus, we learn $U_{d,L}$ correctly. □

Now let queries$_d$ denote the number of queries made in **Eval**$(d, \cdot)$.

Lemma 28. queries$_d \leqslant m^{d-1}(m+\Delta)h$.

Proof. Note that queries$_0 = h$, and for $d \geqslant 1$

$$\text{queries}_d \leqslant (m-1)\text{queries}_{d-1} + (h-m+1)\Delta \leqslant (m-1)\text{queries}_{d-1} + h\Delta.$$

Now solve the recursion. □

In particular, for $d = D-1$ we have $\Delta_{D-1} = \Theta(D\log(q) + \log(m)) = \Theta(D\log(h) + D^2\log(m)) = \Theta(\log(n) + D^2\log(m))$. Since we take D to be a constant, this is merely $\Theta(\log(n))$. We thus see that the total number of queries, queries$_{D-1}$, is bounded by $m^{D-2}(m+\Delta)h \leqslant O(m^{D-1}h\log(n))$ (with the $\log(n)$ term disappearing if $m \geqslant \log(n)$), and each query is $\log(q)$ bits. In bits, this amounts to querying at most $O(m^{D-1}n^{1/D}\log^2(n))$ bits.

We remark that there is nothing special about our choice of $e_1, \ldots, e_{D-1}$. Any set of $D-1$ independent vectors will do for the construction.

Acknowledgments

We thank Chris Umans, Ronen Shaltiel, Oded Goldreich, and Avi Wigderson for helpful discussions and comments. We thank the anonymous referees for numerous corrections and many suggestions that improved the paper presentation.

References

[1] E. Mossel, C. Umans, On the complexity of approximating the VC dimension, J. Comput. System Sci. 65 (2002) 660–671.

[2] M. Sipser, Expanders, randomness, or time vs. space, J. Comput. System Sci. 36 (1988) 379–383.

[3] M. Santha, On using deterministic functions in probabilistic algorithms, Inform. and Comput. 74 (3) (1987) 241–249.

[4] N. Nisan, D. Zuckerman, Randomness is linear in space, J. Comput. System Sci. 52 (1) (1996) 43–52.

[5] D. Zuckerman, General weak random sources, in: Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, 1990, pp. 534–543.

[6] D. Zuckerman, Simulating BPP using a general weak random source, Algorithmica 16 (1996) 367–391.

[7] A. Srinivasan, D. Zuckerman, Computing with very weak random sources, SIAM J. Comput. 28 (1999) 1433–1459.

[8] A. Ta-Shma, On extracting randomness from weak random sources, in: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 1996, pp. 276–285.

[9] D. Zuckerman, Randomness-optimal oblivious sampling, Random Structures Algorithms 11 (1997) 345–367.

[10] L. Trevisan, Extractors and pseudorandom generators, J. ACM 48 (4) (2001) 860–879.

[11] N. Nisan, A. Wigderson, Hardness vs. randomness, J. Comput. System Sci. 49 (1994) 149–167.

[12] R. Impagliazzo, R. Shaltiel, A. Wigderson, Near-optimal conversion of hardness into pseudo-randomness, in: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 181–190.

[13] R. Impagliazzo, R. Shaltiel, A. Wigderson, Extractors and pseudo-random generators with optimal seed length, in: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 1–10.

[14] O. Reingold, R. Shaltiel, A. Wigderson, Extracting randomness via repeated condensing, in: Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, 2000, pp. 22–31.

[15] A. Ta-Shma, C. Umans, D. Zuckerman, Loss-less condensers, unbalanced expanders, and extractors, in: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 143–152.

[16] J. Radhakrishnan, A. Ta-Shma, Bounds for dispersers, extractors, and depth-two superconcentrators, SIAM J. Discrete Math. 13 (1) (2000) 2–24.

[17] A. Wigderson, D. Zuckerman, Expanders that beat the eigenvalue bound: Explicit construction and applications, Combinatorica 19 (1) (1999) 125–138.

[18] C. Umans, Hardness of approximating Σ_2^p minimization problems, in: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 465–474.

[19] A. Russell, D. Zuckerman, Perfect-information leader election in $\log^* n + O(1)$ rounds, J. Comput. System Sci. 63 (2001) 612–626.

[20] O. Goldreich, D. Zuckerman, Another proof that BPP $\subseteq$ PH (and more), Technical report TR97-045, Electronic Colloquium on Computational Complexity, 1997.

[21] A. Ta-Shma, D. Zuckerman, Extractor codes, IEEE Trans. Inform. Theory 50 (2004) 3015–3025.

[22] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, Acta Math. 182 (1999) 105–142.

[23] V. Guruswami, Better extractors for better codes?, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 436–444.

[24] D. Zuckerman, Linear degree extractors and inapproximability, manuscript, 2005.

[25] R. Shaltiel, C. Umans, Simple extractors for all min-entropies and a new pseudo-random generator, in: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 648–657.

[26] C. Umans, Pseudo-random generators for all hardnesses, in: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 627–634.

[27] M. Sudan, L. Trevisan, S. Vadhan, Pseudorandom generators without the XOR lemma, in: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 537–546.

[28] M. Sudan, Decoding of Reed Solomon codes beyond the error-correction bound, J. Complexity 13 (1997) 180–193.

[29] V. Guruswami, J. Hastad, M. Sudan, D. Zuckerman, Combinatorial bounds for list decoding, IEEE Trans. Inform. Theory 48 (2002) 1021–1034.

[30] J. Naor, M. Naor, Small-bias probability spaces: Efficient constructions and applications, SIAM J. Comput. 22 (4) (1993) 838–856.

[31] N. Alon, O. Goldreich, J. Håstad, R. Peralta, Simple constructions of almost k-wise independent random variables, Random Structures Algorithms 3 (3) (1992) 289–303.

[32] M. Santha, U.V. Vazirani, Generating quasi-random sequences from semi-random sources, J. Comput. System Sci. 33 (1986) 75–87.

[33] B. Chor, O. Goldreich, Unbiased bits from sources of weak randomness and probabilistic communication complexity, SIAM J. Comput. 17 (2) (1988) 230–261.

[34] R. Raz, O. Reingold, S. Vadhan, Extracting all the randomness and reducing the error in Trevisan's extractors, in: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 149–158.