5

List-Decodable Codes

The field of *coding theory* is motivated by the problem of communicating reliably over noisy channels — where the data sent over the channel may come out corrupted on the other end, but we nevertheless want the receiver to be able to correct the errors and recover the original message. There is a vast literature studying aspects of this problem from the perspectives of electrical engineering (communications and information theory), computer science (algorithms and complexity), and mathematics (combinatorics and algebra). In this survey, we are interested in codes as "pseudorandom objects." In particular, a generalization of the notion of an error-correcting code yields a framework that we will use to unify all of the main pseudorandom objects covered in this survey (averaging samplers, expander graphs, randomness extractors, list-decodable codes, pseudorandom generators).

5.1 Definitions and Existence

5.1.1 Definition.

The approach to communicating over a noisy channel is to restrict the data we send to be from a certain set of strings that can be easily disambiguated (even after being corrupted).

Definition 5.1. A *q-ary code* is a multiset[1] $\mathcal{C} \subset \Sigma^{\hat{n}}$, where Σ is an *alphabet* of size q. Elements of $\mathcal{C}$ are called *codewords.* We define the following key parameters:

- $\hat{n}$ is the *block length.*
- $n = \log|\mathcal{C}|$ is the *message length.*
- $\rho = n/(\hat{n} \cdot \log|\Sigma|)$ is the *(relative) rate* of the code.

An *encoding function* for $\mathcal{C}$ is an bijective mapping $\mathrm{Enc}\colon \{1,\ldots,|\mathcal{C}|\} \to \mathcal{C}$. Given such an encod-

[1] Traditionally, codes are defined to be sets rather than multisets. However, the generality afforded by multisets will allow the connections we see later to be stated more cleanly. In the case $\mathcal{C}$ is a multiset, the condition that a mapping $\mathrm{Enc} : \{1,\ldots,N\} \to \mathcal{C}$ is bijective means that for every $c \in \mathcal{C}$, $|\mathrm{Enc}^{-1}(c)|$ equals the multiplicity of c in $\mathcal{C}$.

ing function, we view the elements of $\{1,\ldots,|\mathcal{C}|\}$ as *messages*. When $n = \log|\mathcal{C}|$ is an integer, we often think of messages as strings in $\{0,1\}^n$.

We view $\mathcal{C}$ and Enc as being essentially the same object (with Enc merely providing a "labelling" of codewords), with the former being useful for studying the combinatorics of codes and the latter for algorithmic purposes.

We remark that our notation differs from the standard notation in coding theory in several ways. Typically in coding theory, the input alphabet is taken to be the same as the output alphabet (rather than $\{0,1\}$ and Σ, respectively), the blocklength is denoted n, and the message length (over Σ) is denoted k and is referred to as the rate. Our notational choices are made in part to facilitate stating connections to other pseudorandom objects covered in this survey.

So far, we haven't talked at all about the error-correcting properties of codes. Here we need to specify two things: the model of errors (as introduced by the noisy channel) and the notion of a successful recovery.

For the errors, the main distinction is between *random errors* and *worst-case errors*. For random errors, one needs to specify a stochastic model of the channel. The most basic one is the *binary symmetric channel* (over alphabet $\Sigma = \{0,1\}$), where each bit is flipped independently with probability δ. People also study more complex channel models, but as usual with stochastic models, there is always the question of how well the theoretical model correctly captures the real-life distribution of errors. We, however, will focus on *worst-case errors*, where we simply assume that fewer than a δ fraction of symbols have been changed. That is, when we send a codeword $c \in \Sigma^{\hat{n}}$ over the channel, the received word $r \in \Sigma^{\hat{n}}$ differs from c in fewer than $\delta\hat{n}$ places. Equivalently, c are r are close in Hamming distance:

Definition 5.2 (Hamming distance). For two strings $x, y \in \Sigma^{\hat{n}}$, their *(relative) Hamming distance* $d_H(x,y)$ equals $\Pr_i[x_i \neq y_i]$. The *agreement* is defined to be $\mathrm{agr}(x,y) = 1 - d_H(x,y)$.

For a string $x \in \Sigma^{\hat{n}}$ and $\delta \in [0,1]$, the *(open) Hamming ball* of radius δ around x is the set $B(x,\delta)$ of strings $y \in \Sigma^{\hat{n}}$ such that $d_H(x,y) < \delta$.

For the notion of a successful recovery, the traditional model requires that we can uniquely decode the message from the received word (in the case of random errors, this need only hold with high probability). Our main focus will be on a more relaxed notion which allows us to produce a small list of candidate messages. As we will see, the advantage of such list decoding is that it allows us to correct a larger fraction of errors.

Definition 5.3. Let $\mathrm{Enc}\colon \{0,1\}^n \to \Sigma^{\hat{n}}$ be an encoding algorithm for a code $\mathcal{C}$. A *δ-decoding* algorithm for Enc is a function $\mathrm{Dec} : \Sigma^{\hat{n}} \to \{0,1\}^n$ such that for every $m \in \{0,1\}^n$ and $r \in \Sigma^{\hat{n}}$ satisfying $d_H(\mathrm{Enc}(m), r) < \delta$, we have $\mathrm{Dec}(r) = m$. If such a function Dec exists, we call the code *δ-decodable.*

A (δ, L)*-list-decoding* algorithm for Enc is a function $\mathrm{Dec} : \Sigma^{\hat{n}} \to (\{0,1\}^n)^L$ such that for every $m \in \{0,1\}^n$ and $r \in \Sigma^{\hat{n}}$ satisfying $d_H(\mathrm{Enc}(m), r) < \delta$, we have $m \in \mathrm{Dec}(r)$. If such a function *Dec* exists, we call the code (δ, L)*-list-decodable.*

Note that a δ-decoding algorithm is the same as a $(\delta, 1)$-list-decoding algorithm. It is not hard to see that, if we do not care about computational efficiency, the existence of such decoding algorithms depends only on the combinatorics of the set of codewords.

Definition 5.4. The *(relative) minimum distance* of a code $\mathcal{C} \subset \Sigma^{\hat{n}}$ equals $\min_{x \neq y \in \mathcal{C}} d_H(x, y)$.[2]

Proposition 5.5. Let $\mathcal{C} \subset \Sigma^{\hat{n}}$ be a code with any encoding function Enc.

(1) For $\delta\hat{n} \in \mathbb{N}$, Enc is δ-decodable iff its minimum distance is at least $2\delta - 1/\hat{n}$.
(2) Enc is (δ, L)-list-decodable iff for every $r \in \Sigma^{\hat{n}}$, we have $|B(r, \delta) \cap \mathcal{C}| \leq L$.

Proof. Item 2 follows readily from the definitions.

For Item 1, first note that Enc is δ-decodable iff there is no received word $r \in \Sigma^{\hat{n}}$ at distance less than δ from two codewords c_1, c_2. (Such an r should decode to both c_1 and c_2, which is impossible for a unique decoder.) If the code has minimum distance at least $2\delta - 1/\hat{n}$, then c_1 and c_2 disagree in at least $2\delta\hat{n} - 1$ positions, which implies that r disagrees with one of them in at least $\delta\hat{n}$ positions. (Recall that $\delta\hat{n}$ is an integer by hypothesis.) Conversely, if the code has minimum distance smaller than $2\delta - 1/\hat{n}$, then there are two codewords that disagree in at most $2\delta\hat{n} - 2$ positions, and we can construct r that disagrees with each in at most $\delta\hat{n} - 1$ positions. □

Because of the factor of 2 in Item 1, unique decoding is only possible at distances up to 1/2, whereas we will see that list decoding is possible at distances approaching 1 (with small lists).

The main goals in constructing codes are to have infinite families of codes (e.g. for every message length n) in which we:

- Maximize the fraction δ of errors correctible (e.g. constant independent of n and $\hat{n}$).
- Maximize the rate ρ (e.g. a constant independent of n and $\hat{n}$).
- Minimize the alphabet size q (e.g. a constant, ideally $q = 2$).
- Keep the list size L relatively small (e.g. a constant or $\text{poly}(n)$).
- Have computationally efficient encoding and decoding algorithms.

In particular, coding theorists are very interested in obtaining the optimal tradeoff between the constants δ and ρ with efficiently encodable and decodable codes.

5.1.2 Existential Bounds

Like expanders, the existence of very good codes can be shown using the probabilistic method. The bounds will be stated in terms of the q-ary entropy functions, so we begin by defining those.

Definition 5.6. For $q, \hat{n} \in \mathbb{N}$ and $\delta \in [0, 1]$, we define $H_q(\delta, \hat{n}) \in [0, 1]$ to be such that $|B(x, \delta)| = q^{H_q(\delta,\hat{n})\cdot\hat{n}}$ for $x \in \Sigma^{\hat{n}}$, where Σ is an alphabet of size q.

We also define the *q-ary entropy function* $H_q(\delta) = \delta \cdot \log_q((q - 1)/\delta) + (1 - \delta) \cdot \log_q(1/(1 - \delta))$.

The reason we use similar notation for $H_q(\delta, \hat{n})$ and $H_q(\delta)$ is Part 1 of the following:

[2] If any codeword appears in $\mathcal{C}$ with multiplicity greater than 1, then the minimum distance is defined to be zero.

Proposition 5.7. For every $q \in \mathbb{N}$, $\delta \in (0, 1-1/q)$, $\varepsilon \in [0, 1/2]$,

(1) $\lim_{\hat{n}\to\infty} H_q(\delta, \hat{n}) = H_q(\delta)$.
(2) $H_q(\delta) \le H_2(\delta)/\log q + \delta$.
(3) $H_2(1/2 - \varepsilon) = 1 - \Theta(\varepsilon^2)$.

Now we state the bounds for random error-correcting codes.

Theorem 5.8. (1) For all $\hat{n}, q \in \mathbb{N}$ and $\delta \in (0, 1-1/q)$, there exists a q-ary code of block length $\hat{n}$, minimum distance at least δ, and rate at least $\rho = 1 - H_q(\delta, \hat{n})$.
(2) For all integers $\hat{n}, q, L \in \mathbb{N}$ and $\delta \in (0, 1-1/q)$, there exists a (δ, L)-list-decodable q-ary code of block length $\hat{n}$ and rate at least $\rho = 1 - H_q(\delta, \hat{n}) - 1/(L+1)$.

Proof. (1) Pick the codewords $c_1, \ldots, c_N$ in sequence ensuring that c_i is at distance at least δ from $c_1, \ldots, c_{i-1}$. The union of the Hamming balls of radius δ around $c_1, \ldots, c_{i-1}$ contains at most $(i-1) \cdot q^{H_q(\delta,\hat{n})\cdot\hat{n}} < (N-1) \cdot q^{H_q(\delta,\hat{n})\cdot\hat{n}}$, so there is always a choice of c_i outside these balls if we take $N = \lceil q^{(1-H_q(\delta,\hat{n}))\cdot\hat{n}} \rceil$.
(2) We use the probabilistic method. Choose the N codewords randomly and independently from $\Sigma^{\hat{n}}$. The probability that there is a Hamming Ball of radius δ containing at least $L+1$ codewords is at most

$$q^{\hat{n}} \cdot \binom{N}{L+1} \cdot \left(\frac{q^{H_q(\delta,\hat{n})\cdot\hat{n}}}{q^{\hat{n}}} \right)^{L+1} \le \left(\frac{N-1}{q^{(1-H_q(\delta,\hat{n})-1/(L+1))\cdot\hat{n}}} \right)^{L+1},$$

which is less than 1 if we take $N = \lceil q^{(1-H_q(\delta,\hat{n})-1/(L+1))\cdot\hat{n}} \rceil$.

□

Note that while the rate bounds are essentially the same for achieving minimum distance δ and achieving list-decoding radius δ (as we take large list size), the bounds are incomparable because minimum distance δ only corresponds to unique decoding up to radius roughly $\delta/2$. The bound for list decoding is known to be tight up to the dependence on L (Problem 5.1), while the bound on minimum distance is not tight in general. Indeed, there are families of "algebraic-geometric" codes with constant alphabet size q, constant minimum distance $\delta > 0$, and constant rate $\rho > 0$ where $\rho > 1 - H_q(\delta, \hat{n})$ for sufficiently large $\hat{n}$. (Thus, this is a rare counterexample to the phenomenon "random is best".) Identifying the best tradeoff between rate and minimum distance, even for binary codes, is a long-standing open problem in coding theory.

Open Problem 5.9. For each constant $\delta \in (0, 1)$, identify the largest $\rho > 0$ such that for every $\varepsilon > 0$, there exists an infinite family of codes $\mathcal{C}_{\hat{n}} \subset \{0,1\}^{\hat{n}}$ of rate at least $\rho - \varepsilon$ and minimum distance at least δ.

Let's look at some special cases of the parameters in Theorem 5.8. For binary codes ($q = 2$), we will be most interested in the case $\delta \to 1/2$, which corresponds to correcting the maximum possible fraction of errors for binary codes. (No nontrivial decoding is possible for binary codes at distance greater than $1/2$, since a completely random received word will be at distance roughly $1/2$ with most codewords.) In this case, Proposition 5.7 tells us that the rate approaches $1 - H_2(1/2-\varepsilon) = \Theta(\varepsilon^2)$, i.e. the blocklength is $\hat{n} = \Theta(n/\varepsilon^2)$ (for list size $L = \Theta(1/\varepsilon^2)$). For large alphabets q, Proposition 5.7 tells us that the rate approaches $1-\delta$ as q grows. We will be most interested in the case $\delta = 1-\varepsilon$ for small ε, so we can correct a $\delta = 1-\varepsilon$ fraction of errors with a rate arbitrarily close to ε. For example, we can achieve rate of $\rho = .99\varepsilon$, a list size of $L = O(1/\varepsilon)$ and an alphabet of size $\mathrm{poly}(1/\varepsilon)$. More generally, it is possible to achieve rate $\rho = (1-\gamma)\varepsilon$ with an alphabet size of $q = (1/\varepsilon)^{O(1/\gamma)}$.

While we are primarily interested in list-decodable codes, minimum distance is often easier to bound. The following allows us to translate bounds on minimum distance into bounds on list-decodability.

Proposition 5.10 (Johnson Bound). (1) If $\mathcal{C}$ has minimum distance $1-\varepsilon$, then it is $(1-O(\sqrt{\varepsilon}), O(1/\sqrt{\varepsilon}))$-list-decodable.

(2) If a *binary* code $\mathcal{C}$ has minimum distance $1/2-\varepsilon$, then it is $(1/2 - O(\sqrt{\varepsilon}), O(1/\varepsilon))$-list-decodable.

Proof. We prove Item 1, and leave Item 2 as an exercise in the next chapter (Problem 6.3). The proof is by inclusion-exclusion. Suppose for contradiction that there are codewords $c_1, \ldots, c_s$ at distance less than $1-\varepsilon'$ from some $r \in \Sigma^{\hat{n}}$, for $\varepsilon' = 2\sqrt{\varepsilon}$ and $s = \lceil 2/\varepsilon' \rceil$. Then:

$$\begin{aligned} 1 &\geq \text{fraction of positions where } r \text{ agrees with some } c_i \\ &\geq \sum_i \mathrm{agr}(r, c_i) - \sum_{1 \leq i < j \leq s} \mathrm{agr}(c_i, c_j) \\ &> s\varepsilon' - \binom{s}{2} \cdot \varepsilon \\ &\geq 2 - 1 = 1 \end{aligned}$$

where the last inequality is by our setting of parameters. Contradiction. □

Note the quadratic loss in the distance parameter. This means that optimal codes with respect to minimum distance are not necessarily optimal with respect to list decoding. Nevertheless, if we do not care about the exact tradeoff between the rate and the decoding radius, the above can yield codes where the decoding radius is as large as possible (approaching 1 for large alphabets and $1/2$ for binary alphabets).

5.1.3 Explicit Codes

As usual, most applications of error-correcting codes (in particular the original motivating one) require computationally efficient encoding and decoding. For now, we focus on only the efficiency of encoding.

Definition 5.11. A code $\mathrm{Enc} : \{0,1\}^n \to \Sigma^{\hat{n}}$ is *(fully) explicit* if given a message $m \in \{0,1\}^n$ and an index $i \in \hat{n}$, the i'th symbol of $\mathrm{Enc}(m)$ can be computed in time $\mathrm{poly}(n, \log \hat{n}, \log |\Sigma|)$.

The reason we talk about computing individual symbols of the codeword rather than the entire codeword is to have a meaningful definition even for codes where the blocklength $\hat{n}$ is superpolynomial in the message length n. One can also consider weaker notions of explicitness, where we simply require that the entire codeword $\mathrm{Enc}(m)$ can be computed in time $\mathrm{poly}(\hat{n}, \log|\Sigma|)$.

The constructions of codes that we describe will involve arithmetic over finite fields. See Remark 3.25 for the complexity of constructing finite fields and carrying out arithmetic in such fields .

In describing the explicit constructions below, it will be convenient to think of the codewords as functions $c : [\hat{n}] \to \Sigma$ rather than as strings in $\Sigma^{\hat{n}}$.

Construction 5.12 (Hadamard Code). For $n \in \mathbb{N}$, the *(binary) Hadamard code* of message length n is the binary code of blocklength $\hat{n} = 2^n$ consisting of all functions $c : \mathbb{Z}_2^n \to \mathbb{Z}_2$ that are *linear* (modulo 2).

Proposition 5.13. The Hadamard code:

(1) is explicit with respect to the encoding function that takes a message $m \in \mathbb{Z}_2^n$ to the linear function c_m defined by $c_m(x) = \sum_i m_i x_i \bmod 2$.
(2) has minimum distance $1/2$, and
(3) is $(1/2 - \varepsilon, O(1/\varepsilon^2))$ list-decodable for every $\varepsilon > 0$.

Proof. Explicitness is clear by inspection. The minimum distance follows from the fact that for every two distinct linear functions $c, c' : \mathbb{Z}_2^n \to \mathbb{Z}_2$, $\Pr_x[c(x) = c'(x)] = \Pr_x[(c - c')(x) = 0] = 1/2$. The list-decodability follows from the Johnson Bound. □

The advantages of the Hadamard code are its small alphabet (binary) and optimal distance (1/2), but unfortunately its rate is exponentially small ($\rho = n/2^n$). By increasing both the field size and degree, we can obtain complementary properties

---

Construction 5.14 (Reed–Solomon Codes). For a prime power q and $d \in \mathbb{N}$, the q-ary *Reed–Solomon code* of degree d is the code of blocklength $\hat{n} = q$ and message length $n = (d+1) \cdot \log q$ consisting of all polynomials $p : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most d.

Proposition 5.15. The q-ary Reed–Solomon Code of degree d:

(1) is explicit with respect to the encoding function that takes a vector of coefficients $m \in \mathbb{F}_q^{d+1}$ to the polynomial p_m defined by $p_m(x) = \sum_{i=0}^{d} m_i x^i$ (assuming we have a description of the field $\mathbb{F}_q$).
(2) has minimum distance $\delta = 1 - d/q$, and
(3) is $(1 - O(\sqrt{d/q}), O(\sqrt{q/d}))$ list-decodable.

Proof. Again explicitness follows by inspection. The minimum distance follows from the fact that two distinct polynomials of degree at most d agree in at most d points (else their difference would have more than d roots). The list-decodability follows from the Johnson Bound. □

Note that by setting $q = O(d)$, Reed-Solomon codes simultaneously achieve constant rate and constant distance, the only disadvantage being that the alphabet is of nonconstant size (namely $q = \hat{n} > n$.)

Another useful setting of parameters for Reed-Solomon codes in complexity theory is $q = \mathrm{poly}(d)$, which gives polynomial rate ($\hat{n} = \mathrm{poly}(n)$) and distance tending to 1 polynomially fast ($\delta = 1 - 1/\mathrm{poly}(n)$).

The following codes "interpolate" between Hadamard and Reed-Solomon codes, by allowing the number of variables, the degree, and field size all to vary.

Construction 5.16 (Reed–Muller Codes). For a prime power q and $d, m \in \mathbb{N}$, the q-ary *Reed–Muller code* of degree d and dimension m is the code of blocklength $\hat{n} = q^m$ and message length $n = \binom{m+d}{d} \cdot \log q$ consisting of all polynomials $p : \mathbb{F}_q^m \to \mathbb{F}_q$ of (total) degree at most d.

Proposition 5.17. The q-ary Reed–Muller Code of degree d and dimension m:

(1) is explicit with respect to the encoding function that takes a vector of coefficients $v \in \mathbb{F}_q^{\binom{m+d}{d}}$ to the corresponding polynomial p_v.
(2) has minimum distance $\delta \geq 1 - d/q$, and
(3) is $(1 - O(\sqrt{d/q}), O(\sqrt{q/d}))$ list-decodable.

Proof. Same as for Reed–Solomon Codes, except we use the Schwartz-Zippel Lemma (Lemma 2.4) to deduce the minimum distance. □

Note that Reed–Solomon Codes are simply Reed–Muller codes of dimension $m = 1$, and Hadamard codes are essentially Reed–Muller codes of degree $d = 1$ and alphabet size $q = 2$ (except that the Reed–Muller code also contains affine linear functions).

5.2 List-Decoding Algorithms

In this section, we will describe efficient list-decoding algorithms for the Reed–Solomon code and variants. It will be convenient to work with the following notation:

Definition 5.18. Let $\mathcal{C}$ be a code with encoding function $\mathrm{Enc} : \{1, \ldots, N\} \to \Sigma^{\hat{n}}$. For $r \in \Sigma^{\hat{n}}$, define $\mathrm{LIST}(r, \varepsilon) = \{m \in \Sigma^{\hat{n}} : \mathrm{agr}(\mathrm{Enc}(m), r) > \varepsilon\}$.

Then the task of $(1 - \varepsilon)$ list decoding (according to Definition 5.3) is equivalent to producing the elements of $\mathrm{LIST}(r, \varepsilon)$ given $r \in \Sigma^{\hat{n}}$. In this section, we will see algorithms that do this in time polynomial in the bit-length of r, i.e. time $\mathrm{poly}(\hat{n}, \log |\Sigma|)$.

5.2.1 Review of Algebra

The list-decoding algorithms will require some additional algebraic facts and notation:

- For every field $\mathbb{F}$, $\mathbb{F}[X_1, \ldots, X_n]$ is the integral domain consisting of formal polynomials $Q(X_1, \ldots, X_n)$ with coefficients in $\mathbb{F}$, where addition and multiplication of polynomials is defined in the usual way.
- A nonzero polynomial $Q(X_1, \ldots, X_n)$ is *irreducible* if we cannot write $Q = RS$ where R, S are nonconstant polynomials. For a finite field $\mathbb{F}_q$ of characteristic p and $d \in \mathbb{N}$, a univariate irreducible polynomial of degree d over $\mathbb{F}_q$ can be found in deterministically in time $\mathrm{poly}(p, \log q, d)$.
- $\mathbb{F}[X_1, \ldots, X_n]$ is a *unique factorization domain.* That is, every nonzero polynomial Q can be factored as $Q = Q_1 Q_2 \cdots Q_m$, where each Q_i is irreducible and this factorization is unique up to reordering and multiplication by constants from $\mathbb{F}$. Given the description of a finite field $\mathbb{F}_{p^k}$ and the polynomial Q, this factorization can be done probabilistically in time $\mathrm{poly}(\log p, k, |Q|)$ and deterministically in time $\mathrm{poly}(p, k, |Q|)$.
- For a nonzero polynomial $Q(Y, Z) \in \mathbb{F}[Y, Z]$ and $f(Y) \in \mathbb{F}[Y]$, if $Q(Y, f(Y)) = 0$, then $Z - f(Y)$ is one of the irreducible factors of $Q(Y, Z)$ (and thus $f(Y)$ can be found in polynomial time). This is analogous to the fact that if $c \in \mathbb{Z}$ is a root of an integer polynomial $Q(Z)$, then $Z - c$ is one of the factors of Q (and can be proven in the same way, by long division).

5.2.2 List-Decoding Reed-Solomon Codes

~~**Theorem 5.19.** There is a polynomial-time $(1 - \varepsilon)$ list-decoding algorithm for the q-ary Reed-~~ Solomon code of degree d, for $\varepsilon = 2\sqrt{d/q}$. That is, given a function $r : \mathbb{F}_q \to \mathbb{F}_q$ and $d \in \mathbb{N}$, all polynomials of degree at most d that agree with r on more than $\varepsilon q = 2\sqrt{dq}$ inputs can be found in polynomial time.

In fact the constant of 2 can be improved to 1, matching the combinatorial list-decoding radius for Reed–Solomon codes given by an optimized form of the Johnson Bound. See Problem 5.6.

Proof. We are given a received word $r : \mathbb{F}_q \to \mathbb{F}_q$, and want to find all elements of $\mathrm{LIST}(r, \varepsilon)$ for $\varepsilon = 2\sqrt{d/q}$.

Step 1: Find a low-degree Q "explaining" r. Specifically, $Q(Y, Z)$ will be a nonzero bivariate polynomial of degree at most d_Y in its first variable Y and d_Z in its second variable, with the property that $Q(y, r(y)) = 0$ for all $y \in \mathbb{F}_q$. Each such y imposes a linear constraint on the $(d_Y+1)(d_Z+1)$ coefficients of Q. Thus, this system has a nonzero solution provided $(d_Y+1)(d_Z+1) > q$, and it can be found in polynomial time by linear algebra (over $\mathbb{F}_q$).

Step 2: Argue that each $f(Y) \in \mathrm{LIST}(r, \varepsilon)$ is a "root" of Q. Specifically, it will be the case that $Q(Y, f(Y)) = 0$ for each $f \in \mathrm{LIST}(r, \varepsilon)$. The reason is that $Q(Y, f(Y))$ is a univariate polynomial of degree at most $d_Y + d \cdot d_Z$, and has more than εq zeroes (one for each place that f and r agree). Thus, we can conclude $Q(Y, f(Y)) = 0$ provided $\varepsilon q \geq d_Y + d \cdot d_Z$. Then we can enumerate all of the elements of $\mathrm{LIST}(r, \varepsilon)$ by factoring $Q(Y, Z)$ and taking all the factors of the form $Z - f(Y)$.

For this algorithm to work, the two conditions we need to satisfy are

$$(d_Y + 1)(d_Z + 1) > q,$$

and

$$\varepsilon q \geq d_Y + d \cdot d_Z.$$

These conditions can be satisfied by setting $d_Y = \lfloor \varepsilon q/2 \rfloor$, $d_Z = \lfloor \varepsilon q/(2d) \rfloor$, and $\varepsilon = 2\sqrt{d/q}$. □

Note that the rate of Reed-Solomon codes is $\rho = (d+1)/q = \Theta(\varepsilon^2)$. The alphabet size is $q = \tilde{\Omega}(n/\rho) = \tilde{\Omega}(n/\varepsilon^2)$. In contrast, the random codes of Theorem 5.8 achieve $\rho \approx \varepsilon$ and $q = \mathrm{poly}(1/\varepsilon)$. It is not known whether the $\varepsilon = \sqrt{d/q}$ bound on the list-decodability of Reed–Solomon codes can be improved, even with inefficient decoding.

Open Problem 5.20. Do there exist constants $\varepsilon, \rho \in (0, 1)$ with $\varepsilon < \sqrt{\rho}$ and an infinite sequence of prime powers q such that the q-ary Reed–Solomon code of degree $d = \lfloor \rho q \rfloor$ is $(1 - \varepsilon, \mathrm{poly}(q))$-list-decodable?

5.2.3 Parvaresh–Vardy Codes

Our aim is to improve the rate-distance tradeoff to $\rho = \tilde{\Theta}(\varepsilon)$. Intuitively, the power of the Reed–Solomon list-decoding algorithm comes from the fact that we can interpolate the q points $(y, r(y))$ of the received word using a *bivariate* polynomial Q of degree roughly $\sqrt{q}$ in each variable (think of $d = O(1)$ for now). If we could use m variables instead of 2, then the degrees would only have to be around $q^{1/m}$.

First attempt: Replace Step 1 with finding an $(m+1)$-variate polynomial $Q(Y, Z_1, \ldots, Z_m)$ of degree d_Y in Y and d_Z in each Z_i such that $Q(y, r(y), r(y), \ldots, r(y)) = 0$ for every $y \in \mathbb{F}_q$. Then, we will be able to choose a nonzero polynomial Q of degree roughly $q^{1/m}$ such that $Q(Y, f(Y), \ldots, f(Y)) = 0$ for every $f \in \mathrm{LIST}(r, \varepsilon)$, and hence it follows that $Z - f(Y)$ divides the bivariate polynomial $Q^*(Y, Z) = Q(Y, Z, \ldots, Z)$. Unfortunately, Q^* might be the zero polynomial even if Q is nonzero.

Second attempt: Replace Step 1 with finding an $(m+1)$-variate polynomial $Q(Y, Z_1, \ldots, Z_m)$ of degree d_Y in Y and $d_Z = h - 1$ in each Z_i such that $Q(y, r(y), r(y)^h, r(y)^{h^2}, \ldots, r(y)^{h^{m-1}}) = 0$ for every $y \in \mathbb{F}_q$. Then, it follows that $Q^*(Y, Z) = Q(Y, Z, Z^h, \ldots, Z^{h^{m-1}})$ is nonzero if Q is nonzero because every monomial in Q with individual degrees at most $h - 1$ in $Z_1, \cdots, Z_m$ gets mapped to a different power of Z. However, here the difficulty is that the degree of $Q^*(Y, f(Y))$ is too high (roughly $d^* = d_Y + d \cdot h^m > d_Z^m$) for us to satisfy the constraint $\varepsilon q \geq d^*$.

Parvaresh–Vardy codes get the best of both worlds by providing more information with each symbol — not just the evaluation of f at each point, but the evaluation of $m - 1$ other polynomials $f_1, \ldots, f_{m-1}$, each of which is still of degree d (as is good for arguing that

$Q(Y, f(Y), f_1(Y), \ldots, f_{m-1}(Y)) = 0$, but can be viewed as raising f to successive powers of h for the purposes of ensuring that Q^* is nonzero.

To introduce this idea, we need some additional algebra.

- For univariate polynomials $f(Y)$ and $E(Y)$, we define $f(Y) \bmod E(Y)$ to be the remainder when f is divided by E. If $E(Y)$ is of degree k, then $f(Y) \bmod E(Y)$ is of degree at most $k-1$.
- The ring $\mathbb{F}[Y]/E(Y)$ consists of all polynomials of degree at most $k-1$ with arithmetic modulo $E(Y)$ (analogous to $\mathbb{Z}_n$ consisting of integers smaller than n with arithmetic modulo n). If E is irreducible, then $\mathbb{F}[Y]/E(Y)$ is a field (analogous to $\mathbb{Z}_p$ being a field when p is prime). Indeed, this is how the finite field of size p^k is constructed: take $\mathbb{F} = \mathbb{Z}_p$ and $E(Y)$ to be an irreducible polynomial of degree k over $\mathbb{Z}_p$, and then $\mathbb{F}[Y]/E(Y)$ is the (unique) field of size p^k.
- A multivariate polynomial $Q(Y, Z_1, \ldots, Z_m)$ can be reduced modulo $E(Y)$ by writing it as a polynomial in variables $Z_1, \ldots, Z_m$ with coefficients in $\mathbb{F}[Y]$ and then reducing each coefficient modulo $E(Y)$. After reducing Q modulo E, we think of Q as a polynomial in variables $Z_1, \ldots, Z_m$ with coefficients in the field $\mathbb{F}[Y]/E(Y)$.

Construction 5.21 (Parvaresh–Vardy Codes). For a prime power q, integers $m, d, h \in \mathbb{N}$, and an irreducible polynomial $E(Y)$ of degree larger than d, the q-ary *Parvaresh–Vardy Code* of degree d, power h, redundancy m, and irreducible E is defined as follows:

- The alphabet is $\Sigma = \mathbb{F}_q^m$.
- The blocklength is $\hat{n} = q$.
- The message space is $\mathbb{F}_q^{d+1}$, where we view each message as representing a polynomial $f(Y)$ of degree at most d over $\mathbb{F}_q$.
- For $y \in \mathbb{F}_q$, the y'th symbol of the $\mathrm{Enc}(f)$ is

$$[f_0(y), f_1(y), \ldots, f_{m-1}(y)],$$

  where $f_i(Y) = f(Y)^{h^i} \bmod E(Y)$.

Theorem 5.22. For every prime power q, integer $0 \le d < q$, and irreducible polynomial E of degree $d+1$, the q-ary Parvaresh–Vardy code of degree d, redundancy $m = \lceil \log(q/d) \rceil$, power $h = 2$, and irreducible E has rate $\rho = \tilde{\Omega}(d/q)$ and can be list-decoded in polynomial time up to distance $\delta = 1 - \tilde{O}(d/q)$.

Proof. We are given a received word $r : \mathbb{F}_q \to \mathbb{F}_q^m$, and want to find all elements of $\mathrm{LIST}(r, \varepsilon)$, for some $\varepsilon = \tilde{O}(d/q)$.

Step 1: Find a low-degree Q "explaining" r. We find a nonzero polynomial $Q(Y, Z_0, \ldots, Z_{m-1})$ of degree at most d_Y in its first variable Y and at most $h-1$ in each of the remaining variables, and satisfying $Q(y, r(y)) = 0$ for all $y \in \mathbb{F}_q$.

Such a Q exists and can be found by linear algebra in time $\mathrm{poly}(q, m)$ provided that:

$$d_Y \cdot h^m > q. \tag{5.1}$$

Moreover, we may assume that Q is not divisible by $E(Y)$. If it is, we can divide out all the factors of $E(Y)$, which will not affect the conditions $Q(y, r(y)) = 0$ since E has no roots (being irreducible).

Step 2: Argue that each $f(Y) \in \mathrm{LIST}(r, \varepsilon)$ is a "root" of a related univariate polynomial Q^*. First, we argue as before that for $f \in \mathrm{LIST}(r, \varepsilon)$, we have

$$Q(Y, f_0(Y), \ldots, f_{m-1}(Y)) = 0. \tag{5.2}$$

Since each f_i has degree at most $\deg(E) - 1 = d$, this will be ensured provided

$$\varepsilon q \geq d_Y + (h-1) \cdot d \cdot m. \tag{5.3}$$

Once we have this, we can reduce both sides of Equation (5.2) modulo $E(Y)$ and deduce

$$\begin{aligned} 0 &= Q(Y, f_0(Y), f_1(Y), \ldots, f_{m-1}(Y)) \bmod E(Y) \\ &= Q(Y, f(Y), f(Y)^h, \ldots, f(Y)^{h^{m-1}}) \bmod E(Y) \end{aligned}$$

Thus, if we define the univariate polynomial

$$Q^*(Z) = Q(Y, Z, Z^h, \ldots, Z^{h^{m-1}}) \bmod E(Y),$$

then $f(Y)$ is a root of Q^* over the field $\mathbb{F}_q[Y]/E(Y)$.

Observe that Q^* is nonzero because Q is not divisible by $E(Y)$ and has degree at most $h-1$ in each Z_i. Thus, we can find all elements of $\mathrm{LIST}(r, \varepsilon)$ by factoring $Q^*(Z)$. Constructing $Q^*(Z)$ and factoring it can be done in time $\mathrm{poly}(q, d, h^m) = \mathrm{poly}(q)$.

For this algorithm to work, we need to satisfy Conditions (5.1) and (5.3). We can satisfy Condition (5.3) by setting $d_Y = \lceil \varepsilon q - dhm \rceil$, in which case Condition (5.1) is satisfied for

$$\varepsilon = \frac{1}{h^m} + \frac{dhm}{q} = \tilde{O}(d/q), \tag{5.4}$$

for $h = 2$ and $m = \lceil \log(q/d) \rceil$. Observing that the rate is $\rho = d/(mq) = \tilde{\Omega}(d/q)$, this completes the proof of the theorem. □

Note that the obstacles to obtaining an tradeoff $\rho \approx \varepsilon$ are the factor of m in expression for the rate $\rho = d/(mq)$ and the factor of hm in Equation (5.4) for ε. We remedy these in the next section.

5.2.4 Folded Reed–Solomon Codes

In this section, we show how to obtain an optimal rate-distance tradeoff, where the rate ρ is arbitrarily close to the agreement parameter ε.

Consider the Parvaresh–Vardy construction with polynomial $E(Y) = Y^{q-1} - \gamma$, where γ is a generator of the multiplicative group $\mathbb{F}_q^*$; it can be shown that $E(Y)$ is irreducible. (That is, $\{\gamma, \gamma^2, \ldots, \gamma^{q-1}\} = \mathbb{F}_q \backslash \{0\}$.) Then it turns out that $f^q(Y) \bmod E(Y) = f(Y^q) \bmod E(Y) = f(\gamma Y)$.

So, we set $h = q$ and the degree of $f_i(Y) = f^{h^i}(Y) \bmod E(Y) = f(\gamma^i Y)$ is d even though E has degree $q-1$. For each nonzero element y of $\mathbb{F}_q$, the y'th symbol of the PV encoding of $f(Y)$ is then

$$[f(y), f(\gamma y), \ldots, f(\gamma^{m-1}y)] = [f(\gamma^j), f(\gamma^{j+1}), \ldots, f(\gamma^{j+m-1})], \tag{5.5}$$

where we write $y = \gamma^j$.

Thus, the symbols of the PV encoding have a lot of overlap. For example, the γ^j'th symbol and the γ^{j+1}'th symbol share all but one component. Intuitively, this means that we should only have to send a $1/m$ fraction of the symbols of the codeword, saving us a factor of m in the rate. (The other symbols can be automatically filled in by the receiver.) Thus, the rate becomes $\rho = d/q$, just like in Reed–Solomon codes.

More formally, we use the following codes.

Construction 5.23 (Folded Reed–Solomon Codes). For a prime power q, a generator γ of $\mathbb{F}_q^*$, integers $m, d \in \mathbb{N}$, the q-ary *folded Reed–Solomon code* of degree d, folding parameter m, and generator γ is defined as follows:

- The alphabet is $\Sigma = \mathbb{F}_q^m$.
- The blocklength is $\hat{n} = \lfloor (q-1)/m \rfloor$.
- The message space is $\mathbb{F}_q^{d+1}$, where we view each message as representing a polynomial $f(Y)$ of degree at most d over $\mathbb{F}_q$.
- For $k \in [\hat{n}]$, the k'th symbol of $\mathrm{Enc}(f)$ is

$$[f(\gamma^{(k-1)m}), f(\gamma^{(k-1)m+1}), \ldots, f(\gamma^{k\cdot m-1})].$$

We now show that these codes can be efficiently list-decoded at distance arbitrarily close to $1-\rho$, where $\rho = d/q$ is the rate.

Theorem 5.24. For every prime power q, integers $0 \le d, m \le q$, and generator γ of $\mathbb{F}_q^*$, the q-ary folded Reed–Solomon code of degree d, folding parameter m, and generator γ has rate at least $\rho = d/q$ and can be list-decoded in time $q^{O(m)}$ up to distance $\delta = 1 - d/q - O(\sqrt{1/m})$.

Note that this theorem is most interesting (and improves on the Reed–Solomon decoding of Theorem 5.19) when m is larger than q/d, in contrast to the setting of parameters in Theorem 5.22, where m is logarithmic in q. We can afford the larger setting of m because now the rate ρ does not depend on m, and it is this change in parameters that enables us to save us the factor of hm in Equation (5.4). However, the running time, list size, and alphabet size do grow exponentially with m.

Proof. We are given a received word $r : [\hat{n}] \to \mathbb{F}_q^m$, and want to find all elements of $\mathrm{LIST}(r, \varepsilon)$ for an appropriate choice of $\varepsilon = d/q + O(\sqrt{1/m})$.

Step 0: Unpack to a PV received word. From r, we will obtain a received word $r' : \mathbb{F}_q \to \mathbb{F}_q^{m'}$ for the Parvaresh–Vardy code of degree d, power $h = q$, redundancy $m' = \lfloor \sqrt{m-1} \rfloor$, and irreducible

$E(Y) = Y^{q-1} - \gamma$. Specifically, following Equation (5.5), each symbol of r yields $m - m'$ symbols of r'. If r agrees with the FRS encoding of f in more than $\varepsilon\hat{n}$ positions, then r' will agree with the PV encoding of f in more than $\varepsilon\hat{n} \cdot (m - m')$ positions. We have

$$\varepsilon\hat{n} \cdot (m - m') \geq \varepsilon \cdot ((q-1)/m - 1) \cdot (m - m') = (\varepsilon - O(1/\sqrt{m})) \cdot q.$$

Thus our task is now to find $\mathrm{LIST}(r', \varepsilon')$ for $\varepsilon' = \varepsilon - O(1/\sqrt{m})$, which we do in a similar manner to the decoding algorithm for Parvaresh–Vardy codes.

Step 1: Find a low-degree Q "explaining" r'. We find a nonzero polynomial $Q(Y, Z_0, \ldots, Z_{m'-1})$ satisfying $Q(y, r'(y)) = 0$ for all $y \in \mathbb{F}_q$ and such that Q has degree at most d_Y in its first variable Y and *total degree* at most $d_Z = 1$ in the Z_i variables. That is, Q only has monomials of the form $Y^j Z_i$ for $j \leq d_Y$. The number of these monomials is $(d_Y + 1) \cdot m'$, so we can find such a Q in time $\mathrm{poly}(q, m') = \mathrm{poly}(q, m)$ provided:

$$(d_Y + 1) \cdot m' > q. \tag{5.6}$$

As before, we may assume that Q is not divisible by $E(Y)$. (Note that using total degree 1 instead of individual degrees $h-1$ in the Z_i's requires an exponentially larger setting of m' compared to the setting of m needed for Equation (5.1). However, this will provide a significant savings below. In general, it is best to consider a combination of the two constraints, requiring that the total degree in the Z_i's is at most some d_Z *and* that the individual degrees are all at most $h - 1$.)

Step 2: Argue that each $f(Y) \in \mathrm{LIST}(r', \varepsilon')$ is a "root" of a related univariate polynomial Q^*. First, we argue as before that for $f \in \mathrm{LIST}(r', \varepsilon')$, we have

$$Q(Y, f_0(Y), \ldots, f_{m'-1}(Y)) = 0, \tag{5.7}$$

where $f_i(Y) = f^{h^i}(Y) \bmod E(Y) = f(\gamma^i Y)$. Since each f_i has degree at most d, this will be ensured provided

$$\varepsilon' q \geq d_Y + d. \tag{5.8}$$

Note the savings of the factor $(h-1) \cdot m$ as compared to Inequality (5.3); this is because we chose Q to be of total degree 1 in the Z_i's instead of having individual degree 1.

Now, as in the Parvaresh–Vardy decoding, we can reduce both sides of Equation (5.7) modulo $E(Y)$ and deduce

$$\begin{aligned} 0 &= Q(Y, f_0(Y), f_1(Y), \ldots, f_{m-1}(Y)) \bmod E(Y) \\ &= Q(Y, f(Y), f(Y)^h, \ldots, f(Y)^{h^{m-1}}) \bmod E(Y). \end{aligned}$$

Thus, if we define the univariate polynomial

$$Q^*(Z) = Q(Y, Z, Z^h, \ldots, Z^{h^{m-1}}) \bmod E(Y),$$

then $f(Y)$ is a root of Q^* over the field $\mathbb{F}_q[Y]/E(Y)$.

Observe that Q^* is nonzero because Q is not divisible by $E(Y)$ and has degree at most $h - 1$ in each Z_i. Thus, we can find all elements of $\mathrm{LIST}(r', \varepsilon')$ by factoring $Q^*(Z)$. Constructing and factoring $Q^*(Z)$ can be done in time $\mathrm{poly}(q, h^m) = q^{O(m)}$.

For this algorithm to work, we need to satisfy Conditions (5.6) and (5.8). We can satisfy Condition (5.6) by setting $d_Y = \lfloor q/m' \rfloor$, in which case Condition (5.8) is satisfied for

$$\varepsilon' \geq 1/m' + d/q.$$

Recalling that $\varepsilon' = \varepsilon - O(1/\sqrt{m})$ and $m' = \lfloor \sqrt{m-1} \rfloor$, we can take $\varepsilon = d/q + O(1/\sqrt{m})$. $\square$

Setting parameters appropriately gives:

Theorem 5.25. The following holds for all *constants* $\varepsilon, \gamma > 0$. For every $n \in \mathbb{N}$, there is an explicit code of message length n and rate $\rho = \varepsilon - \gamma$ that can be list-decoded in polynomial time from distance $1 - \varepsilon$, with alphabet size $q = \text{poly}(n)$.

The polynomials in the running time, alphabet size, and list size depend exponentially on γ; specifically they are of the form $n^{O(1/\gamma^2)}$. Nonconstructively, it is possible to have alphabet size $(1/\varepsilon)^{O(\varepsilon/\gamma)}$ and list size $O(1/\gamma)$ (see discussion after Theorem 5.8), and one could hope for running time that is a fixed polynomial in n, with an exponent that is independent of γ. Recent work has achieved these parameters, albeit with a randomized construction of codes; it remains open to have a fully explicit and deterministic construction. However, for a fixed constant-sized alphabet, e.g. $q = 2$, it is still not known how to achieve list-decoding capacity.

Open Problem 5.26. For any desired constants $\rho, \delta > 0$ such that $\rho > 1 - H_2(\delta)$, construct an explicit family of of codes $\text{Enc}_n : \{0,1\}^n \to \{0,1\}^{\hat{n}}$ that have rate at least ρ and are δ-list-decodable in polynomial time.

5.3 List-decoding views of samplers and expanders

In this section, we show how averaging samplers and expander graphs can be understood as variants of list-decodable codes. The general list decoding framework we use to describe these connections will also allow us to capture the other pseudorandom objects we study in this survey.

We begin with the syntactic correspondence between the three objects, so we can view each as a function $\Gamma : [N] \times [D] \to [M]$:

Construction 5.27 (Syntactic Correspondence of Expanders, Samplers, and Codes).

(1) Given a bipartite multigraph G with N left vertices, M right vertices, and left-degree D, we let $\Gamma : [N] \times [D] \to [M]$ be its neighbor function, i.e. $\Gamma(x, y)$ is the y'th neighbor of x in G.

(2) Given a sampler $\text{Samp} : [N] \to [M]^D$, we define $\Gamma : [N] \times [D] \to [M]$ by

$$\Gamma(x, y) = \text{Samp}(x)_y.$$

(3) Given a code $\text{Enc} : [N] \to [q]^D$, we set $M = q \cdot D$, associate the elements of $[q] \times [D]$ with $[M]$, and define $\Gamma : [N] \times [D] \to [M]$ by

$$\Gamma(x, y) = (y, \text{Enc}(x)_y).$$

The correspondence between expanders and samplers is identical to the one from Problem 4.7, which shows that the vertex expansion of G is equivalent to the hitting sampler properties of Samp.

Note that codes correspond to expanders and samplers where the first component of a neighbor/sample equals the edge-label/sample-number. Conversely, any such expander/sampler can be viewed as a code. Many constructions of expanders and samplers have or can be easily modified to have this property. This syntactic constraint turns out to be quite natural in the setting of samplers. It corresponds to the case where we have D functions $f_1, \ldots, f_D : [M] \to [0,1]$, we are interested in estimating the total average $(1/D)\sum_i \mu(f_i)$, but can only evaluate each f_i on a single sample.

We now consider the following generalized notion of list decoding.

Definition 5.28. For a function $\Gamma : [N] \times [D] \to [M]$, a set $T \subset [M]$, and $\varepsilon \in [0,1)$, we define

$$\begin{aligned} \mathrm{LIST}_\Gamma(T,\varepsilon) &= \{x : \Pr_y[\Gamma(x,y) \in T] > \varepsilon\} \text{ and} \\ \mathrm{LIST}_\Gamma(T,1) &= \{x : \forall y\ \Gamma(x,y) \in T\}. \end{aligned}$$

More generally, for a function $f : [M] \to [0,1]$, we define

$$\mathrm{LIST}_\Gamma(f,\varepsilon) = \{x : \mathop{\mathrm{E}}_y[f(\Gamma(x,y))] > \varepsilon\}.$$

We can formulate the list-decoding property of Enc, the averaging sampler property of Samp, and the vertex expansion property of Γ in this language as follows:

Proposition 5.29. Let Enc and Γ be as in Construction 5.27. Then Enc is $(1-1/q-\varepsilon, K)$ list-decodable iff for every $r \in [M]^D$, we have

$$|\mathrm{LIST}_\Gamma(T_r, 1/q+\varepsilon)| \le K,$$

where $T_r = \{(y, r_y) : y \in [D]\}$.

Proof. Observe that, for each $x \in [N]$ and $r \in [M]^D$,

$$\begin{aligned} \Pr_y[\Gamma(x,y) \in T_r] &= \Pr_y[(y, \mathrm{Enc}(x)_y) \in T_r] \\ &= \Pr_y[\mathrm{Enc}(x)_y = r_y] \\ &= \mathrm{agr}(\mathrm{Enc}(x), r). \end{aligned}$$

Thus $|\mathrm{LIST}(T_r, 1/q+\varepsilon)| \le K$ if and only if there are at most K messages x whose encodings have agreement greater than $1/q+\varepsilon$ with r, which is the same as being $(1-1/q-\varepsilon, K)$ list-decodable. □

Proposition 5.30. Let Samp and Γ be as in Construction 5.27. Then

(1) Samp is a (δ,ε) averaging sampler iff for every function $f : [M] \to [0,1]$, we have

$$|\mathrm{LIST}_\Gamma(f, \mu(f)+\varepsilon)| \le \delta N.$$

(2) Samp is a (δ, ε) boolean averaging sampler iff for every set $T \subset [M]$, we have

$$|\mathrm{LIST}_\Gamma(T, \mu(T) + \varepsilon)| \leq \delta N.$$

Proof. Observe that $x \in \mathrm{LIST}_\Gamma(f, \mu(f) + \varepsilon)$ if and only if x is a "bad" set of coin tosses for Samp — namely $(1/D) \cdot \sum_{i=1}^{D} f(z_i) > \mu(f) + \varepsilon$, where $(z_1, \ldots, z_D) = \mathrm{Samp}(x)$. Thus Samp errs with probability at most δ over $x \overset{\mathrm{R}}{\leftarrow} U_{[N]}$ if and only if $|\mathrm{LIST}_\Gamma(f, \mu(f) + \varepsilon)| \leq \delta N$. The boolean averaging sampler case follows by viewing boolean functions f as characteristic functions of sets T. □

Noting that the sets T_r in Proposition 5.29 have density $\mu(T_r) = 1/q$, we see that the averaging-sampler property implies the standard list-decoding property:

Corollary 5.31. If Samp is a (δ, ε) boolean averaging sampler of the form $\mathrm{Samp}(x)_y = (y, \mathrm{Enc}(x)_y)$, then Enc is $(1 - 1/q - \varepsilon, \delta N)$ list-decodable.

Note, however, that the typical settings of parameters of samplers and list-decodable codes are very different. With codes, we want the alphabet size q to be as small as possible (e.g. $q = 2$) and the blocklength D to be linear or polynomial in the message length $n = \log N$, so $M = qD$ is also linear or polynomial in $n = \log N$. In contrast, we usually are interested in samplers for functions on exponentially large domains (e.g. $M = 2^{\Omega(n)}$).

In Chapter 6, we will see a converse to Corollary 5.31 when the alphabet size is small: if Enc is $(1 - 1/q - \varepsilon, \delta N)$, list-decodable, then Samp is an $(\delta/\varepsilon, q \cdot \varepsilon)$ averaging sampler.

For expanders, it will be convenient to state the list-decoding property in terms of the following variant of vertex expansion, where we only require that sets of size exactly K expand:

Definition 5.32. For $K \in \mathbb{N}$, a bipartite multigraph G is an $(= K, A)$ *vertex expander* if all sets S consisting of K left-vertices, the neighborhood $N(S)$ is of size at least $A \cdot K$.

Thus, G is a (K, A) vertex expander in the sense of Definition 4.3 iff G is an $(= K', A)$ vertex expander for all positive integers $K' \leq K$.

Proposition 5.33. For $K \in \mathbb{N}$, $\Gamma : [N] \times [D] \to [M]$ is an $(= K, A)$ vertex expander iff for every set $T \subset [D] \times [M]$ such that $|T| < KA$, we have:

$$|\mathrm{LIST}_\Gamma(T, 1)| < K.$$

Proof.

$$\begin{aligned} \Gamma \text{ not an } (= K, A) \text{ expander} &\Leftrightarrow \exists S \subset [N] \text{ s.t. } |S| = K \text{ and } |N(S)| < KA \\ &\Leftrightarrow \exists S \subset [N] \text{ s.t. } |S| \geq K \text{ and } |N(S)| < KA \\ &\Leftrightarrow \exists T \subset [M] \text{ s.t. } |\mathrm{LIST}(T, 1)| \geq K \text{ and } |T| < KA, \end{aligned}$$

where the last equivalence follows because if $T = N(S)$, then $S \subset \mathrm{LIST}(T, 1)$, and conversely if $S = \mathrm{LIST}(T, 1)$ then $N(S) \subset T$. □

On one hand, this list-decoding property seems easier to establish than the ones for codes and samplers because we look at $\mathrm{LIST}(T,1)$ instead of $\mathrm{LIST}(T,\mu(T)+\varepsilon)$. On the other hand, to get expansion (i.e. $A>1$), we require a very tight relationship between $|T|$ and $|\mathrm{LIST}(T,1)|$. In the setting of codes or samplers, we would not care much about a factor of 2 loss in $|\mathrm{LIST}(T)|$, as this just corresponds to a factor of 2 in list size or error probability. But here it corresponds to a factor of 2 loss in expansion, which can be quite significant. In particular, we cannot afford it if we are trying to get $A=(1-\varepsilon)\cdot D$, as we will be in the next section.

5.4 Expanders from Parvaresh–Vardy Codes

Despite the substantial differences in the standard settings of parameters between codes, samplers, and expanders, it can be very useful to translates ideas and techniques from one object to the other using the connections described in the previous section. In particular, in this section we will see how to build graphs with extremely good vertex expansion ($A=(1-\varepsilon)D$) from Parvaresh–Vardy codes.

Consider the bipartite multigraph obtained from the Parvaresh–Vardy codes (Construction 5.21) via the correspondence of Construction 5.27. That is, we define a neighbor function $\Gamma:\mathbb{F}_q^n\times\mathbb{F}_q\to\mathbb{F}_q\times\mathbb{F}_q^m$ by

$$\Gamma(f,y)=[y,f_0(y),f_1(y),\ldots,f_{m-1}(y)],\tag{5.9}$$

where $f(Y)$ is a polynomial of degree at most $n-1$ over $\mathbb{F}_q$, and we define $f_i(Y)=f(Y)^{h^i}$ mod $E(Y)$, where E is a fixed irreducible polynomial of degree n over $\mathbb{F}_q$. (Note that we are using $n-1$ instead of d to denote degree of f.)

Theorem 5.34. Let $\Gamma:\mathbb{F}_q^n\times\mathbb{F}_q\to\mathbb{F}_q^{m+1}$ be the neighbor function of the bipartite multigraph corresponding to the q-ary Parvaresh–Vardy code of degree $d=n-1$, power h, and redundancy m via Construction 5.27. Then Γ is a (K_{max},A) expander for $K_{max}=h^m$ and $A=q-nhm$.

Proof. Let K be any integer less than or equal to $K_{max}=h^m$, and let $A=q-nmh$. By Proposition 5.33, it suffices to show that for every set $T\subset\mathbb{F}_q^{m+1}$ of size at most $AK-1$, we have $|\mathrm{LIST}(T)|\le K-1$.

We begin by doing the proof for $K=K_{max}=h^m$, and later describe the modifications to handle smaller values of K. The proof goes along the same lines as the list-decoding algorithm for the Parvaresh–Vardy codes from Section 5.2.3.

Step 1: Find a low-degree Q vanishing on T. We find a nonzero polynomial $Q(Y,Z_0,\ldots,Z_{m-1})$ of degree at most $d_Y=A-1$ in its first variable Y and at most $h-1$ in each of the remaining variables such that $Q(z)=0$ for all $z\in T$. (Compare this to $Q(y,r(y))=0$ for all $y\in\mathbb{F}_q$ in the list-decoding algorithm, which corresponds to taking $T=T_r$.)

This is possible because

$$A\cdot h^m=AK>|T|.$$

Moreover, we may assume that Q is not divisible by $E(Y)$. If it is, we can divide out all the factors of $E(Y)$, which will not affect the conditions $Q(z)=0$ since E has no roots (being irreducible).

Step 2: Argue that each $f(Y) \in \mathrm{LIST}(T,1)$ is a "root" of a related univariate polynomial Q^*. First, we argue as in the list-decoding algorithm that if $f \in \mathrm{LIST}(T,1)$, we have

$$Q(Y, f_0(Y), \ldots, f_{m-1}(Y)) = 0.$$

This is ensured because

$$q > A - 1 + nmh.$$

(In the list-decoding algorithm, the left-hand side of this inequality was εq, since we were bounding $|\mathrm{LIST}(T_r, \varepsilon)|$.)

Once we have this, we can reduce both sides modulo $E(Y)$ and deduce

$$\begin{aligned} 0 &= Q(Y, f_0(Y), f_1(Y), \ldots, f_{m-1}(Y)) \bmod E(Y) \\ &= Q(Y, f(Y), f(Y)^{h^2}, \ldots, f(Y)^{h^{m-1}}) \bmod E(Y) \end{aligned}$$

Thus, if we define the univariate polynomial

$$Q^*(Z) = Q(Y, Z, Z^h, \ldots, Z^{h^{m-1}}) \bmod E(Y),$$

then $f(Y)$ is a root of Q^* over the field $\mathbb{F}_q[Y]/E(Y)$.

Observe that Q^* is nonzero because Q is not divisible by $E(Y)$ and has degree at most $h-1$ in each Z_i. Thus,

$$|\mathrm{LIST}(T,1)| \leq \deg(Q^*) \leq h - 1 + (h-1)\cdot h + (h-1)\cdot h^2 + \cdots + (h-1)\cdot h^{m-1} = K - 1.$$

(Compare this to the list-decoding algorithm, where our primary goal was to efficiently enumerate the elements of $\mathrm{LIST}(T,\varepsilon)$, as opposed to bound its size.)

Handling smaller values of K. We further restrict $Q(Y, Z_1, \ldots, Z_m)$ to only have nonzero coefficients on monomials of the form $Y^i \mathrm{Mon}_j(Z_1, \ldots, Z_m)$ for $0 \leq i \leq A-1$ and $0 \leq j \leq K-1 \leq h^m - 1$, where $\mathrm{Mon}_j(Z_1, \ldots, Z_m) = Z_1^{j_0} \cdots Z_m^{j_{m-1}}$ and $j = j_0 + j_1 h + \cdots + j_{m-1} h^{m-1}$ is the base-h representation of j. Note that this gives us $AK > |T|$ monomials, so Step 1 is possible. Moreover $M_j(Z, Z^h, Z^{h^2}, \ldots, Z^{h^{m-1}}) = Z^j$, so the degree of Q^* is at most $K-1$, and we get the desired list-size bound in Step 3. □

Setting parameters, we have:

Theorem 5.35. For every constant $\alpha > 0$, every $N \in \mathbb{N}$, $K \leq N$, and $\varepsilon > 0$, there is an explicit $(K, (1-\varepsilon)D)$ expander with N left-vertices, M right-vertices, left-degree $D = O((\log N)(\log K)/\varepsilon)^{1+1/\alpha}$ and $M \leq D^2 \cdot K^{1+\alpha}$. Moreover, D is a power of 2.

Proof. Let $n = \log N$ and $k = \log K_{\max}$. Let $h = \lceil (2nk/\varepsilon)^{1/\alpha} \rceil$ and let q be the power of 2 in the interval $(h^{1+\alpha}/2, h^{1+\alpha}]$.

Set $m = \lceil (\log K_{\max})/(\log h) \rceil$, so that $h^{m-1} \leq K_{\max} \leq h^m$. Then, by Theorem 5.34, the graph $\Gamma : \mathbb{F}_q^n \times \mathbb{F}_q \to \mathbb{F}_q^{m+1}$ defined in (5.9) is an (h^m, A) expander for $A = q - nhm$. Since $K_{\max} \leq h^m$, it is also a (K_{max}, A) expander.

Note that the number of left-vertices in Γ is $q^n \geq N$, and the number of right-vertices is

$$M = q^{m+1} \leq q^2 \cdot h^{(1+\alpha)\cdot(m-1)} \leq q^2 \cdot K_{\max}^{1+\alpha}.$$

The degree is

$$D = q \le h^{1+\alpha} = O(nk/\varepsilon)^{1+1/\alpha} = O((\log N)(\log K_{\max})/\varepsilon)^{1+1/\alpha}.$$

To see that the expansion factor $A = q - nhm \ge q - nhk$ is at least $(1-\varepsilon)D = (1-\varepsilon)q$, note that

$$nhk \le (\varepsilon/2) \cdot h^{1+\alpha} \le \varepsilon q,$$

where the first inequality holds because $h^\alpha \ge 2nk/\varepsilon$.

Finally, the construction is explicit because a description of $\mathbb{F}_q$ for q a power of 2 (i.e. an irreducible polynomial of degree $\log q$ over $\mathbb{F}_2$) as well as an irreducible polynomial $E(Y)$ of degree n over $\mathbb{F}_q$ can be found in time $\mathrm{poly}(n, \log q) = \mathrm{poly}(\log N, \log D)$. □

These expanders are of polylogarithmic rather than constant degree. But the expansion is almost as large as possible given the degree ($A = (1-\varepsilon) \cdot D$), and the size of the right-hand side is almost as small as possible (in a (K, A) expander, we must have $M \ge KA = (1-\varepsilon)KD$). In particular, these expanders can be used in the data structure application of Problem 4.10 — storing a K-sized subset of $[N]$ using $K^{1.01} \cdot \mathrm{polylog}(N)$ bits in such a way that membership can be probabilistically tested by reading 1 bit of the data structure. (An efficient solution to that application actually requires more than the graph being explicit in the usual sense, but also that there are efficient algorithms for finding all left-vertices having at least some δ fraction neighbors in a given set $T \subset [M]$ of right vertices, but the expanders above can be shown to have that property by a variant of the list-decoding algorithm above.)

A deficiency of the expander of Theorem 5.35 is that the size of the right-hand side is polynomial in K and D (for constant α), whereas the optimal bound is $M = O(KD/\varepsilon)$. Achieving the latter, while keeping the left-degree polylogarithmic, is an open problem:

Open Problem 5.36. Construct $(= K, A)$ bipartite expanders with N left-vertices, degree $D = \mathrm{poly}(\log N)$, expansion $A = .99D$, and $M = O(KD)$ right-hand vertices.

We remark that a construction where D is *quasipolynomial* in $\log N$ is known.

5.5 Exercises

~~**Problem 5.1 (Limits of List Decoding).** Show that if there exists a q-ary code $\mathcal{C} \subset \Sigma^{\hat{n}}$ of rate~~ ρ that is (δ, L) list-decodable, then $\rho \le 1 - H_q(\delta, \hat{n}) + (\log_q L)/\hat{n}$

Problem 5.2 (Concatenated Codes). For codes $\mathrm{Enc}_1 : \{1, \ldots, N\} \to \Sigma_1^{n_1}$ and $\mathrm{Enc}_2 : \Sigma_1 \to \Sigma_2^{n_2}$, their *concatenation* $\mathrm{Enc} : \{1, \ldots, N\} \to \Sigma_2^{n_1 n_2}$ is defined by

$$\mathrm{Enc}(m) = \mathrm{Enc}_2(\mathrm{Enc}_1(m)_1)\mathrm{Enc}_2(\mathrm{Enc}_1(m)_2) \cdots \mathrm{Enc}_2(\mathrm{Enc}_1(m)_{n_1}).$$

This is typically used as a tool for reducing alphabet size, e.g. with $\Sigma_2 = \{0, 1\}$.

(1) Prove that if Enc_1 has minimum distance δ_1 and Enc_2 has minimum distance δ_2, then Enc has minimum distance at least $\delta_1 \delta_2$.

(2) Prove that if Enc_1 is $(1-\varepsilon_1, \ell_1)$ list-decodable and Enc_2 is (δ_2, ℓ_2) list-decodable, then Enc is $((1-\varepsilon_1\ell_2)\cdot\delta_2, \ell_1\ell_2)$ list-decodable.

(3) By concatenating a Reed–Solomon code and a Hadamard code, show that for every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is a (fully) explicit code $\text{Enc} : \{0,1\}^n \to \{0,1\}^{\hat{n}}$ with blocklength $\hat{n} = O(n^2/\varepsilon^2)$ with minimum distance at least $1/2 - \varepsilon$. Furthermore, show that with blocklength $\hat{n} = \text{poly}(n, 1/\varepsilon)$, we can obtain a code that is $(1/2-\varepsilon, \text{poly}(1/\varepsilon))$ list-decodable in *polynomial time.* (Hint: the inner code can be decoded by brute force.)

Problem 5.3 (List Decoding implies Unique Decoding for Random Errors). (1) Suppose that $\mathcal{C} \subset \{0,1\}^{\hat{n}}$ is a code with minimum distance at least $1/4$ and rate at most $\alpha\varepsilon^2$ for a constant $\alpha > 0$, and we transmit a codeword $c \in \mathcal{C}$ over a channel in which each bit is flipped with probability $1/2 - 2\varepsilon$. Show that if α is a sufficiently small constant (independent of $\hat{n}$ and ε), then all but exponentially small probability over the errors, c will be the unique codeword at distance at most $1/2 - \varepsilon$ from the received word r.

(2) Using Problem 5.2, deduce that for every $\varepsilon > 0$ and $n \in \mathbb{N}$, there is an explicit code of blocklength $\hat{n} = \text{poly}(n, 1/\varepsilon)$ that can be uniquely decoded from $(1/2 - 2\varepsilon)$ random errors as above in polynomial time.

Problem 5.4 (Linear Codes). For a prime power q, a q-ary code $\mathcal{C} \subset \mathbb{F}_q^{\hat{n}}$ is called *linear* if $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^{\hat{n}}$. That is, for every $u, v \in \mathcal{C}$ and $\alpha \in \mathbb{F}_q$, we have $u + v \in \mathcal{C}$ and $\alpha u \in \mathcal{C}$.

(1) Verify that the Hadamard, Reed–Solomon, and Reed–Muller codes are all linear.

(2) Show that if $\mathcal{C}$ is linear, then the minimum distance of $\mathcal{C}$ equals the *minimum weight* of $\mathcal{C}$, which is defined to be $\min_{c \in \mathcal{C}\setminus\{0\}} |\{i : c_i \neq 0\}|/\hat{n}$.

(3) Show that if $\mathcal{C}$ is a subspace of dimension n, then its rate is $n/\hat{n}$ and it has an encoding function $\text{Enc} : \mathbb{F}_q^n \to \mathbb{F}_q^{\hat{n}}$ that is a linear map. Moreover, the encoding function can be made to have the property that there is a set $S \subset [\hat{n}]$ of n coordinates such that $\text{Enc}(m)|_S = m$ for every $m \in \mathbb{F}_q^{\hat{n}}$. (Here $c|_S$ is the projection of c onto the coordinates in S.) Such encodings are called *systematic*, and will be useful when we study locally decodable codes in Chapter 7.

(4) Find explicit systematic encodings for the Hadamard and Reed–Solomon codes.

(5) Show that the nonconstructive bound of Theorem 5.8, Part 1, can be achieved by a linear code. That is, for all prime powers q, integers $n, \hat{n} \in \mathbb{N}$, and $\delta \in (0, 1-1/q)$, there exists a *linear* q-ary code of block length $\hat{n}$, minimum distance at least δ, and rate at least $\rho = n/\hat{n}$ provided $\rho \leq 1 - H_q(\delta, \hat{n})$. (Hint: chose a random linear map $\text{Enc} : \mathbb{F}_q^n \to \mathbb{F}_q^{\hat{n}}$ and use Part 2.)

Problem 5.5 (LDPC Codes). Given a bipartite multigraph G with N left-vertices and M right-vertices, we can obtain a linear code $\mathcal{C} \subset \{0,1\}^N$ (where we view $\{0,1\}$ as the field of two elements) by:

$$\mathcal{C} = \{c \in \{0,1\}^N : \forall j \in [M] \ \ \oplus_{i \in \Gamma(j)} c_i = 0\},$$

where $\Gamma(j)$ denotes the set of neighbors of vertex j. When G has small left-degree D (e.g. $D = O(1)$), then $\mathcal{C}$ is called a *low-density parity check (LDPC) code.*

(1) Show that $\mathcal{C}$ has rate at least $1 - M/N$.
(2) Show that if G is a (K, A) expander for $A > D/2$, then $\mathcal{C}$ has minimum distance at least $\delta = K/N$.
(3) Show that if G is a $(K, (1-\varepsilon)D)$ expander for a sufficiently small constant ε, then $\mathcal{C}$ has a polynomial-time δ-decoder for $\delta = (1 - 3\varepsilon) \cdot K/N$. Assume that G is given as input to the decoder. (Hint: given a received word $r \in \{0,1\}^n$, flip all coordinates of r for which at least $2/3$ of the neighboring parity checks are not satisfied, and argue that the number of errors decreases by a constant factor. It may be useful to use the results of Problem 4.10.)

By a probabilistic argument like Theorem 4.4, graphs G as above exist with $D = O(1)$, $K = \Omega(N)$, $M = (1 - \Omega(1))N$, arbitrarily small constant $\varepsilon > 0$, and $N \to \infty$, and in fact explicit constructions are known. This yields explicit LDPC codes with constant rate and constant distance ("asymptotically good LDPC codes").

Problem 5.6 (Improved list decoding of Reed–Solomon Codes). (1) Show that there is a polynomial-time algorithm for list decoding the Reed-Solomon codes of degree d over $\mathbb{F}_q$ up to distance $1 - \sqrt{2d/q}$, improving the $1 - 2\sqrt{d/q}$ bound from Theorem 5.19. (Hint: do not use fixed upper bounds on the individual degrees of the interpolating polynomial $Q(Y, Z)$, but rather allow as many monomials as possible.)

(2) (*) Improve the list-decoding radius further to $1 - \sqrt{d/q}$ by using the following "method of multiplicities". First, require the interpolating polynomial $Q(Y, Z)$ to have a zero of multiplicity s at each point $(y, r(y))$ — that is, the polynomial $Q(Y + y, Z + r(y))$ should have no monomials of degree smaller than s. Second, use the fact that a univariate polynomial $R(Y)$ of degree t can have at most t roots, counting multiplicities.

Problem 5.7 (Twenty Questions). In the game of 20 questions, an oracle has an arbitrary secret $s \in \{0,1\}^n$ and the aim is to determine the secret by asking the oracle as few yes/no questions about s as possible. It is easy to see that n questions are necessary and sufficient. Here we consider a variant where the oracle has two secrets $s_1, s_2 \in \{0,1\}^n$, and can adversarially decide to answer each question according to either s_1 or s_2. That is, for a question $f : \{0,1\}^n \to \{0,1\}$, the oracle may answer with either $f(s_1)$ or $f(s_2)$. Here it turns out to be impossible to pin down either of the secrets with certainty, no matter how many questions we ask, but we can hope to compute a small list L of secrets such that $|L \cap \{s_1, s_2\}| \neq 0$. (In fact, $|L|$ can be made as small as 2.) This variant of twenty questions apparently was motivated by problems in Internet traffic routing.

(1) Let Enc : $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ be a code such that every two codewords in Enc *agree* in at least a $1/2 - \varepsilon$ fraction of positions and that Enc has a polynomial-time $(1/4 + \varepsilon, \ell)$ list-decoding algorithm. Show how to solve the above problem in polynomial time by asking the $\hat{n}$ questions $\{f_i\}$ defined by $f_i(x) = \text{Enc}(x)_i$.

(2) Taking Enc to be the code constructed in Problem 1, deduce that $\hat{n} = \text{poly}(n)$ questions suffices.

5.6 Chapter Notes and References

Standard texts on coding theory include MacWilliams–Sloane [MS] and the Handbook of Coding Theory [PHB]. The lecture notes of Sudan [Sud2] present coding theory with an algorithmic perspective, and Guruswami [Gur2] gives a thorough treatment of list decoding.

The field of coding theory began with Shannon's seminal 1948 paper [Sha3], which proposed the study of stochastic error models and proved that a random error-correcting code achieves an optimal rate-distance tradeoff with high probability. The study of decoding from worst-case errors began a couple of years later with the work of Hamming [Ham]. The notion of list decoding was proposed independently in the late 1950's, in the work of Elias [Eli1] and Wozencraft [Woz].

The nonconstructive bound for the tradeoff between rate and minimum distance of Theorem 5.8 (Part 1) is due to Gilbert [Gil1], and its extension to linear codes in Problem 5.4 (Part 5) is due to Varshamov [Var]; together they are known as the Gilbert–Varshamov Bound. For more on the algebraic geometric codes and how they can be used to beat the Gilbert–Varshamov Bound, see the text by Stichtenoth [Sti1]. The nonconstructive bound for list decoding of Theorem 5.8 (Part 2) is due to Elias [Eli3], and it has been extended to linear codes in [ZP, GHSZ, GHK]. The Johnson Bound (Proposition 5.10) was proven for binary codes by Johnson [Joh2, Joh1]. An optimized form of the bound can be found in [GS3].

The binary ($q = 2$) case of Reed–Muller codes was introduced independently by Reed [Ree] and Muller [Mul] in the mid-50's. Reed–Solomon codes were introduced in 1960 by Reed and Solomon [RS]. Polynomial time unique-decoding algorithms for Reed–Solomon Codes include those of Peterson [Pet] and Berlekamp [Ber2], . The first nontrivial list decoding algorithm was given by Goldreich and Levin [GL]; while their algorithm is stated in the language of "hardcore bits for one-way functions," it can be viewed as an efficient "local" list-decoding algorithm for the Hadamard Code. (Local decoding algorithms are discussed in Chapter 7.) The list-decoding algorithm for Reed–Solomon Codes of Theorem 5.19 is from the seminal work of Sudan [Sud1], which sparked a resurgence in the study of list decoding. The improved decoding algorithm of Problem 5.6, Part 2 is due to Guruswami and Sudan [GS1]. Parvaresh–Vardy codes and their decoding algorithm are from [PV]. Folded Reed–Solomon Codes and their capacity-achieving list-decoding algorithm are due to Guruswami and Rudra [GR]. (In all cases, our presentation uses a simplified setting of parameters compared to the original algorithms.) The list size, decoding time, and alphabet size of Folded Reed–Solomon Codes have recently been improved in [Gur3, DL, GX], with the best parameters to date being achieved in the randomized construction of [GX], while [DL] give a fully deterministic construction.

The list-decoding views of expanders and samplers emerged out of work on randomness extractors (the subject of Chapter 6). Specifically, a close connection between extractors and expanders was understood already at the time that extractors were introduced by Nisan and Zuckerman [NZ]. An equivalence between extractors and averaging samplers was established by Zuckerman [Zuc2] (building on previous connections between other types of samplers and expanders [Sip2, CW1]). A connection between list-decodable codes and extractors emerged in the work of Trevisan [Tre1], and the list-decoding view of extractors was crystallized by Ta-Shma and Zuckerman [TZ]. The list-decoding formulation of vertex expansion and the construction of expanders from Parvaresh–Vardy codes is due to Guruswami, Umans, and Vadhan [GUV].

Code concatenation (Problem 5.2) was introduced by Forney [For], who used it to construct explicit, efficiently decodable binary codes that achieve capacity for random errors. Efficient list-decoding algorithms for the concatenation of a Reed–Solomon code with a Hadamard code were given in [GS2]. Low-density parity check (LDPC) codes (Problem 5.5) were introduced by Gallager [Gal]. The use of expansion to analyze such codes and their decoding algorithm comes from Sipser and Spielman [SS]. Explicit expanders suitable for Problem 5.5 were given by [CRVW]. The surveys of Guruswami [Gur1, Gur2] describe the state of the art in expander-based constructions of codes and in LDPC codes, respectively.

The question of Problem 5.7 (Twenty Questions) was posed in [CGL], motivated by Internet traffic routing applications. The solution using list decoding is from [AGKS].