Introduction to Coding Theory **CMU: Spring 2010**

Notes 1: Introduction, linear codes

January 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

The theory of error-correcting codes and more broadly, information theory, originated in Claude Shannon's monumental work A mathematical theory of communication, published over 60 years ago in 1948. Shannon's work gave a precise measure of the information content in the output of a random source in terms of its *entropy*. The *noiseless coding theorem* or the source coding theorem informally states that n i.i.d. random variables each with entropy $H(X)$ can be compressed into $n(H(X)+\epsilon)$ bits with negligible probability of information loss, and conversely compression into $n(H(X)-\epsilon)$ bits would entail almost certain information loss.

More directly relevant to this course is Shannon's *noisy coding theorem* which considered communication of a message (say consisting of k bits that are output by a source coder) on a noisy communication channel whose behavior is given by a stochastic channel law. The noisy coding theorem states that every such channel has a precisely defined real number called *capacity* that quantifies the maximum rate at which reliable communication is possible on that channel. More precisely, given a noisy channel with capacity C, if information is transmitted at rate R (which means $k = nR$ message bits are communicated in n uses of the channel), then if $R < C$ then *exist* coding schemes (comprising an encoder/decoder pair) that guarantee negligible probability of miscommunication, whereas if $R > C$, then regardless of the coding scheme, the probability of error at the receiver is bounded below by some constant (which increased as R increases). (Later, a strong converse to the Shannon coding theorem was proved, which shows that when $R > C$, the probability of miscommunication goes exponentially (in k) to 1.) Shannon's theorem was one of the early uses of the probabilistic method; it asserted the existence of good coding schemes at all rates below capacity, but did not give any efficient method to construct a good code or for that matter to verify that a certain code was good.

We will return to Shannon's probabilistic viewpoint and in particular his noisy coding theorem in a couple of lectures, but we will begin by introducing error-correcting codes from a more combinatorial/geometric viewpoint, focusing on aspects such as the minimum distance of the code. This viewpoint was pioneered by Richard Hamming in his celebrated 1950 paper Error detecting and error correcting codes. The Hamming approach is more suitable for tackling worst-case/adversarial errors, whereas the Shannon theory handles stochastic/probabilistic errors. This corresponds to a rough dichotomy in coding theory results – while the two approaches have somewhat different goals and face somewhat different limits and challenges, they share many common constructions, tools, and techniques. Further, by considering meaningful relaxations of the adversarial noise model or the requirement on the encoder/decoder, it is possible to bridge the gap between the Shannon and Hamming approaches. (We will see some results in this vein during the course.)

The course will be roughly divided into the following interrelated parts. We will begin by results on the existence and limitations of codes, both in the Hamming and Shannon approaches. This will highlight some criteria to judge when a code is good, and we will follow up with several explicit constructions of "good" codes (we will encounter basic finite field algebra during these

constructions). While this part will mostly have a combinatorial flavor, we will keep track of important algorithmic challenges that arise. This will set the stage for the algorithmic component of the course, which will deal with efficient (polynomial time) algorithms to decode some important classes of codes. This in turn will enable us to approach the absolute limits of error-correction "constructively," via explicit coding schemes and efficient algorithms (for both worst-case and probabilistic error models).

Codes, and ideas behind some of the good constructions, have also found many exciting "extraneous" applications such as in complexity theory, cryptography, pseudorandomness and explicit combinatorial constructions. (For example, in the spring 2009 offering of 15-855 (the graduate complexity theory course), we covered in detail the Sudan-Trevisan-Vadhan proof of the Impagliazzo-Wigderson theorem that P = BPP under a exponential circuit lower bound for E, based on a highly efficient decoding algorithm for Reed-Muller codes.) Depending on time, we may mention/discuss some of these applications of coding theory towards the end of the course, though given that there is plenty to discuss even restricting ourselves to primarily coding-theoretic motivations, this could be unlikely.

We now look at some simple codes and give the basic definitions concerning codes. But before that, we will digress with some recreational mathematics and pose a famous "Hat" puzzle, which happens to have close connections to the codes we will soon introduce (that's your hint, if you haven't seen the puzzle before!)

Guessing hat colors

The following puzzle made the New York Times in 2001.

15 players enter a room and a red or blue hat is placed on each person's head. The color of each hat is determined by a coin toss, with the outcome of one coin toss having no effect on the others. Each person can see the other players' hats but not his own.

No communication of any sort is allowed, except for an initial strategy session before the game begins. Once they have had a chance to look at the other hats, the players must simultaneously guess the color of their own hats or pass. The group wins the game if at least one player guesses correctly and no players guess incorrectly.

One obvious strategy for the players, for instance, would be for one player to always guess "red" while the other players pass. This would give the group a 50 percent chance of winning the prize. Can the group achieve a higher probability of winning (probability being taken over the initial random assignment of hat colors)? If so, how high a probability can they achieve?

(The same game can be played with any number of players. The general problem is to find a strategy for the group that maximizes its chances of winning the prize.)

1 A simple code

Suppose we need to store 64 bit words in such a way that they can be correctly recovered even if a single bit per word gets flipped. One way is to store each information bit by duplicating it

three times. We can thus store 21 bits of information in the word. This would permit only about a fraction $\frac{1}{3}$ of information to be stored per bit of the word. However, it would allow us to correct any single bit flip since the majority value of the three copies of the bit gives the correct value of the bit, even if one of the copies is flipped.

Hamming in 1950 introduced a code, now called the "Hamming code," which could also correct 1-bit errors using fewer redundant (or extra) bits. The code is defined in such a way that a chunk of 4 information bits x_1, x_2, x_3, x_4 gets mapped (or "encoded") to a "codeword" of 7 bits as

$$x_1,\ x_2,\ x_3,\ x_4,\ x_2 \oplus x_3 \oplus x_4,\ x_1 \oplus x_3 \oplus x_4\, x_1 \oplus x_2 \oplus x_4,$$

This transformation can equivalently be represented as a mapping from x to Gx (the operations are done modulo 2) where x is the column vector $[x_1\, x_2\, x_3\, x_4]^T$ and G is the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

It is easy to verify that two distinct 4-bit vectors x and y get mapped to codewords Gx and Gy which differ in at least 3 bits. (Define $w = x - y$ so that $w \neq 0$. Now check that for each non-zero w, Gw has at least 3 bits which are 1.) It follows that the above code can correct all single bit flips, since for any 7-bit vector there is always at most one codeword which can be obtained by a single bit flip. (As we will see soon, these codes also have the remarkable property that for $y \in \{0,1\}^7$ which is not a codeword, there is always a codeword which can be obtained by a single bit flip.)

2 Some basic definitions

Let us get a few simple definitions out of the way.

Definition 1 (Hamming distance) *The Hamming distance between two strings x and y of the same length over a finite alphabet Σ, denoted $\Delta(x, y)$, is defined as the number of positions at which the two strings differ, i.e., $\Delta(x, y) = |\{i | x_i \neq y_i\}|$. The fractional Hamming distance or relative distance between $x, y \in \Sigma^n$ is given by $\delta(x, y) = \frac{\Delta(x,y)}{n}$.*

It is trivial to check that the Hamming distance defines a metric on Σ^n.

Definition 2 (Hamming weight) *The Hamming weight of a string x over alphabet Σ is defined as the number of non-zero symbols in the string. More formally, the Hamming weight of a string $\mathrm{wt}(x) = |\{i | x_i \neq 0\}|$. Note that $\mathrm{wt}(x - y) = \Delta(x, y)$.*

Given a string $x \in \Sigma^n$, the *Hamming ball* or radius r around x is the set $\{y \in \Sigma^n \mid \Delta(x, y) \leq r\}$.

Definition 3 (Code) *An error correcting code or block code C of length n over a finite alphabet Σ is a subset of Σ^n. The elements of C are called the codewords in C, and the collection of all codewords is sometimes called a codebook.*

The alphabet of C is Σ, and if $|\Sigma| = q$, we say that C is a q-ary code. When $q = 2$, we say that C is a binary code. The length n of the codewords of C is called the block length of C.

Associated with a code is also an encoding map E which maps the message set $\mathcal{M}$, identified in some canonical way with $\{1, 2, \ldots, |C|\}$ say, to codewords belonging to Σ^n. The code is then the image of the encoding map.

We now define two crucial parameters concerning a code: its *rate* which measures the amount of redundancy introduced by the code, and its *minimum distance* which measures the error-resilience of a code quantified in terms of how many errors need to be introduced to confuse one codeword for another.

Definition 4 (Rate) *The rate of a code $C \subseteq \Sigma^n$, denoted $R(C)$, is defined by*

$$R(C) = \frac{\log |C|}{n \log |\Sigma|} \ .$$

Thus, $R(C)$ is the amount of non-redundant information per bit in codewords of C.

The dimension of C is defined to $\frac{\log |C|}{\log |\Sigma|}$; this terminology will make sense once we define linear codes shortly. Note that a q-ary code of dimension ℓ has q^ℓ codewords.

Definition 5 (Distance) *The minimum distance, or simply distance, of a code C, denoted $\Delta(C)$, is defined to be the minimum Hamming distance between two distinct codewords of C. That is,*

$$\Delta(C) = \min_{\substack{c_1, c_2 \in C \\ c_1 \neq c_2}} \Delta(c_1, c_2) \ .$$

In particular, for every pair of distinct codewords in C the Hamming distance between them is at least $\Delta(C)$.

The relative distance of C, denoted $\delta(C)$, is the normalized quantity $\frac{\Delta(C)}{n}$, where n is the block length of C. Thus, any two codewords of C differ in at least a fraction $\delta(C)$ of positions.

Example 1 *The parity check code, which maps k bits to $k + 1$ bits by appending the parity of the message bits, is an example of distance 2 code. Its rate is $k/(k + 1)$.*

Example 2 *The Hamming code discussed earlier is an example of distance 3 code, and has rate $4/7$.*

The following simple fact highlights the importance of distance of a code for correcting (worst-case) errors. The proof follows readily from the definition of minimum distance.

Lemma 6 *For a code, the following statements are equivalent:*

1. *C has minimum distance $2t+1$.*
2. *C can be used to correct all t symbol errors.*
3. *C can be used to detect all $2t$ symbol errors.*
4. *C can be used to correct all $2t$ symbol erasures. (In the erasure model, some symbols are erased and the rest are intact, and we know the locations of the erasures. The goal is to fill in the values of the erased positions, using the values of the unerased positions and the redundancy of the code.)*

3 Linear codes

A general code might have no structure and not admit any representation other than listing the entire codebook. We now focus on an important subclass of codes with additional structure called linear codes. Many of the important and widely used codes are linear.

Linear codes are defined over alphabets Σ which are finite fields. Throughout, we will denote by $\mathbb{F}_q$ the finite field with q elements, where q is a prime power. (Later on in the course, it is valuable to have a good grasp of the basic properties of finite fields and field extensions. For now, we can safely think of q as a prime, in which case $\mathbb{F}_q$ is just $\{0, 1, \ldots, q-1\}$ with addition and multiplication defined modulo q.)

Definition 7 (Linear code) *If Σ is a field and $C \subset \Sigma^n$ is a subspace of Σ^n then C is said to be a linear code.*

As C is a subspace, there exists a basis $c_1, c_2, \ldots, c_k$ where k is the dimension of the subspace. Any codeword can be expressed as the linear combination of these basis vectors. We can write these vectors in matrix form as the columns of a $n \times k$ matrix. Such a matrix is called a generator matrix.

Definition 8 (Generator matrix and encoding) *Let $C \subseteq \mathbb{F}_q^n$ be a linear code of dimension k. A matrix $G \in \mathbb{F}_q^{n \times k}$ is said to be a generator matrix for C if its k columns span C.*
The generator matrix G provides a way to encode a message $x \in \mathbb{F}_q^k$ (thought of as a column vector) as the codeword $Gx \in C \subseteq \mathbb{F}_q^n$. Thus a linear code has an encoding map $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ which is a linear transformation $x \to Gx$.

Comment: Many coding texts define the "transpose" version, where the rows of the $k \times n$ generator matrix span the code. We prefer the above definition since it is customary to treat vectors as column vectors (even in these coding texts) and it is therefore nice to multiply by vectors on the right and avoid taking transposes of vectors.

Note that a linear code admits many different generator matrices, corresponding to the different choices of basis for the code as a vector space.

Notation: A q-ary linear code of block length n and dimension k will be referred to as an $[n, k]_q$ code. Further, it the code has minimum distance d, it will be referred to as an $[n, k, d]_q$ code. When the alphabet size q is clear from the context, or not very relevant to the discussion, we omit the subscript.

Example 3 *Some simple examples of binary linear codes:*

- *The binary parity check code: This is an* $[n, n-1, 2]_2$ *code consisting of all vectors in* $\mathbb{F}_2^n$ *of even Hamming weight.*
- *The binary repetition code: This is an* $[n, 1, n]_2$ *code consisting of the two vectors* 0^n *and* 1^n*.*
- *The Hamming code discussed above is a* $[7, 4, 3]_2$ *linear code.*

Exercise 1 *Show that for a linear code* C*, its minimum distance equals the minimum Hamming weight of a nonzero codewords of* C*, i.e.,*

$$\Delta(C) = \min_{\substack{c \in C \\ c \neq \mathbf{0}}} \mathrm{wt}(c) \ .$$

Exercise 2 (Systematic or standard form) *Let* C *be an* $[n, k]_q$ *linear code. Prove that after permuting coordinates if necessary,* C *has a generator matrix of the form* $[I_k \mid G']^T$ *where* I_k *is the* $k \times k$ *identity matrix and* G' *is some* $k \times (n-k)$ *matrix.*

A generator matrix in the form $[I_k \mid G']^T$ is said to be in *systematic form*. When such a generator matrix is used for encoding, the encoding is called *systematic*: the first k symbols of the codeword are just the message symbols, and the remaining $n-k$ symbols comprise redundant check symbols. Thus, after permuting coordinates if needed, every linear code admits a systematic encoding. The above-mentioned encoding map for the $[7, 4, 3]$ Hamming code was systematic.

3.1 Parity check matrices

The following is a good way to flex your basic linear algebra muscles (Exercise 2 is a useful way to proceed):

Exercise 3 *Prove that* C *is an* $[n, k]_q$ *code if and only if there is a matrix* $H \in \mathbb{F}_q^{(n-k)\times n}$ *of full row rank such that*

$$C = \{c \in \mathbb{F}_q^n \mid Hc = \mathbf{0}\} \ .$$

In other words, C is the nullspace of H. Such a matrix H is called a *parity check matrix* for C. A linear code can thus be compactly represented by either its generator matrix or its parity check matrix. The minimum distance of a code has the following characterization in terms of the parity check matrix.

Lemma 9 *If* H *is the parity check matrix of a linear code* C*, then* $\Delta(C)$ *equals the minimum number of columns of* H *that are linearly dependent.*

3.2 Hamming code revisited

The Hamming code is best understood by the structure of its parity check matrix. This will also allow us to generalize Hamming codes to larger lengths.

We defined the $C_{\text{Ham}} = [7, 4, 3]_2$ Hamming code using generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

If we define the matrix

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

then one can check that $HG = \mathbf{0}$ and that H is a parity check matrix for C_{Ham}. Note that H has a rather nice structure: its columns are the integers 1 to 7 written in binary.

Correcting single errors with the Hamming code: Suppose that y is a corrupted version of some (unknown) codeword $c \in C_{\text{Ham}}$, with a single bit flipped. We know by the distance property of C_{Ham} that c is uniquely determined by y. In particular, a naive method to determine c would be to flip each bit of y and check if the resulting vector is in the null space of H.

A more slick (and faster) way to correct y is as follows. We know that $y = c + e_i$ where e_i is the column vector of all zeros except a single 1 in the i'th position. Note that $Hy = H(c + e_i) = Hc + He_i = He_i =$ the ith column of H. The ith column of H is the binary representation of i, and thus this method recovers the location i of the error.

Definition 10 (Syndrome) *The vector Hy is said to be the syndrome of y.*

Generalized Hamming codes: Define H_r to be the $r \times (2^r - 1)$ matrix where column i of H_r is the binary representation of i. This matrix must contain e_1 through e_r, which are the binary representations of all powers of two from 1 to 2^{r-1}, and thus has full row rank.

Now we can define the r'th generalized Hamming code

$$C^{(r)}_{\text{Ham}} = \{c \in \mathbb{F}_2^{2^r-1} \mid H_r c = \mathbf{0}\}.$$

to be the binary code with parity check matrix H_r.

Lemma 11 *$C^{(r)}_{\text{Ham}}$ is an $[2^r - 1, 2^r - 1 - r, 3]_2$ code.*

PROOF: Since H_r has rank r, it follows that the dimension of $C^{(r)}_{\text{Ham}}$ equals r. By Lemma 9, we need to check that no two columns of H_r are linearly dependent, and there are 3 linearly dependent

columns in H_r. The former follows since the columns of H_r are all distinct. For the latter, note that the first 3 columns of H_r, being the binary representations of $1, 2, 3$, add up to 0. $\square$

Despite its simplicity, the Hamming code is amazing in that it is optimal in the following sense.

Lemma 12 *If C is a binary code of block length n and minimum distance 3, then*

$$|C| \leq \frac{2^n}{n+1} \ . \tag{1}$$

It follows that the Hamming code $C^{(r)}_{\text{Ham}}$ has the maximum possible number of codewords (and thus has largest rate) amongst all binary codes of block length $2^r - 1$ and minimum distance at least 3.

PROOF: This is a special case of the "Hamming volume" (upper) bound on the size of codes. For each $c \in C$, define its neighborhood $N(c) = \{y \in \{0,1\}^n \mid \Delta(y, c) \leq 1\}$ of all strings that differ in at most one bit from c. Since C has distance 3, by the triangle inequality we have $N(c) \cap N(c') = \emptyset$ for $c \neq c' \in C$. Therefore

$$2^n \geq |\bigcup_{c \in C} N(c)| = \sum_{c \in C} |N(c)| = |C| \cdot (n+1)$$

giving the desired upper bound on $|C|$. Note that $C^{(r)}_{\text{Ham}}$ has dimension $2^r - 1 - r$, and therefore size $2^{2^r-1}/2^r$ which meets the upper bound (1) for block length $n = 2^r - 1$. $\square$

The obvious generalization of the above argument to arbitrary distances d gives the following bound. This is called the *Hamming bound* (also sometimes called the *Volume bound*). We will meet this bound again and also discuss some more sophisticated combinatorial bounds a few lectures down.

Lemma 13 (Hamming bound) *Let C be a binary code of block length n and distance d. Then*

$$|C| \leq \frac{2^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}} \ . \tag{2}$$

Note that for equality to hold in (2) something remarkable must happen. Hamming balls of radius $\lfloor \frac{d-1}{2} \rfloor$ around the codewords must cover each point in $\{0,1\}^n$ *exactly once*. Thus we would have a *perfect packing* of non-overlapping Hamming spheres that cover the full space. (There is no way to pack balls in Euclidean space in such a way, with no "holes" in between!) Codes which meet the bound (2) are therefore called *perfect codes*.

By Lemma 12, Hamming codes are perfect. It turns out that perfect codes are a rare phenomenon. The following theorem (which is beyond the scope of this course) enumerates all (binary) perfect codes:

Theorem 14 (Tietavainen and van Lint) *The following are all the binary perfect codes:*

- *The Hamming codes $C^{(r)}_{\text{Ham}}$*

- *The* $[23, 12, 7]_2$ *Golay code*
- *The trivial codes consisting of just one codeword, or the whole space, or the repetition code* $\{0^n, 1^n\}$ *for odd* n

The Golay code Gol_{23} is a remarkable object with many equivalent definitions. The following defines it as a so-called "cyclic" code: Gol_{23} consists of $(c_0, c_1, \ldots, c_{22}) \in \{0,1\}^{23}$ when the polynomial $c(X) = c_0 + c_1 X + \cdots + c_{22} X^{22}$ is divisible by $g(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}$ in the ring $\mathbb{F}_2[X]/(X^{23} - 1)$.

3.3 Dual of a code

Since a linear code is a subspace of $\mathbb{F}_q^n$ we can define its dual or orthogonal space.

Definition 15 (Dual code) *If* $C \subseteq \mathbb{F}_q^n$ *is a linear code, then its dual, denoted* $C^\perp$*, is a linear code over* $\mathbb{F}_q$ *defined as*

$$C^\perp = \{z \in \mathbb{F}_q^n \mid \langle z \cdot c \rangle = 0 \ \forall \ c \in C\}.$$

where $\langle z, c \rangle = \sum_{i=1}^n z_i c_i$ *is the dot product over* $\mathbb{F}_q$ *of vectors* z *and* c*.*

Using Exercise 3, verify the following.

Exercise 4
1. *If* C *is an* $[n, k]_q$ *linear code, then* $C^\perp$ *is an* $[n, n-k]_q$ *linear code.*
2. $(C^\perp)^\perp = C$*.*
3. *If* H *is a parity check matrix for* C*, then* H^T *is a generator matrix for* $C^\perp$*. Equivalently, if* G *is a generator matrix for* C*, then* G^T *is a parity check matrix for* $C^\perp$*.*

Unlike vector spaces over $\mathbb{R}$, where the dual (or orthogonal complement) $W^\perp$ of a subspace $W \subseteq \mathbb{R}^n$ satisfies $W \cap W^\perp = \{\mathbf{0}\}$ (and $W + W^\perp = \mathbb{R}^n$), for subspaces of $\mathbb{F}_q^n$, C and $C^\perp$ can intersect non-trivially. One can have $C^\perp \subseteq C$ (such a code C is called *self-orthogonal*) or even $C = C^\perp$ (called *self-dual codes*).

Exercise 5 *For every even* n*, give an example of a self-dual binary linear code of block length* n*.*

A rather mysterious phenomenon in coding theory is that for many constructions of good codes, their dual also has some nice properties. We will now discuss the dual of the Hamming code, a code called the *Hadamard code* which has been highly influential and played a fundamental role in the computer-scientists' study of coding theory.

3.4 Hadamard code

The dual of the Hamming code $C_{\mathrm{Ham}}^{(r)}$ has as generator matrix $G_r = (H_r)^T$, which is a $(2^r - 1) \times r$ matrix whose rows are all non-zero bit vectors of length r. This is a $[2^r - 1, r]_2$ code and is called the *simplex code*. The *Hadamard code* is obtained by adding the all-zeroes row to G_r.

Definition 16 (Hadamard code) *The binary Hadamard code* Had_r *is a* $[2^r, r]_2$ *linear code whose* $2^r \times r$ *generator matrix has all* r*-bit vectors as its rows. Thus the encoding map for the Hadamard code encodes* $x \in \mathbb{F}_2^r$ *by a string in* $\mathbb{F}_2^{2^r}$ *consisting of the dot product* $\langle x, a \rangle$ *for every* $a \in \mathbb{F}_2^r$*.*

The Hadamard code can also be defined over $\mathbb{F}_q$*, but encoding a message in* $\mathbb{F}_q^k$ *with its dot product with every vector in* $\mathbb{F}_q^k$*.*

We note that the Hadamard code is the most redundant linear code in which no two codeword symbols are equal in every codeword. Hadamard codes have excellent distance property:

Lemma 17 *The Hadamard code* Had_r *(as well as the Simplex code) has minimum distance* 2^{r-1}*. The* q*-ary Hadamard code of dimension* r *has distance* $(1 - 1/q)q^r$*.*

PROOF: We prove that for $x \neq 0$, $\langle x, a \rangle \neq 0$ for exactly 2^{r-1} (i.e., half of the) elements $a \in \mathbb{F}_2^r$. Assume for definiteness that $x_1 = 1$. Then for every a, $\langle x, a \rangle + \langle x, a + e_1 \rangle = x_1 = 1$, and therefore exactly one of $\langle x, a \rangle$ and $\langle x, a + e_1 \rangle$ equals 1. The proof for the q-ary case is similar. □

We will later see that binary codes cannot have relative distance more than $1/2$ (unless they only have a fixed constant number of codewords). Thus the relative distance of Hadamard codes is optimal, but their rate is (necessarily) rather poor.

Comment: The *first order Reed-Muller code* is a code that is closely related to the Hadamard code. Linear algebraically, it is simply the subspace spanned by the Hadamard code and the all 1's vector (i.e., the union of the Hadamard code H and its coset $H + \mathbf{1}$). It maps a message $m_1, m_2, \ldots, m_r, m_{r+1}$ to $(m_1 a_1 + \cdots + m_r a_r + m_{r+1})_{a \in \mathbb{F}_2^r}$, or equivalently the evaluations $(M(a))_{a \in \mathbb{F}_2^r}$ of the r-variate polynomial $M(X_1, X_2, \ldots, X_r) = \sum_{i=1}^{r} m_i X_i + m_{r+1}$. It is a $[2^r, r+1, 2^{r-1}]_2$ code. We will later see that no binary code of block length n and relative distance $1/2$ can have more than $2n$ codewords, so the first order Reed-Muller code is optimal in this sense.

Code families and Asymptotically good codes

The Hamming and Hadamard codes exhibit two extremes in the trade-off between rate and distance. Hamming codes have rate approaching 1 (and in fact optimal rate) but their distance is only 3. Hadamard codes have (optimal) relative distance $1/2$, but their rate approaches 0. A natural question this raises is whether there are codes which have both good rate and good relative distance (say, with neither of them approaching 0 for large block lengths).

To formulate this question formally, and also because our focus in this course is on the asymptotic behavior of codes for large block lengths, we consider code families. Specifically, we define a family of codes to be an infinite collection $\mathcal{C} = \{C_i | i \in \mathbb{N}\}$ where C_i is a q_i-ary code of block length n_i with $n_i > n_{i-1}$ and $q_i \geq q_{i-1}$. Most of the constructions of codes we will encounter in this book will naturally belong to an infinite family of codes that share the same general structure and properties (and it is usually these asymptotic properties that will often guide our constructions).

We have defined the alphabet sizes q_i of the codes in the family to also grow with i (and n_i). Code families where $q_i = q$ for all i for some fixed q will be of special interest (and turn out to be more

challenging to understand and construct). We will call such families as *code families over a fixed alphabet* or more specifically as q-ary code families.

The notions of rate and relative distance naturally extend to code families. The *rate* of an infinite family of codes $\mathcal{C}$ is defined as

$$R(\mathcal{C}) = \liminf_i \left\{ \frac{k_i}{n_i} \right\} .$$

The *(relative) distance* of a family of codes $\mathcal{C}$ equals

$$\delta(\mathcal{C}) = \liminf_i \left\{ \frac{\Delta(C_i)}{n_i} \right\} .$$

A q-ary family of codes is said to be *asymptotically good* if both its rate and relative distance are bounded away from zero, i.e., if there exist constants $R_0 > 0$ and $\delta_0 > 0$ such that $R(\mathcal{C}) \geq R_0$ and $\delta(\mathcal{C}) \geq \delta_0$.

In this terminology, the question raised above concerning (binary) codes with both good rate and good distance, can be phrased as: "Are there asymptotically good binary code families?"

For a code family, achieving a large rate and large relative distance are naturally conflicting codes, and there are fundamental trade-offs between these. We will study some of these in the course. A good deal is known about the existence (or even explicit construction) of good codes and as well as limitations of codes, but the best possible asymptotic trade-off between rate and relative distance for binary codes remains a fundamental open question. In fact, the asymptotic bounds have seen no improvement since 1977!

Introduction to Coding Theory **CMU: Spring 2010**

Notes 2: Gilbert-Varshamov bound

January 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

1 Asymptotically good codes and Gilbert-Varshamov bound

We begin by answering the question raised at the end of the previous notes on the existence of asymptotically good codes.

Suppose we are interested in q-ary codes (not necessarily linear) of block length n and minimum distance d that have many codewords. What is the largest size such a code can have? This is a fundamental quantity for which we define a notation below.

Definition 1 *Let $A_q(n,d)$ be the largest size of a q-ary code of block length n and minimum distance d. The binary case is of special importance, and in this case $A_2(n,d)$ is denoted simply as $A(n,d)$.*

There is a natural greedy approach to construct a code of distance at least d: start with any codeword, and keep on adding codewords which have distance at least d from all previously chosen codewords, until we can proceed no longer. Suppose this procedure halts after picking a code C. Then Hamming balls in $\{0,1,\ldots,q-1\}^n$ of radius $d-1$ centered at the codewords of C must cover the whole space. (Otherwise, we can pick one more codeword which has distance at least d from every element of C, and the process would not have terminated.)

Definition 2 *For integers q, n, ℓ, denote by $\mathrm{Vol}_q(n,\ell)$ the volume of (i.e., the number of strings in) a Hamming ball of radius ℓ in $\{0,1,\ldots,q-1\}$. Note that this number does not depend on where the ball is centered and equals*

$$\mathrm{Vol}_q(n,\ell) = \sum_{j=0}^{\ell} \binom{n}{j} (q-1)^j \ .$$

Therefore, the greedy procedure terminates with a code C satisfying

$$|C| \cdot \mathrm{Vol}_q(n, d-1) \geq q^n \ .$$

We therefore have the following lower bound.

Lemma 3 (Gilbert-Varshamov bound) *The maximal size of a q-ary code of block length n and distance d satisfies*

$$A_q(n,d) \geq \frac{q^n}{\mathrm{Vol}_q(n,d-1)} = \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j}(q-1)^j} \ . \tag{1}$$

There also exist linear codes of size given by the Gilbert-Varshamov bound:

Exercise 1 *By a suitable adaptation of the greedy procedure, prove that there also exists a linear code over $\mathbb{F}_q$ of dimension at least $n - \lfloor \log_q \mathrm{Vol}_q(n, d-1) \rfloor$.*

The Gilbert-Varshamov bound was actually proved in two independent works (Gilbert, 1952) and (Varshamov, 1957). The latter actually proved the existence of *linear codes* and in fact got a slightly sharper bound stated below. (You can verify that the Hamming code in fact attains this bound for $d = 3$.)

Exercise 2 *For every prime power q, and integers n, k, d, prove that there exists an $[n, k, d]_q$ linear code with*

$$k \geq n - \lfloor \log_q \Big(\sum_{j=0}^{d-2} \binom{n-1}{j} (q-1)^j \Big) \rfloor - 1 \ .$$

In fact, one can prove that a random linear code almost matches the Gilbert-Varshamov bound with high probability, so such linear codes exist in abundance. But before stating this, we will switch to the asymptotic viewpoint, expressing the lower bound in terms of the rate vs. relative distance trade-off.

1.1 Entropy function and volume of Hamming balls

We now give an asymptotic estimate of the volume $\mathrm{Vol}_q(n, d)$ when $d = pn$ for $p \in [0, 1 - 1/q]$ held fixed and n growing. This volume turns out to be very well approximated by the exponential $q^{h_q(p)n}$ where $h_q()$ is the "entropy function" defined below.

Definition 4 (Entropy function) *For a positive integer $q \geq 2$, define the q-ary entropy function $h_q : [0, 1] \to \mathbb{R}$ as follows:*

$$h_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q (1-x) \ .$$

Of special interest is the binary entropy function

$$h(x) = x \log \frac{1}{x} + (1-x) \log \frac{1}{1-x}$$

where we use the notational convention that $\log = \log_2$.

If X is the $\{0, 1\}$-valued random variable such that $\mathbb{P}[X = 1] = p$ and $\mathbb{P}[X = 0] = 1 - p$, then the Shannon entropy of X, $H(X)$, equals $h(p)$. In other words, $h(p)$ is the uncertainty in the outcome of a p-biased coin toss (which lands heads with probability p and tails with probability $1 - p$). The function h_q is continuous and increasing in the interval $[0, 1 - 1/q]$ with $h_q(0) = 0$ and $h_q(1-1/q) = 1$. The binary entropy function is symmetric around the $x = 1/2$ line: $h(1-x) = h(x)$.

We can define the inverse of the entropy function as follows. For $y \in [0, 1]$, the inverse $h_q^{-1}(y)$ is equal to the unique $x \in [0, 1 - 1/q]$ satisfying $h_q(x) = y$.

Lemma 5 *For an integer $q \geq 2$ and $p \in [0, 1-1/q]$,*

$$\mathrm{Vol}_q(n, pn) \leq q^{h_q(p)n} \ .$$

PROOF: We have

$$\begin{aligned}
\frac{\mathrm{Vol}_q(n,pn)}{q^{h_q(p)n}} &= \frac{\sum_{j=0}^{pn}\binom{n}{j}(q-1)^j}{(q-1)^{pn}p^{-pn}(1-p)^{-(1-p)n}} \\
&= \sum_{j=0}^{pn}\binom{n}{j}(q-1)^j(q-1)^{-pn}p^{pn}(1-p)^{(1-p)n} \\
&= \sum_{j=0}^{pn}\binom{n}{j}(q-1)^j(1-p)^n\left(\frac{p}{(q-1)(1-p)}\right)^{pn} \ .
\end{aligned}$$

Since $p \leq 1-1/q$, $\frac{p}{q-1} \leq 1-p$, and therefore the above quantity is at most

$$\sum_{j=0}^{pn}\binom{n}{j}(q-1)^j(1-p)^n\left(\frac{p}{(q-1)(1-p)}\right)^{j} = \sum_{j=0}^{pn}\binom{n}{j}(1-p)^{n-j}p^j \ .$$

The latter sum is at most

$$\sum_{j=0}^{n}\binom{n}{j}(1-p)^{n-j}p^j = 1$$

by the binomial theorem. □

The above upper bound is tight up to lower order terms. The quantity $\mathrm{Vol}_q(n, pn)$ is at least as large as $\binom{n}{pn}(q-1)^{pn}$. By Stirling's formula $m! = \sqrt{2\pi m}(m/e)^m(1+o(1))$, it follows that

$$\binom{n}{pn} \geq \left(\frac{1}{p}\right)^{pn}\left(\frac{1}{1-p}\right)^{(1-p)n}\exp(-o(n)) = 2^{h(p)n-o(n)}$$

and therefore

$$\mathrm{Vol}_q(n,pn) \geq \binom{n}{pn}(q-1)^{pn} \geq q^{h_q(p)n-o(n)} \ .$$

For a self-contained derivation of the entropy estimate for the binomial coefficients, we can work with a crude estimate of $m!$ given by the integral estimate

$$\sum_{i=1}^{m-1}\ln i \leq \int_1^m \ln x \leq \sum_{i=2}^{m}\ln i$$

which gives

$$\frac{m^m}{e^{m-1}} \leq m! \leq \frac{m^{m+1}}{e^{m-1}} \ .$$

This immediately gives the lower bound

$$\binom{n}{pn} \geq 2^{h(p)n}\cdot\frac{1}{en\sqrt{p(1-p)}} \geq 2^{h(p)n-o(n)} \ .$$

We summarize the above discussion in the following important estimate.

Lemma 6 *For positive integers $n, q \geq 2$ and real p, $0 \leq p \leq 1 - 1/q$,*

$$q^{(h_q(p)-o(1))n} \leq \mathrm{Vol}_q(n, pn) \leq q^{h_q(p)n} \ .$$

1.2 Asymptotic form of GV bound

Combining the greedy construction of Lemma 3 with the estimate of the Hamming volume from Lemma 6 gives the following asymptotic version of the Gilbert-Varshamov bound.

Theorem 7 (Asymptotic Gilbert-Varshamov bound) *For every q and $\delta \in [0, 1 - 1/q]$, there exists an infinite family $\mathcal{C}$ of q-ary codes with rate*

$$R(\mathcal{C}) \geq 1 - h_q(\delta) - o(1) \ .$$

(In fact, such codes exist for every block length.)

Since $h_q(\delta) < 1$ for $\delta < 1 - 1/q$, the above implies that for every $\delta < 1 - 1/q$ there exists an asymptotically good family of q-ary codes of rate at least $R_0(\delta) > 0$ and relative distance at least δ. By Exercises 1 and 2 this also holds for linear codes over $\mathbb{F}_q$. We now give an alternate proof based on the probabilistic method.

1.3 Random linear codes attain the GV bound

Theorem 8 *For every prime power q, $\delta \in [0, 1 - 1/q)$, $0 < \epsilon < 1 - h_q(p)$, and sufficiently large positive integer n, the following holds for $k = \lceil (1 - h_q(\delta) - \epsilon)n \rceil$. If $G \in \mathbb{F}_q^{n \times k}$ is chosen uniformly at random, then the linear code with G as generator matrix has rate at least $(1 - h_q(\delta) - \epsilon)$ and relative distance at least δ with probability at least $1 - e^{-\Omega(n)}$.*

Proof: The claim about rate follows whenever G has full column rank. The probability that the i'th column is in the span of the first $(i-1)$ columns is at most q^{i-1}/q^n. By a union bound, G has rank k with probability at least $1 - \frac{k}{q^{n-k}} \geq 1 - e^{-\Omega(n)}$.

For each nonzero $x \in \mathbb{F}_q^k$, the vector Gx is a uniformly random element of $\mathbb{F}_q^k$. (Indeed, say that $x_k \neq 0$, then conditioned on the choice of the first $k-1$ columns G' of G, $Gx = G'x + g_k x_k$ is uniformly distributed since the k'th column g_k is chosen uniformly at random from $\mathbb{F}_q^n$.) Therefore the probability that $\mathrm{wt}(Gx) \leq \delta n$ is at most

$$\frac{\mathrm{Vol}_q(n, \delta n)}{q^n} \leq q^{(h_q(\delta)-1)n} \ .$$

Now a union bound over all nonzero x implies that the probability that the code generated by the columns of G has distance at most δn is bounded from above by

$$q^k q^{(h_q(\delta)-1)n} \leq q^{(1-h_q(\delta)-\epsilon)n+1} q^{(h_q(\delta)-1)n} = q \cdot q^{-\epsilon n} \leq e^{-\Omega(n)} \ .$$

We conclude that with probability at least $1 - e^{-\Omega(n)}$, the code generated by G has relative distance at least δ and rate at least $1 - h_q(\delta) - \epsilon$. □

Exercise 3 *Establish a similar result by picking a random $(n-k) \times n$ parity check matrix for the code.*

1.4 Some comments on attaining/beating the GV bound

We have seen that there exist binary linear codes that meet the Gilbert-Varshamov bound, and thus have rate approaching $1-h(\delta)$ for a target relative distance of δ, $0 < \delta < 1/2$. The proof of this was non-constructive, based on an exponential time algorithm to construct such a code (by a greedy algorithm), or by picking a generator matrix (or a parity check matrix) at random. The latter leads to a polynomial time randomized Monte Carlo construction. If there were a way to ascertain if a randomly chosen linear code has the claimed relative distance, then this would be a practical method to construct codes of good distance; we will have a Las Vegas construction that picks a random linear code and then checks that it has good minimum distance. Unfortunately, given a linear code, computing (or even approximating) the value of its minimum distance is NP-hard.

A natural challenge therefore is to give an explicit (i.e., deterministic polynomial time) construction of a code that meets the Gilbert-Varshamov bound (i.e., has rate R and relative distance close to $h_q^{-1}(1-R)$). Giving such a construction of binary codes (even non-linear ones) remains an outstanding open question.

For prime powers $q = p^{2k}$ for $q \geq 49$, explicit constructions of q-ary linear codes that not only attain but surpass the GV bound are known! These are based on algebraic geometry and a beautiful construction of algebraic curves with many rational points and small genus. This is also one of the rare examples in combinatorics where we know an explicit construction that beats the parameters obtained by the probabilistic method. (Another notable example is the Lubotzky-Phillips-Sarnak construction of Ramanujan graphs whose girth surpasses the probabilistic bound.)

What about codes over smaller alphabets, and in particular binary codes? The Hamming upper bound on size of codes (Lemma 13 in Notes 1) leads to the asymptotic upper bound $R \leq 1-h(\delta/2)$ on the rate. This is off by a factor of 2 in the coefficient of δ compared to the achievable $1-h(\delta)$ rate. We will later see improvements to the Hamming bound, but the best bound will still be far from the Gilbert-Varshamov bound. Determining the largest rate possible for binary codes of relative distance $\delta \in (0, 1/2)$ is another fundamental open problem in the subject. The popular conjecture seems to be that the Gilbert-Varshamov bound on rate is asymptotically tight (i.e., a binary code of relative distance δ must have rate $1-h(\delta)+o(1)$), but arguably there is no strong evidence that this must be the case.

While we do not know explicit constructions of binary codes approaching the GV bound, it is still interesting to construct codes which achieve good trade-offs. This leads to the following questions, which are the central questions in coding theory for any noise model (once some existential bounds are established on the trade-offs, the questions below pertaining to the worst-case or adversarial noise model where we impose no restriction on the channel other than a limit on the total number of errors caused):

1. Can one explicitly construct an asymptotically good family of binary codes with a "good" rate vs. relative distance trade-off?
2. Can one construct such codes together with an efficient algorithm to correct a fraction of errors approaching half-the-relative distance (or even beyond)?

We will answer both the questions in the affirmative in this course.

Introduction to Coding Theory **CMU: Spring 2010**

Notes 3: Stochastic channels and noisy coding theorem bound

January 2010

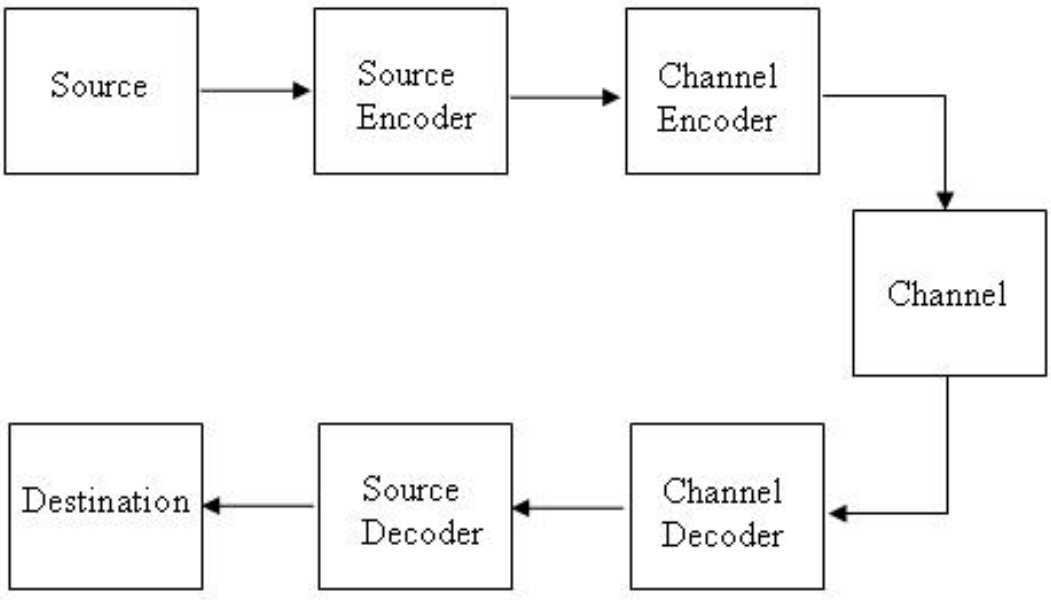
Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

We now turn to the basic elements of Shannon's theory of communication over an intervening noisy channel.

1 Model of information communication and noisy channel

To quote Shannon from his paper *A Mathematical theory of communication*: "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point." The basic setup of the communication problem consists of a source that generates digital information which is to reliably communicated to a destination through a channel, preferably in the most efficient manner possible. This "destination" could be spatially separated (eg., a distant satellite is sending images of Jupiter back to the space station on Earth), or could be temporally separated (eg., we want to retrieve date stored on our hard disk at a later point of time).

The following is a schematic of the communication model:



The first step in the communication model is to exploit the redundancy in the output of the source and compress the information to economize the amount of "raw, non-redundant" data that must be transmitted across the channel. This data compression step in called *source coding.* If at each time step the source outputs an i.i.d copy of a random variable Z supported on a finite set $\mathcal{Z}$, then

Shannon's source coding theorem states that one can compress its output to $H(Z)$ bits per time step (on average, over n i.i.d samples from the source Z, as $n \to \infty$). In other words n samples from the source can be coded as one of $M \approx 2^{H(Z)n}$ possible outputs. Here $H(Z)$ is the fundamental Shannon entropy defined as

$$H(Z) = \sum_{z \in \mathcal{Z}} \mathbb{P}[Z = z] \log \frac{1}{\mathbb{P}[Z = z]} \ . \tag{1}$$

where log is to the base 2. Thus the entropy of a fair coin toss is 1, and that of a p-biased coin toss is $h(p) = -p \log p - (1-p) \log(1-p)$. The *source decoder* at the other end of the communication channel then decompresses the received information into (hopefully) the original output of the source.

The output of the source coder, say m, must then be communicated over a noisy channel. The channel's noisy behavior causes errors in the received symbols at the destination. To recover from the errors incurred due to the channel, one should *encode* the information output by the source coder by adding systematic redundancy to it. This is done through channel coding which maps m to a codeword c of some suitable error-correcting code (the study of channel coding will be our focus in this course).

1.1 Modeling the noisy channel

The basic channel model consists of an input alphabet $\mathcal{X}$ and output alphabet $\mathcal{Y}$. We will focus on *memoryless channels* — for each $x \in \mathcal{X}$ there is a distribution D_x on $\mathcal{Y}$ such that when input $x \in \mathcal{X}$ is fed at one end of the channel, the channel distorts it to $y \in \mathcal{Y}$ according to an independent sample drawn according to D_x. (In particular, the channel has no "state," and its behavior is independent of the history of previously transmitted symbols.) The collection of the distributions D_x comprise the "channel law" for the behavior of the channel. In a discrete memoryless channel (DMC), given by a triple $\Lambda = (\mathcal{X}, \mathcal{Y}, \Pi)$, the input and output alphabets $\mathcal{X}, \mathcal{Y}$ are finite, and therefore the channel law can be specified by a $|\mathcal{X}| \times |\mathcal{Y}|$ conditional probability matrix Π which is a stochastic matrix where each row sums to 1:

$$|\mathcal{X}| \left\{ \overbrace{\left(\quad \Pi(y|x) \quad \right)}^{|\mathcal{Y}|} \right.$$

The (x, y)'th entry $\Pi(y|x)$ is the conditional probability $\mathbb{P}(Y = y | X = x)$ of the receiving y when x was transmitted on the channel.

1.2 Noisy coding and joint source-channel coding theorems

Suppose at the output of the source coder, we have a message m from one of $M \approx 2^{H(Z)n}$ possible messages (that encode n samples from the source Z), which is to be communicated across a channel $(\mathcal{X}, \mathcal{Y}, \Pi)$. Then the channel encoder it into a sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in C \subseteq \mathcal{X}^n$ for some error-correcting code C and the information is sent via n uses of the channel. At the other end, a sequence $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathcal{Y}^n$ is received with conditional probability

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} \Pi(y_i|x_i) \tag{2}$$

(due to the memoryless nature of the channel). The decoder must then map this sequence $\mathbf{y}$ into a legal codeword $c \in C$ (or equivalently into a message $m \in \mathcal{M}$).

A piece of notation: For a DMC $(\mathcal{X}, \mathcal{Y}, \Pi)$, a positive integer n, and $\mathbf{x} \in \mathcal{X}^n$, let us denote by $\Pi(\mathbf{x})$ the above distribution (2) on $\mathcal{Y}^n$ induced by the Π on input sequence $\mathbf{x}$.

Theorem 1 (Shannon's noisy coding theorem) *For every discrete memoryless channel $\Lambda = (\mathcal{X}, \mathcal{Y}, \Pi)$, there exists a real number $C_0 = C_0(\Lambda)$ called its channel capacity, such that the following holds for every $R < C_0$. For all large enough n, there exists an integer $M \geq 2^{Rn}$ and*

1. *an encoding map* $\mathrm{Enc} : \{1, 2, \ldots, M\} \to \mathcal{X}^n$ *(of some error-correcting code over alphabet $\mathcal{X}$ of rate $R/\log|\mathcal{X}|$), and*
2. *a decoding map* $\mathrm{Dec} : \mathcal{Y}^n \to \{1, 2, \ldots, M\} \cup \{\text{fail}\}$

such that for every $m \in \{1, 2, \ldots, M\}$

$$\mathbb{P}_\Pi[\mathrm{Dec}(\Pi(\mathrm{Enc}(m))) = m] \geq 1 - 2^{-\Omega_{R,C_0}(n)}$$

where the probability is over the behavior of the channel Π (on input $\mathrm{Enc}(m)$*).*

Further, the capacity C_0 is given by the expression

$$\max_{p \in \mathrm{Dist}_{\mathcal{X}}} H(Y) - H(Y|X)$$

where the maximum is taken over all probability distributions p on $\mathcal{X}$. In the above, $H(Y)$ is the entropy of the $\mathcal{Y}$-valued random variable Y with distribution function

$$\mathbb{P}[Y = y] = \sum_{x \in \mathcal{X}} \mathbb{P}(Y = y|X = x)p(x) = \sum_{x \in \mathcal{X}} \Pi(y|x)p(x)$$

and $H(Y|X)$ is the conditional entropy

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x)H(Y|X = x) = \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} \Pi(y|x) \log \frac{1}{\Pi(y|x)} \ .$$

Remark 2 *The quantity $H(Y) - H(Y|X)$ is called the mutual information between X and Y, and denoted $I(X,Y)$. It represents the decrease in uncertainty of a random variable Y given the knowledge of random variable X, which intuitively captures how much information X reveals about Y. If Y is independent of X, then $H(Y|X) = H(Y)$, and $I(X,Y) = 0$. On the other hand if $Y = f(X)$ for some function f (i.e., Y is determined by X), then $H(Y|X) = 0$ and $I(X,Y) = H(Y)$.*

Combining Shannon's source coding and noisy coding theorems, and the two-stage communication process comprising a separate source coding stage followed by channel coding stage, one can conclude that reliable communication of the output of a source Z on a noisy channel Λ is possible as long as $H(Z) < C_0(\Lambda)$, i.e., the source outputs data at a rate that is less than the capacity of the channel. This result has a converse (called the converse to the joint source-channel coding theorem) that says that if $H(Z) > C_0(\Lambda)$ then reliable communication is not possible.

Together, these imply a "separation theorem," namely that it is information-theoretically optimal to do source and channel coding separately, and thus one can gain modularity in communication system design without incurring any loss in rate of data transfer. While this converse to the joint source-channel coding theorem is rather intuitive in the setting of point-to-point communication between a sender and a receiver, it is worth remarking that the separation theorem breaks down in some scenarios with multiple users and correlated sources.

We will not prove Shannon's theorem in the above generality here, but content ourselves with establishing a special case (for the binary symmetric channel). The proof for the general case follows the same general structure once some basic information theory tools are set up, and we will remark briefly about this at the end. But first we will see some important examples of noisy channels.

2 Examples of channels

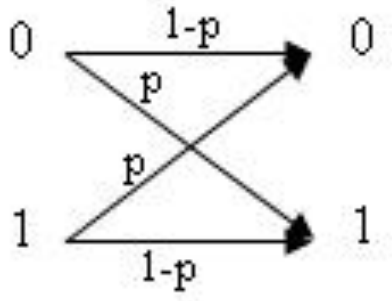
A discrete channel with finite input and output alphabets $\mathcal{X}$ and $\mathcal{Y}$ respectively, specified by the conditional probability matrix $\Pi(y|x)$, can also be represented pictorially by an input-output diagram, which is a bipartite graph with nodes on left identified with $\mathcal{X}$ and nodes on right identified with $\mathcal{Y}$ and a directed edge between $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with weight $\Pi(y|x)$.

2.1 Binary symmetric channel

The *Binary Symmetric Channel* (BSC) has input alphabet $\mathcal{X} = \{0,1\}$ and output alphabet $\mathcal{Y} = \{0,1\}$. The BSC is parameterized by a real number p, $0 \leq p \leq 1/2$ called the *crossover probability*, and often denoted BSC_p. The channel flips its input with probability p, in other words,

$$\Pi(y|x) = \begin{cases} p & \text{if } y = x \\ 1-p & \text{if } y = 1-x \end{cases}$$

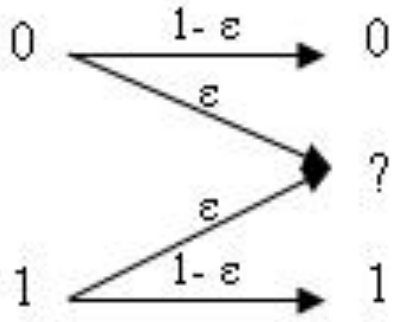
Pictorially, BSC_p can be represented as



If a uniform input $X \in \{0,1\}$ is fed as input to BSC_p, then the output Y is also uniformly distributed. Given $X = x$, Y is distributed as a p-biased coin, and $H(Y|X = x) = h(p)$. Thus $H(Y|X) = h(p)$, and therefore $I(X,Y) = H(Y) = H(Y|X) = 1 - h(p)$. It can be checked that the uniformly distributed X maximizes $I(X,Y)$, and so Shannon's theorem implies that $1 - h(p)$ is the capacity of BSC_p. We will shortly prove this special case of Shannon's theorem.

2.2 Binary erasure channel

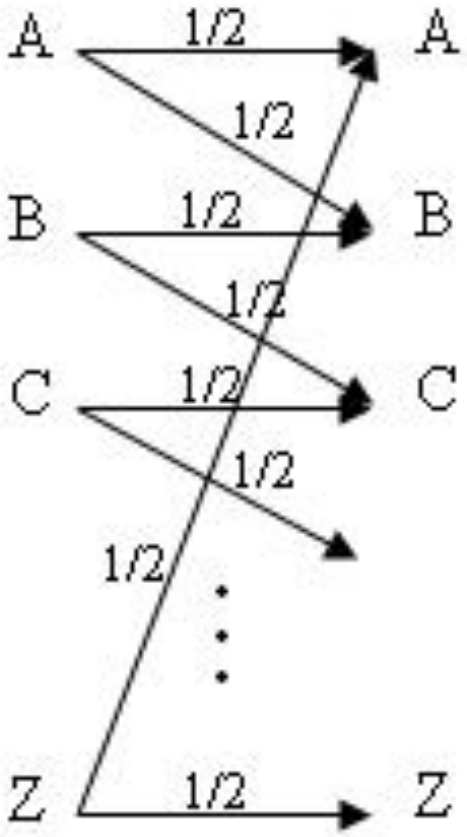
The Binary Erasure Channel (BEC) is parameterized by a real ϵ, $0 \leq \epsilon \leq 1$, which is called the *erasure probability*, and is denoted BEC_ϵ. Its input alphabet is $\mathcal{X} = \{0,1\}$ and output alphabet is $\mathcal{Y} = \{0,1,?\}$. Upon input $x \in \mathcal{X}$, the channel outputs x with probability $1 - \epsilon$, and outputs ? (corresponding to erasing the symbol) with probability ϵ. (It never flips the value of a bit.) Pictorially:



When a bit string of length n for large n is transmitted across BEC_ϵ, with high probability only $\approx (1-\epsilon)n$ bits are received unerased at the other end. This suggests that the maximum rate at which reliable communication is possible is at most $1-\epsilon$. It turns out that a rate approaching $1-\epsilon$ can be achieved, and the capacity of BEC_ϵ equals $1 - \epsilon$.

2.3 Noisy Typewriter Channel

The noisy typewriter channel is given by the following diagram:



If we restrict the code to send only one of the symbols $\{A, C, E, \ldots, Y\}$ in each channel use, we can communicate one of 13 possible messages with **zero** error. Therefore the capacity of the channel is at least $\log_2 13$. One can prove that this rate is the maximum possible and the capacity of the channel is exactly $\log 13$. (Indeed, this follows from Shannon's capacity formula: Since $|\mathcal{Y}| = 26$, $H(Y)$ is at most $\log 26$. Also $H(Y|X) = 1$ for every distribution of the channel input X. Hence $H(Y) - H(Y|X) \leq \log 13$.)

Note that we can achieve a rate equal to capacity with *zero* probability of miscommunication. For the BSC_p with $p > 0$ on the other hand, zero error communication is not possible at *any* positive rate, since for every pair of strings $x, x' \in \{0,1\}^n$, there is a positive probability that x will get distorted to x' by the noise caused by the BSC_p.

The study of zero error capacity of channels was introduced in another classic work of Shannon. Estimating the zero error capacity of even simple channels (such as the 5-cycle) has led to some beautiful results in combinatorics, including Lovász's celebrated work on the Theta function.

2.4 Continuous Output Channel

We now see an example of a continuous output channel that is widely used to model noise and compare the performance (typically via simulation) of different coding schemes. The binary input

additive white Gaussian noise (BIAWGN) channel has input alphabet $\mathcal{X} = \{1, -1\}$ (it is more convenient to encode binary symbols by ± 1 instead of $\{0, 1\}$) and output alphabet $\mathcal{Y} = \mathbb{R}$. The input $x \in \{1, -1\}$ is "modulated" into the real number βx and the channel adds additive noise distributed according to $N(0, \sigma^2)$ to βx. Thus the output distribution is a Gaussian with mean βx and variance σ^2. Formally

$$\mathbb{P}[Y \leq y | X = x] = \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(z-\beta x)^2/(2\sigma^2)} \, dz \ .$$

The quantity $(\beta/\sigma)^2$ is commonly referred to as the *signal-to-noise ratio* (SNR for short), with β^2 corresponding to the energy per input bit and σ^2 corresponding to the amount of noise. The SNR is usually measured in decibel units (dB), and expressed as the value $10 \log_{10}(\beta/\sigma)^2$. As one might expect, the capacity of the AWGN channel increases as its SNR increases.

3 Shannon's capacity theorem for the binary symmetric channel

We now turn to establishing the capacity of BSC_p to be $1 - h(p)$.

3.1 Connection to minimum distance

First, let us connect this question to the Hamming world. If we have a family of binary codes of relative distance more than $(2p + \epsilon)$, then we claim that this enables communicating on the BSC_p with exponentially small probability of miscommunication. The reason is that by the Chernoff bound for independent Bernoulli random variables (stated below), the probability that at least $(p + \epsilon/2)n$ are corrupted out of n bits transmitted on a BSC_p is exponentially small. When the number of errors is less than $(p+\epsilon/2)n$, the received word has a unique closest codeword in Hamming distance, which is also the original transmitted codeword.

Lemma 3 (Chernoff bound for i.i.d. Bernoulli random variables) *If $X_1, X_2, \ldots X_n$ are i.i.d. $\{0, 1\}$-valued random variables with $\mathbb{P}[X_i = 1] = p$, then for every $\epsilon > 0$, for large enough n the following tail estimates hold:*

$$\mathbb{P}[\sum_{i=1}^{n} X_i \geq (p + \epsilon)n] \leq 2^{\frac{-\epsilon^2 n}{3}}$$

$$\mathbb{P}[\sum_{i=1}^{n} X_i \leq (p - \epsilon)n] \leq 2^{\frac{-\epsilon^2 n}{3}}$$

Together with the Gilbert-Varshamov bound, we conclude the existence of codes of rate at least $1 - h(2p)$ for reliable communication on BSC_p. This rate is positive only for $p < 1/4$, and falls short of the bound $1 - h(p)$ which we "know" to be the capacity of BSC_p from Shannon's general theorem.

The Hamming upper bound on rate for codes of relative distance $2p$ was also equal to $1 - h(p)$. So if the Hamming bound could be attained, we could achieve the capacity of BSC_p simply by using

codes of relative distance $2p$. However, we will soon see that the Hamming upper bound can be improved, and there are no codes of positive rate for relative distance $2p$ for $p \geq 1/4$ or of rate $1 - h(p)$ for $p < 1/4$.

3.2 Intuition: mostly disjoint packings

The key to Shannon's theorem is that we do not need every pair of codewords to differ in a fraction $2p$ of locations, but only that for *most* (as opposed to for all) points obtained by flipping about a p fraction of bits of a codeword c have no other codeword closer than c. In other words, it suffices to be able to pack $\approx 2^{(1-h(p))n}$ "mostly-disjoint" Hamming balls of radius pn so that most points in $\{0,1\}^n$ belong to at most one such Hamming ball. Indeed, we will show below (Theorem 4) that such a packing exists, and therefore one can reliably communicate on BSC_p with rate approaching $1 - h(p)$.

The intuition for the case of general discrete memoryless channels as stated in Theorem 1 is similar. For a typical sequence $x \in \mathcal{X}^n$ (chosen according to the product distribution $p^{\otimes n}$), when x is transmitted, there are $\approx 2^{H(Y|X)n}$ possible received sequences in $\mathcal{Y}^n$ (call this the "neighborhood" of x), out of a total volume of $2^{H(Y)n}$. It turns out it is possible to pick a collection of $\approx 2^{(H(Y)-H(Y|X))n}$ sequences in $\mathcal{X}^n$ whose neighborhoods are mostly disjoint. This enables reliable communication at rate approaching $H(Y) - H(Y|X)$.

3.3 Converse to capacity theorem for BSC

We now give an explanation for why $1 - h(p)$ ought to be an *upper bound* on capacity of the BSC_p. Suppose a code $C \subset \{0,1\}^n$ achieves negligible error probability for communication on BSC_p with some decoding rule $D : \{0,1\}^n \to C \cup \{\text{fail}\}$. When c is transmitted, with overwhelming probability the received word belongs to a set Typical_c of $\approx 2^{h(p)n}$ possible strings whose Hamming distance to c is close to pn (say in the range $[(p - o(1))n, (p + o(1))n]$, and these possibilities are all roughly equally likely. Therefore, in order to ensure that c is recovered with high probability from its noisy version, the decoding rule D must map most of the strings in Typicakl_c to c. Thus we must have $|D^{-1}(c)| \approx 2^{h(p)n}$ for each $c \in C$, leading to the upper bound $|C| \leq 2^{(1-h(p)+o(1))n}$.

A different way to argue about the $1 - h(p)$ upper bound is related to a discussion in our very first lecture. It is based on the observation that when communication is successful, the decoder not only recovers the transmitted codeword but also the locations of the (typically around pn) errors. The former carries $\log |C|$ bits of information, whereas the latter typically conveys $\approx h(p)n$ bits of information. Since the total amount of non-redundant information that can be reliably conveyed by n bits cannot exceed n, we again get the upper bound $|C| \leq 2^{(1-h(p)+o(1))n}$.

Exercise 1 *Develop the above arguments into a formal proof that communication at a rate of $1 - h(p) + \epsilon$ on BSC_p incurs a probability of error bounded below by an absolute constant, and in fact by $1 - 2^{-\Omega_{\epsilon,p}(n)}$ where n is the block length of the code.*

3.4 The theorem

We conclude these notes with the formal statement and proof of the capacity theorem for BSC_p.

Theorem 4 *For every $p \in (0, 1/2)$ such that $0 \leq p < \frac{1}{2}$, and every $0 < \gamma < 1/2 - p$ and all large enough integers n, there exists a $\delta = \delta(\gamma, p)$ and a code with encoding map $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ for $k = (1 - h(p+\gamma))n$ and a decoding rule $D : \{0,1\}^n \to \{0,1\}^k \cup \{\text{fail}\}$ such that*

$$\mathbb{P}_z[D(E(m) + z) = m] \geq 1 - 2^{-\delta n}$$

where the probability is over the noise z caused by BSC_p.

PROOF: The construction is by the probabilistic method. Let $\ell = k + 1$. The encoding function $\mathrm{Enc} : \{0,1\}^\ell \to \{0,1\}^n$ is chosen uniformly at random from all possible functions. In other words, for every message $m \in \{0,1\}^\ell$, the corresponding codeword, $\mathrm{Enc}(m)$ is chosen uniformly at random from $\{0,1\}^n$. (Note that this might assign the same codeword to two different messages but this (tiny) probability will be absorbed into the decoding error probability.)

Pick $\epsilon = \epsilon(\gamma) > 0$ to be a small enough constant. The decoding function D is defined as follows: $D(y) = m$ if $\mathrm{Enc}(m)$ is the unique codeword such that $\Delta(y, \mathrm{Enc}(m)) \leq (p + \epsilon)n$ and $D(y) = \text{fail}$ otherwise.

For $z \in \{0,1\}^n$, let $\mathrm{prob}(z)$ denote the probability that the noise caused by BSC_p on input the all 0's vector equals z (note that $\mathrm{prob}(z) = p^{\mathrm{wt}(z)}(1-p)^{n-\mathrm{wt}(z)}$).

Fix a message m. For each possible Enc, the probability that $D(\mathrm{Enc}(m) + z) \neq m$ taken over the noise z caused by BSC_p is at most

$$\begin{aligned}
\mathbb{P}_z[D(\mathrm{Enc}(m) + z) \neq m)] &\leq \mathbb{P}_z[\mathrm{wt}(z) > (p+\epsilon)n] + \sum_{z \in B(0,(p+\epsilon)n)} \mathrm{prob}(z)\mathbf{1}(D(\mathrm{Enc}(m) + z) \neq m) \\
&\leq 2^{-\Omega(\epsilon^2 n)} + \sum_{z \in B(0,(p+\epsilon)n)} \mathrm{prob}(z) \sum_{m' \neq m} \mathbf{1}(\Delta(\mathrm{Enc}(m) + z, \mathrm{Enc}(m')) \leq (p+\epsilon)n)
\end{aligned}$$

where the notation $\mathbf{1}(E)$ stands for the indicator random variable of the event E, the first estimate follows from the Chernoff bound, and the second estimate because when the decoding is unsuccessful when at most $(p + \epsilon)n$ errors occur, there must be some other codeword besides $\mathrm{Enc}(m)$ that is close to the received word $\mathrm{Enc}(m) + z$.

Now let us bound the expected value of this probability of miscommunication over the random choice of Enc. For each fixed z, and $m \neq m'$,

$$\begin{aligned}
\mathbb{E}_{\mathrm{Enc}}[\mathbf{1}(\Delta(\mathrm{Enc}(m) + z, \mathrm{Enc}(m')))] &\leq \mathbb{P}_{\mathrm{Enc}}[\Delta(\mathrm{Enc}(m) + z, \mathrm{Enc}(m')) \leq (p+\epsilon)n] \\
&= \frac{\mathrm{Vol}(n, (p+\epsilon)n)}{2^n} \\
&\leq 2^{-(1-h(p+\epsilon))n}
\end{aligned}$$

Therefore, by linearity of expectation

$$\begin{aligned}
\mathbb{E}_{\mathrm{Enc}}\mathbb{P}_z[D(\mathrm{Enc}(m) + z) \neq m)] &\leq 2^{-\Omega(\epsilon^2 n)} + \sum_{z \in B(0,(p+\epsilon)n)} \mathrm{prob}(z) 2^\ell 2^{-(1-h(p+\epsilon))n} \\
&\leq 2^{-\Omega(\epsilon^2 n)} + 2 \cdot 2^{-(h(p+\gamma) - h(p+\epsilon))n} < \frac{1}{2} \cdot 2^{-\delta n}
\end{aligned}$$

for some $\delta = \delta(\gamma, p) > 0$ when ϵ is chosen small enough.

We can conclude from the above for each fixed m that the probability over Enc that the error probability in communicating m (over the channel noise) exceeds $2^{-\delta n/2}$ is $2^{-\delta n/2}$. We would like to find an encoding Enc for which the error probability is low for every m simultaneously. The bound is too weak to do a union bound over all 2^ℓ messages. So we proceed as follows.

Since $\mathbb{E}_{\text{Enc}}\mathbb{P}_z[D(\text{Enc}(m) + z) \neq m)] < 2^{-1-\delta n}$ for each fixed m, this also holds on average over all choices of m. That is

$$\mathbb{E}_m\mathbb{E}_{\text{Enc}}\mathbb{P}_z[D(\text{Enc}(m) + z) \neq m)] < 2^{-1-\delta n} \ .$$

Changing the order of expectations

$$\mathbb{E}_{\text{Enc}}\mathbb{E}_m\mathbb{P}_z[D(\text{Enc}(m) + z) \neq m)] < 2^{-1-\delta n} \ .$$

Therefore there must exist an encoding Enc^* for which

$$\mathbb{E}_m\mathbb{P}_z[D(\text{Enc}^*(m) + z) \neq m)] < 2^{-1-\delta n} \ .$$

By an averaging argument, for at most $1/2$ the messages $m \in \{0,1\}^\ell$ one can have $\mathbb{P}_z[D(\text{Enc}^*(m) + z) \neq m)] \geq 2^{-\delta n}$. Expurgating these messages, we get an encoding $\text{Enc}' : \{0,1\}^{\ell-1} \to \{0,1\}^n$ and a decoding function D' such that for every $m' \in \{0,1\}^k$, $\mathbb{P}_z[D'(\text{Enc}'(m') + z) \neq m'] < 2^{-\delta n}$. This finishes the proof of the theorem. □

We remark that neither the encoding function nor the decoding function in the above proof are efficiently computable. The challenge put forth by Shannon's work is to "constructivize" his result and find explicit codes with polynomial time encoding and decoding that achieve capacity.

Exercise 2 *Prove that Theorem 4 also holds with a linear code, and that a random linear code achieves capacity of the BSC with high probability. (In fact, the proof becomes easier in this case, as no expurgation is needed at the end.)*

We end these notes by noting another connection between the Shannon and Hamming worlds. Though minimum distance is not the governing factor for achieving capacity on the BSC, a large minimum distance is necessary to to have a positive error exponent (i.e., achieve exponentially small error probability). We leave it as an exercise to justify this claim.

Introduction to Coding Theory **CMU: Spring 2010**

Notes 4: Elementary bounds on codes

January 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

We now turn to *limitations* of codes, in the form *upper bounds* on the rate of codes as a function of their relative distance. We will typically give concrete bounds on the size of codes, and then infer as corollaries the asymptotic statement for code families relating rate and relative distance. All the bounds apply for general codes and they do not take advantage of linearity. However, for the most sophisticated of our bounds, the linear programming bound, which we discuss in the next set of notes, we will present the proof only for linear codes as it is simpler in this case.

We recall the two bounds we have already seen. The Gilbert-Varshamov bound asserted the existence of (linear) q-ary codes of block length n, distance at least d, and size at least $\frac{q^n}{\mathrm{Vol}_q(n,d-1)}$. Or in asymptotic form, the existence of codes of rate approaching $1 - h_q(\delta)$ and relative distance δ. The Hamming or sphere-packing bound gave an upper bound on the size (or rate) of codes, which is our focus in these notes. The Hamming bound says that a q-ary code of block length n and distance d can have at most $\frac{q^n}{\mathrm{Vol}_q(n,\lfloor (d-1)/2 \rfloor)}$ codewords. Or in asymptotic form, a q-ary code of relative distance δ can have rate at most $1 - h_q(\delta/2) + o(1)$.

As remarked in our discussion on Shannon's theorem for the binary symmetric channel, if the Hamming bound could be attained, that would imply that we can correct *all* (as opposed to most) error patterns of weight pn with rate approaching $1 - h(p)$. Recall that there are perfect codes (such as the Hamming codes) that meet the Hamming bound. However, these codes have very small distance (3 in the case of Hamming codes). A generalization of Hamming codes called binary BCH codes (the acronym stands for the code's independent inventors Hocquenghem (1959) and Bose and Ray-Chaudhuri (1960)) show that when d is a fixed constant and the block length is allowed to grow, the Hamming bound is again tight up to lesser order terms. However, we will improve upon the Hamming bound and show that its asymptotic form (for any relative distance bounded away from zero) cannot be attained for any fixed alphabet. The proof method has some connections to *list decoding*, which will be an important focus topic later in the course.

1 Singleton bound

We begin with the simplest of the bounds:

Theorem 1 *Let C be a code of block length n and minimum distance d over an alphabet of size q. Then $|C| \leq q^{n-d+1}$.*

Proof: Suppose not, and $|C| > q^{n-d+1}$. By the pigeonhole principle there must be two codewords $c_1, c_2 \in C$, $c_1 \neq c_2$ that agree on the first $n - d + 1$ locations. But then $\Delta(c_1, c_2) \leq d - 1 < d$, contradicting the hypothesis that C has minimum distance d. □

This gives an alphabet-independent asymptotic upper bound on the rate as a function of relative distance.

Corollary 2 *The rate R and relative distance δ of a code satisfy $R \leq 1 - \delta + o(1)$.*

Though really simple, the Singleton bound is tight in general — we will later see an algebraic family of codes called Reed-Solomon codes which achieve the Singleton bound and have dimension $n - d + 1$ and minimum distance d. The family of codes which meet the Singleton bound are called *maximum distance separable* (MDS) codes.

However, Reed-Solomon and other MDS codes will be (necessarily) defined over an alphabet that grows with the block length. For code families over a fixed alphabet such as binary codes, substantial improvements to the Singleton bound are possible. We turn to such bounds next.

2 The Plotkin bound

The Gilbert-Varshamov bound asserts the existence of positive rate binary codes only for relative distance $\delta < 1/2$. The Hamming bound on the other hand does not rule out positive rate binary codes even for $\delta > 1/2$, in fact not even for any $\delta < 1$. Thus there is a qualitative gap between these bounds in terms of identifying the largest possible distance for asymptotically good binary codes. We now prove an upper bound which shows that the relative distance has to be at most $1/2$ (and thus the Hamming bound is quite weak for large δ) unless the code has very few codewords (and in particular has vanishing rate).

While the same underlying ideas are involved, the proofs are simpler to present for the binary case, so we will focus on binary codes. We will state the bound for the q-ary case and leave the details as an exercise. Our proofs reduce the task of bounding the size of the code to bounding the number of pairwise far-apart unit vectors in Euclidean space, and then use a geometric argument for the latter task.

We first state the geometric lemma we need.

Lemma 3 *Let $v_1, v_2, \ldots, v_m$ be m unit vectors in $\mathbb{R}^n$.*

1. *Suppose $\langle v_i, v_j \rangle \leq -\epsilon$ for all $1 \leq i < j \leq m$. Then $m \leq 1 + \frac{1}{\epsilon}$.*
2. *Suppose $\langle v_i, v_j \rangle \leq 0$ for all $1 \leq i < j \leq m$. Then $m \leq 2n$.*

Proof: We only prove the first part, and leave the second as an (interesting) exercise. Note that bound of $2n$ is best possible, as we can take n orthonormal vectors and their negations. For the first part, we have

$$0 \leq \langle \sum_{i=1}^{m} v_i, \sum_{i=1}^{m} v_i \rangle = \sum_{i=1}^{m} \|v_i\|^2 + 2 \sum_{1 \leq i < j \leq m} \langle v_i, v_j \rangle \leq m - m(m-1)\epsilon \ ,$$

which gives $m \leq 1 + 1/\epsilon$. $\square$

Using the above, we can prove that a binary code of block length n and distance $d \geq n/2$ cannot have too many codewords.

Theorem 4 *Let C be a binary code C of block length n and distance d.*

1. *If $d > n/2$, then $|C| \leq \frac{2d}{2d-n}$.*
2. *If $d \geq n/2$, then $|C| \leq 2n$.*

PROOF: Let $m = |C|$ and let $c_1, c_2, \ldots, c_m \in \{0,1\}^n$ be the codewords of C. By hypothesis $\Delta(c_i, c_j) \geq d$ for $1 \leq i < j \leq m$. We will map the codewords into unit vectors $v_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, m$, such that the angle between every pair of vectors is at least 90 degrees (i.e., their dot product $\langle v_i, v_j \rangle < 0$). These vectors are defined as follows:

$$v_i = \frac{1}{\sqrt{n}}\left((-1)^{c_i[1]}, (-1)^{c_i[2]}, \cdots, (-1)^{c_i[n]}\right) ,$$

where $c_i[\ell]$ is the ℓ'th bit of the codeword c_i. It is easy to check that

$$\langle v_i, v_j \rangle = \frac{1}{n}(n - 2\Delta(c_i, c_j)) \leq \frac{n-2d}{n} .$$

When $d \geq n/2$, these dot products are non-positive, and by the second part of Lemma 3, we can bound $m \leq 2n$.

For the first part, if $2d > n$, then $\langle v_i, v_j \rangle \leq -\frac{2d-n}{n} < 0$, and therefore by the first part of Lemma 3, we can bound

$$m \leq 1 + \frac{n}{2d-n} = \frac{2d}{2d-n} .$$

□

The above shows that a code family of relative distance $\delta \geq 1/2 + \gamma$ can have at most $O(1/\gamma)$ codewords. Thus a code family cannot have relative distance strictly bounded away from $1/2$ with a number of codewords that is growing with the block length. In particular, such code families have zero rate. We now prove that this is also the case if the relative distance is $1/2$.

We now use a "puncturing" argument, which implies that the complement of the feasible rate vs relative distance region is convex, to derive an upper bound of rate for relative distances $\delta < 1/2$.

Theorem 5 *If a binary code C has block length n and distance $d < n/2$, then $|C| \leq d \cdot 2^{n-2d+2}$.*

PROOF: Let $\ell = n - 2d + 1$ and $S = \{1, 2, \ldots, \ell\}$. For each $a \in \{0,1\}^\ell$, define the subcode C_a to be the subcode of C consisting of all codewords which have a in the first ℓ positions, projected on $S^c = \{1, 2, \ldots, n\} \setminus S$. Formally, $C_a = \{c_{|S^c} \mid c_i = a_i \text{ for } 1 \leq i \leq \ell\}$. Each C_a is a binary code of block length $n - \ell = 2d - 1$. Note that since C has distance at least d, so does C_a. By Theorem 4, $|C_a| \leq 2d$. Of course, $|C| = \sum_{a \in \{0,1\}^\ell} |C_a|$. So we conclude $|C| \leq 2d \cdot 2^\ell = d \cdot 2^{n-2d+2}$. □

We record the asymptotic implication of the above upper bound as:

Corollary 6 *The rate R of a binary code of relative distance δ must satisfy $R \leq 1 - 2\delta + o(1)$.*

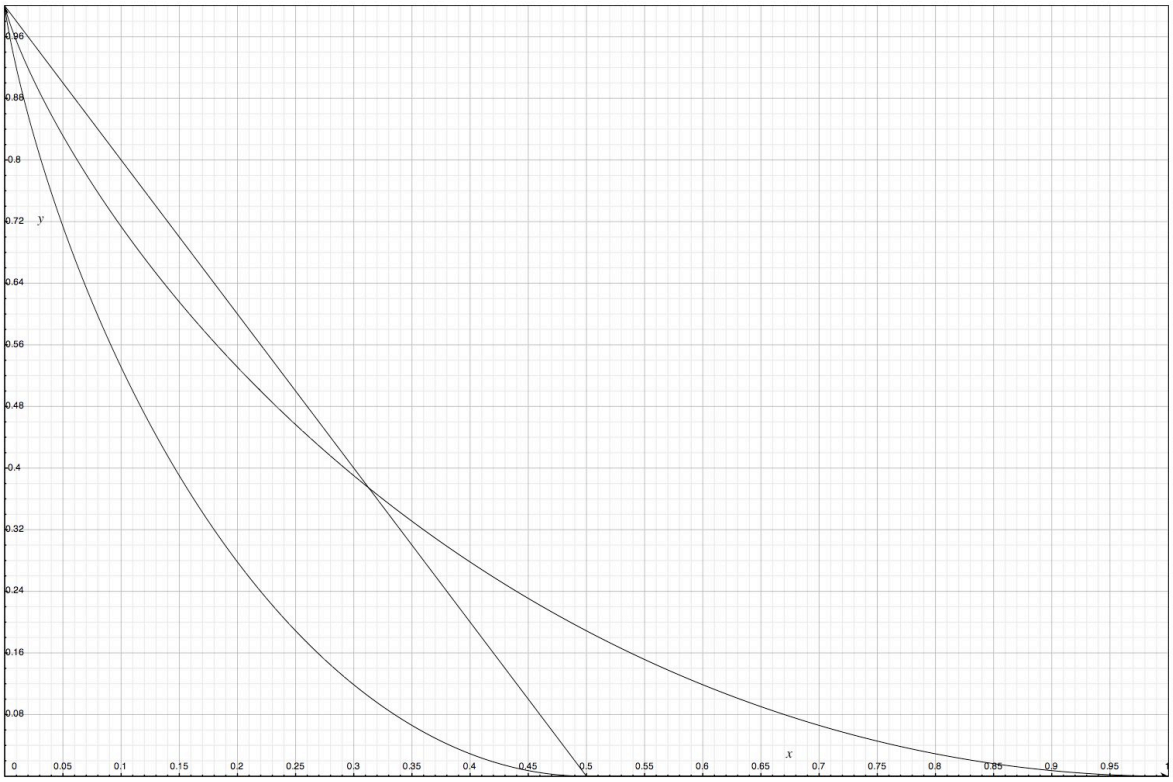
The above arguments can be extended to the q-ary case. The idea is to map q symbols to q unit vectors whose pairwise dot product is exactly $-\frac{1}{q-1}$.

Exercise 1 *Let C be a q-ary code of block length n and minimum distance at least d.*

1. *If $d > (1 - 1/q)n$, then $|C| \leq \frac{qd}{qd-(q-1)n}$.*
2. *When $d < (1 - 1/q)n$, $|C| \leq \frac{q^3 d}{q-1} q^{n-qd/(q-1)}$.*

Deduce that the R of a q-ary code of relative distance δ must satisfy $R \leq 1 - \frac{q}{q-1}\delta + o(1)$.

Here is a plot of the Gilbert-Varshamov lower bound on rate, and the Hamming and Plotkin upper bounds on rate, for binary codes. On the horizontal axis is the relative distance $\delta \in [0, 1]$ and the vertical axis is the rate $R \in [0, 1]$. Any (R, δ) point under the leftmost curve (the Gilbert-Varshamov bound) is achievable, and any (R, δ) point above either the Plotkin bound (the straight line) or the Hamming bound is not achievable. Notice that the Hamming bound is stronger than the Plotkin bound for low distances (or high rates). We now proceed to a bound that improves both the Hamming and Plotkin bounds.



3 Johnson and Elias-Bassalygo bounds

Recall that $A_q(n,d)$ denotes the size of the largest q-ary code of block length n and distance d. We denote by $A_q(n,d,w)$ the size of a largest *constant weight* code of block length n and distance d all of whose codewords have Hamming weight w. We also denote by $A'_q(n,d,w)$ the largest size of a code of block length n and distance d all of whose codewords have Hamming weight *at most* w. For the case $q=2$, we just denote these quantities by $A(n,d)$, $A(n,d,w)$, and $A'(n,d,w)$ respectively.

On your problem set, you are asked to prove the following (just for the binary case).

Exercise 2 *Prove that* $A(n,d) \leq \frac{2^n}{\binom{n}{w}} A(n,d,w)$. *More generally, prove that* $A_q(n,d) \leq \frac{q^n}{\binom{n}{w}(q-1)^w} A_q(n,d,w)$

The above gives a method to upper bound the size $A_q(n,d)$ of unrestricted codes via upper bounds on the size $A_q(n,d,w)$ of constant weight codes. Note that when $w < d/2$, $A_q(n,d,w) = 1$, so the Hamming like upper bound $A_q(n,d) \leq \frac{q^n}{\binom{n}{w}(q-1)^w}$ is a special case of this. In general the larger the w as a function of d for which we can prove a good upper bound on $A(n,d,w)$ (as either a constant or a polynomial function of n), the better the upper bound on $A_q(n,d)$.

We will now prove such an upper bound, in fact for the more general quantity $A'_q(n,d,w)$. Such a bound, called the *Johnson bound*, is intimately connected to *list decoding*. Proving that $A'_q(n,d,w)$ is small, say at most L which is either a constant or bounded by $n^{O(1)}$, implies that for *every* q-ary code C of block length n and distance d, every Hamming ball of radius w contains at most L codewords of C. In other words, if a codeword is transmitted and at most w errors corrupt it, then one can *list decode* a small list of at most L candidate codewords one of which must equal the original codeword. Of course, for $w < d/2$, we have $L = 1$, and the key here is that one can have $w \gg d/2$ and still ensure a small worst-case *list size* L.

Once we prove the Johnson bound, we will deduce our desired bound on $A_q(n,d)$, called the Elias-Bassalygo bound after their inventors, by appealing to Exercise 2. Our proofs of the Johnson bound will be geometric in nature, relying on Lemma 3. We prove the bounds for binary codes, and leave the extension to larger alphabets as exercises.

3.1 Binary codes

Theorem 7 (Binary Johnson bound) *For integers* $1 \leq w \leq d \leq n/2$, *if* $w \leq \frac{1}{2}(n - \sqrt{n(n-2d)})$, *then* $A'(n,d,w) \leq 2n$. *(We will often refer to the quantity* $\frac{1}{2}(n - \sqrt{n(n-2d)})$ *as* $J_2(n,d)$, *the (binary) Johnson radius.)*

Proof: Let $C = \{c_1, \ldots, c_m\} \subseteq \{0,1\}^n$ be a code such that $\Delta(c_i, c_j) \geq d$ for $i \neq j$, and $\mathrm{wt}(c_i) \leq w$ for each $i = 1, 2, \ldots, m$. We will map the codewords c_i into vectors $v_i \in \mathbb{R}^n$ similarly to the proof of the Plotkin bound (Theorem 4), except we won't normalize them to unit vectors here:

$$v_i = \left((-1)^{c_i[1]}, (-1)^{c_i[2]}, \cdots, (-1)^{c_i[n]}\right) ,$$

where $c_i[\ell]$ is the ℓ'th bit of the codeword c_i. Likewise the all 0's vector is mapped to the vector $r \in \mathbb{R}^n$

$$r = (1, 1, \cdots, 1) \ .$$

Let $\alpha > 0$ be a parameter to be picked later. The parameter α will be picked so that all the pairwise dot products $\langle v_i - \alpha r, v_j - \alpha r \rangle$ are nonpositive for $i \neq j$. Now

$$\begin{aligned} \langle v_i - \alpha r, v_j - \alpha r \rangle &= n - 2\Delta(c_i, c_j) + \alpha^2 n + \alpha(2\mathrm{wt}(c_i) - n + 2\mathrm{wt}(c_j) - n) \\ &\leq n - 2d + \alpha^2 n + 2\alpha(2w - n) \ . \end{aligned}$$

The latter quantity is at most 0 provided

$$4w \leq 2n - \left(\alpha n + \frac{n - 2d}{\alpha} \right) \ .$$

The choice $\alpha = \sqrt{n(n-2d)}$ maximizes the right hand side, and leads to the requirement

$$w \leq \frac{n}{2} - \frac{\sqrt{n(n-2d)}}{2}$$

which is met by hypothesis about w. Therefore, for this choice of α, $\langle v_i - \alpha r, v_j - \alpha r \rangle \leq 0$ for $1 \leq i < j \leq m$. are nonpositive for $i \neq j$. Appealing to (the second part of) Lemma 3, we can conclude $m \leq 2n$, and thus $A'(n, d, w) \leq 2n$. □

Remark 8 *It is possible to improve the upper bound on $A'(n, d, w)$ for $w < J_2(n, d)$ from $2n$ to n by noting that for the choice of parameters $\langle v_i - \alpha r, r \rangle > 0$ for each i. This together with the nonpositive pairwise dot products can be used to improve the geometric upper bound on the number of vectors from $2n$ to n.*

For w slightly less than the Johnson radius $J_2(n, d)$, one can sharpen the upper bound on $A'(n, d, w)$ to a constant independent of n.

Exercise 3 *Prove that when $w \leq \frac{n}{2} - \frac{\sqrt{n(n-2d+2d/L)}}{2}$, $A'(n, d, w) \leq L$.*
*(**Hint**: Follow the above proof, and pick parameters so that the first part of Lemma 3 can be used.)*

Together with Exercise 2, we thus have the following upper bound, called the Elias-Bassalygo bound, on the size (rate) of codes of certain distance (relative distance).

Theorem 9 *For integers $1 \leq d \leq n/2$,*

$$A(n, d) \leq \frac{n 2^{n+1}}{\binom{n}{J_2(n,d)}}$$

where $J_2(n, d) = (n - \sqrt{n(n-2d)})/2$.
Thus a binary code family of relative distance δ has rate at most

$$1 - h\Big(\frac{1-\sqrt{1-2\delta}}{2}\Big) + o(1) \ .$$

3.2 Statement for larger alphabets

As with the Plotkin bound, we leave the extension of the Johnson and Elias-Bassalygo bounds to larger alphabets exercises. The hint is to map q symbols into appropriate vectors in $\mathbb{R}^q$ so that large distance between codewords translates into small dot product between their associated vectors.

Exercise 4 *For all integers $q \geq 2$ and $1 \leq d \leq (1-1/q)n$, $A'_q(n,d,w) \leq n(q-1)$ provided*

$$w < J_q(n,d) = n\Big(1-\frac{1}{q}\Big)\left(1-\sqrt{1-\frac{qd}{(q-1)n}}\right) \ .$$

Further, if

$$w \leq n\Big(1-\frac{1}{q}\Big)\left(1-\sqrt{1-\frac{qd(1-\epsilon)}{(q-1)n}}\right)$$

then $A'_q(n,d,w) \leq 1/\epsilon$.

Together with Exercise 2, this gives the Elias-Bassalygo upper bound for q-ary codes:

Theorem 10 *A q-ary code family of relative distance δ has rate at most*

$$1 - h_q\left((1-1/q)\Big(1-\sqrt{1-\frac{q\delta}{q-1}}\Big)\right) + o(1) \ .$$

3.3 Alphabet oblivious bound for Johnson radius

When discussing list decoding of codes such as Reed-Solomon codes which are defined over an alphabet size that grows with the block length, it will be useful to have the following "alphabet oblivious" version of the Johnson bound. (This version is also a simpler and often good enough approximation to the q-ary Johnson radius when q is somewhat large.)

Theorem 11 *Let $C \subseteq \Sigma^n$ be a code of block length n and distance d. Then the following hold:*

1. *Every Hamming ball of radius at most*

$$J(n,d) = n - \sqrt{n(n-d)}$$

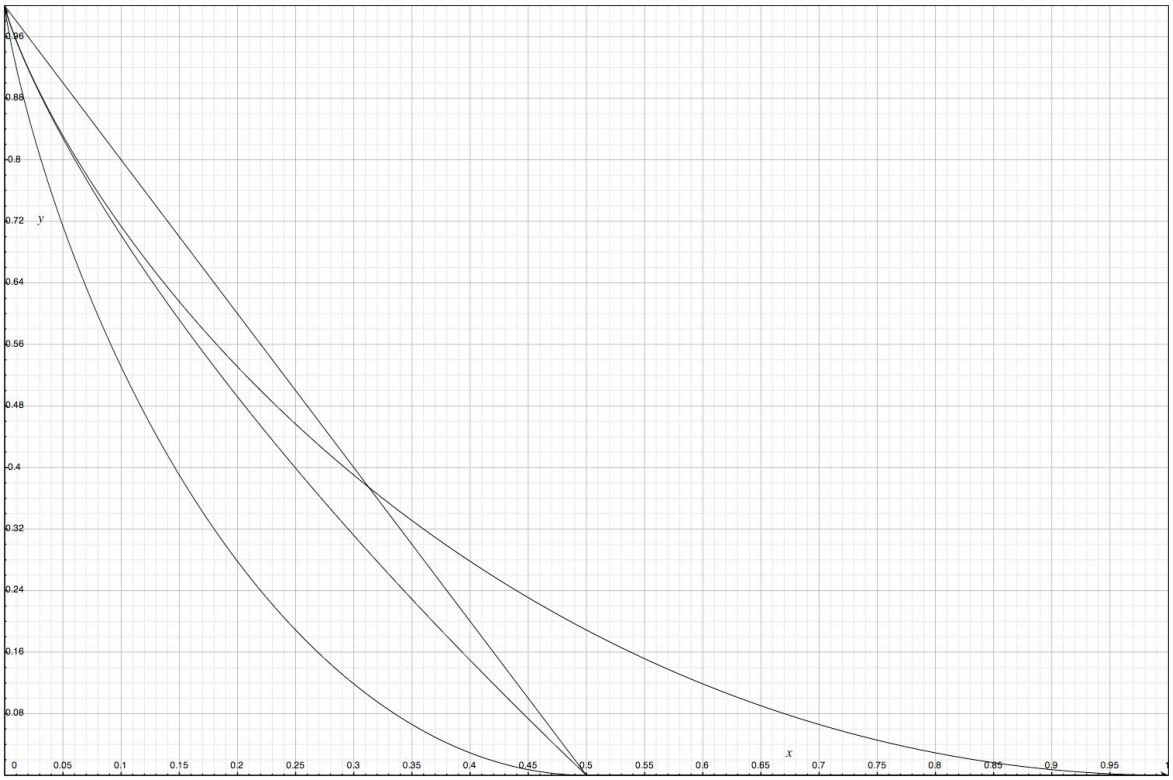
in Σ^n has at most $O(n|\Sigma|)$ codewords of C.

2. *Every Hamming ball of radius at most* $n - \sqrt{n(n-d+d\epsilon)}$ *in* Σ^n *has at most* $1/\epsilon$ *codewords of* C*.*

The above statement follows from Exercise 4 by verifying that for every $q \geq 2$ and $0 \leq x \leq 1 - 1/q$,

$$1 - \sqrt{1-x} \leq (1-1/q)\sqrt{1 - \frac{qx}{q-1}} \ .$$

We conclude with a plot that adds the Elias-Bassalygo upper bound to the earlier plot. Note that this bound improves on both the Hamming and Plotkin bounds, but for small distances the difference between the Hamming the Elias-Bassalygo bounds is small.



Introduction to Coding Theory **CMU: Spring 2010**

Notes 5.1: Fourier Transform, MacWillams identities, and LP bound

February 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkat Guruswami & Srivatsan Narayanan*

We will discuss the last and most sophisticated of our (upper) bounds on rate of codes with certain relative distance, namely the first linear programming bound or the first JPL bound due to McEliece, Rodemich, Rumsey, and Welch, 1977 (henceforth, MRRW). This bound is the best known asymptotic upper bound on the rate of a binary code for a significant range of relative distances (which is roughly $\delta \in (0.273, 1/2)$). We will present a complete and self-contained proof of the this bound. A variant called the second JPL bound gives the best known upper bound for the remainder of the range, and we will mention this bound (without proof) at the end.

The linear programming bound is so-called because it is based on Delsarte's linear programming approach which shows that the distance distribution of a binary code satisfies a family of linear constraints whose coefficients are the evaluations of a certain family of orthogonal polynomials (in this case, the Krawtchouk polynomials). The optimum (maximum) of this linear program gives an upper bound on $A(n, d)$. MRRW constructed good feasible solutions to the dual of linear program using tools from the theory of orthogonal polynomials, and their value gave an upper bound on $A(n, d)$ by weak duality.

In these notes, we will use Fourier analysis of functions defined on the hypercube to derive a relationship between the weight distribution of a linear code and its dual, called the MacWilliams identifies. These give the linear constraints of the above-mentioned linear program.

Instead of the using the linear program or its dual and the theory of orthogonal polynomials (and specifically properties of Krawtchouk polynomials), in the second part of these notes, we will give a *self-contained proof* of the first linear programming bound for binary linear codes using a Fourier analytic approach. This is based on the methods of Friedman and Tillich, which was later extended also to general codes by Navon and Samorodnitsky, that shows that the dual of a linear code of large distance must have small "essential covering radius" (which means that Hamming balls of small radii around the dual codewords will cover a large fraction of the Hamming space $\{0,1\}^n$). This shows that the dual must have large size, and therefore the code itself cannot be too large. The method can be extended to non-linear codes, but we will be content with deriving the linear programming bound for (binary) linear codes.

1 Fourier analysis over the Boolean hypercube

Let $\mathcal{F}_n$ be set of all real-valued functions over the boolean hypercube, *i.e.*, $\mathcal{F}_n = \{f : \{0,1\}^n \to \mathbb{R}\}$. Then the following characterization is straightforward.

Exercise 1 *Show that $\mathcal{F}_n$ forms a vector space with dimension 2^n. In fact, show that $\{e_\alpha : \alpha \in$*

$\{0,1\}^n\}$ *forms a basis for* $\mathcal{F}_n$*, where* $e_\alpha : \{0,1\}^n \to \mathbb{R}$ *is defined by:*

$$e_\alpha(x) = \delta_{x\alpha} = \begin{cases} 1, & \text{if} x = \alpha \\ 0, & \text{otherwise} \end{cases}$$

In fact, the above exercise views a function $f \in \mathcal{F}_n$ as simply a vector of dimension 2^n, indexed by the "coordinates" $\alpha \in \{0,1\}^n$. This motivates us to define an inner product on $\mathcal{F}_n$:

Definition 1 *For* $f, g \in \mathcal{F}_n$*, define the inner product between* f *and* g *to be:*

$$\langle f, g \rangle = \frac{1}{2^n} \sum_x f(x)g(x) = \mathbb{E}_x \left[f(x)g(x) \right]$$

(This is just the standard inner product for vectors over reals, but suitably normalized.)

Now, we will define another basis for $\mathcal{F}_n$, called the *Fourier basis*. This needs the following simple lemma.

Lemma 2 *For every binary linear code* $C \subseteq \{0,1\}^n$*,*

$$\sum_{c \in C} (-1)^{\alpha \cdot c} = \begin{cases} |C|, & \text{if } \alpha \in C^\perp, \\ 0, & \text{otherwise.} \end{cases}$$

where $\cdot$ *denotes the dot product modulo* 2*.*

Proof: If $\alpha \in C^\perp$, then the claim is obvious. Suppose that $\alpha \notin C^\perp$. Then, there exists a $c_0 \in C$ such that $\alpha \cdot c_0 = 1$. Now, for each $c \in C$,

$$(-1)^{\alpha \cdot c} + (-1)^{\alpha \cdot (c + c_0)} = (-1)^{\alpha \cdot c} \left(1 + (-1)^{\alpha \cdot c_0}\right) = 0 \tag{1}$$

Summing Equation 1 for all $c \in C$, we get:

$$0 = \sum_{c \in C} \left((-1)^{\alpha \cdot c} + (-1)^{\alpha \cdot (c + c_0)} \right) = \sum_{c \in C} (-1)^{\alpha \cdot c} + \sum_{c \in C} (-1)^{\alpha \cdot (c + c_0)} = 2 \sum_{c \in C} (-1)^{\alpha \cdot c},$$

giving the claim. □

Corollary 3 *We have*

$$\sum_{c \in \{0,1\}^n} (-1)^{\alpha \cdot c} = \begin{cases} 2^n, & \text{if } \alpha = 0, \\ 0, & \text{otherwise.} \end{cases}$$

Proof: In Lemma 2, take C to be the whole vector space $\{0,1\}^n$, so that $C^\perp = \{0\}$. □

Remark 4 *In this lecture, the notation* 0 *is typically overloaded to mean either a single alphabet symbol, or the zero vector* (0^n) *of the vector space. However, the right definition should be clear from the context.*

For each $\alpha \in \{0,1\}^n$, define $\chi_\alpha : \{0,1\}^n \to \mathbb{R}$ by $\chi_\alpha(x) = (-1)^{\alpha \cdot x}$ (where $\cdot$ refers to the inner product between vectors, taken modulo 2). The function χ_α is often called a *character function*. We show that the set of all character functions also forms an orthonormal basis for $\mathcal{F}_n$.

Lemma 5 $\langle \chi_\alpha, \chi_\beta \rangle = \delta_{\alpha\beta}$

Proof:

$$\langle \chi_\alpha, \chi_\beta \rangle = \mathbb{E}_x \left[(-1)^{\alpha \cdot x} (-1)^{\beta \cdot x} \right] = \frac{1}{2^n} \sum_x (-1)^{(\alpha - \beta) \cdot x} = \begin{cases} 1, & \text{if } \alpha - \beta = 0, \\ 0, & \text{otherwise} \end{cases}$$

using Corollary 3. The claim follows from the definition of $\delta_{\alpha\beta}$. □

Corollary 6 *Let* B *be the set of character functions, i.e.,* $B = \{\chi_\alpha : \alpha \in \{0,1\}^n\}$. *Then,* B *is an orthonormal basis for* $\mathcal{F}_n$, *called its Fourier basis.*

Proof: From Lemma 5, it follows that B is a linearly independent set. Also the cardinality of B is 2^n, which equals the dimension of the whole space $\mathcal{F}_n$. Therefore, B must be a basis. The orthonormality of B is directly implied again by Lemma 5. □

By the definition of a basis, any function $f \in \mathcal{F}_n$ can be expressed uniquely as a linear combination of the character functions. That is, there exist $\hat{f}(\alpha) \in \mathbb{R}$ such that

$$f = \sum_\alpha \hat{f}(\alpha) \chi_\alpha \ .$$

(We use the notation $\hat{f}(\alpha)$, instead of the conventional c_α to remind us that the coefficients depend on f.) Note that this is equivalent to saying

$$f(x) = \sum_\alpha \hat{f}(\alpha) \chi_\alpha(x)$$

for all $x \in \{0,1\}^n$.

The following are some immediate consequences of this fact.

Lemma 7 *Let* $f, g \in \mathcal{F}_n$. *Then the following hold.*

1. $\langle f, \chi_\alpha \rangle = \hat{f}(\alpha)$
2. *(Parseval's identity)* $\langle f, g \rangle = \sum_\alpha \hat{f}(\alpha) \hat{g}(\alpha)$
3. $\hat{f}(0) = \mathbb{E}_x f(x)$

PROOF: Each of the above claims can be shown by a straightforward calculation.

1. $\langle f, \chi_\alpha \rangle = \langle \sum_\beta \hat{f}(\beta)\chi_\beta, \chi_\alpha \rangle = \sum_\beta \hat{f}(\beta)\langle \chi_\beta, \chi_\alpha \rangle = \sum_\beta \hat{f}(\beta)\delta_{\alpha\beta} = \hat{f}(\alpha)$
2. $\langle f, g \rangle = \langle f, \sum_\alpha \hat{g}(\alpha)\chi_\alpha \rangle = \sum_\alpha \hat{g}(\alpha)\langle f, \chi_\alpha \rangle = \sum_\alpha \hat{f}(\alpha)\hat{g}(\alpha)$
3. $\hat{f}(0) = \langle f, \chi_0 \rangle = \mathbb{E}_x \left[f(x)(-1)^{0 \cdot x} \right] = \mathbb{E}_x f(x)$

□

2 Dual codes, Fourier analysis, and MacWilliams identities

Let us introduce the following notation: for any $S \subseteq \{0,1\}^n$, define $1_S : \{0,1\}^n \to \mathbb{R}$, called the *characteristic function of* S, by

$$1_S(x) = \begin{cases} 1, & \text{if } x \in S, \\ 0, & \text{otherwise.} \end{cases}$$

We will now show that Fourier transform of the characteristic function of a code is *essentially the same* (up to a constant scaling factor) as the characteristic function of its dual. This is useful because the Fourier transform can be viewed as a notion of duality for functions. Fortunately, there is a natural correspondence between the two notions (dual codes and Fourier transforms).

Lemma 8 *For any linear code* $C \subseteq \{0,1\}^n$,

$$\widehat{1_C} = \frac{|C|}{2^n} 1_{C^\perp}$$

PROOF: For every $\alpha \in \{0,1\}^n$,

$$\widehat{1_C}(\alpha) = \langle 1_C, \chi_\alpha \rangle = \frac{1}{2^n} \sum_x 1_C(x)\chi_\alpha(x) = \frac{1}{2^n} \sum_{x \in C} (-1)^{\alpha \cdot x} = \begin{cases} \frac{1}{2^n}|C|, & \text{if } \alpha \in C^\perp, \\ 0, & \text{otherwise} \end{cases}$$

using Lemma 2. Therefore,

$$\widehat{1_C}(\alpha) = \frac{|C|}{2^n} 1_{C^\perp}(\alpha),$$

for all $\alpha \in \{0,1\}^n$, giving the claim. □

Definition 9 *For any* $S \subseteq \{0,1\}^n$, *let*

$$W_i^S = \#\{x \in S : \mathrm{wt}(x) = i\},$$

that is, W_i^S *denotes the number of points in* S *of weight* i. *Further, by weight distribution of* S, *we denote the* $(n+1)$*-tuple* $W^S = \langle W_0^S, W_1^S, \ldots, W_n^S \rangle$.

Now, our goal is to relate the "weight distribution" of a code C to that of its dual $C^\perp$. Let $\ell \in \{0, 1, \ldots, n\}$. Then,

$$\begin{aligned}
W_\ell^{C^\perp} &= \sum_{\alpha:\mathrm{wt}(\alpha)=\ell} 1_{C^\perp}(\alpha) \\
&= \frac{2^n}{|C|} \sum_{\alpha:\mathrm{wt}(\alpha)=\ell} \widehat{1_C}(\alpha) \\
&= \frac{2^n}{|C|} \sum_{\alpha:\mathrm{wt}(\alpha)=\ell} \mathbb{E}_x \left[1_C(x)(-1)^{\alpha\cdot x}\right] \\
&= \frac{2^n}{|C|} \mathbb{E}_x \left[\sum_{\alpha:\mathrm{wt}(\alpha)=\ell} 1_C(x)(-1)^{\alpha\cdot x}\right] \\
&= \frac{2^n}{|C|} \mathbb{E}_x \left[1_C(x) \sum_{\alpha:\mathrm{wt}(\alpha)=\ell} (-1)^{\alpha\cdot x}\right]
\end{aligned}$$

For completeness, we calculate the sum $\sum_{\alpha:\mathrm{wt}(\alpha)=\ell}(-1)^{\alpha\cdot x}$ in the following lemma. The exact sum is not of any significance for our purposes in this course. We will however use the fact that this sum depends *only* on the weight of x.

Lemma 10 *For any $x \in \{0,1\}^n$ with $\mathrm{wt}(x) = i$,*

$$\sum_{\alpha:\mathrm{wt}(\alpha)=\ell} (-1)^{\alpha\cdot x} = \sum_{j=0}^{\ell} (-1)^j \binom{i}{j}\binom{n-i}{\ell-j} .$$

The latter quantity will be denoted as $K_\ell(i)$ — the value of the Krawtchouk polynomial at i.

PROOF: Notice that summation is taken over all α of a given weight ℓ. So, by symmetry, it depends only the number of 1's in x, and not on their positions. Hence, without any loss in generality, assume that $x = 1^i 0^{n-i}$. A vector α of weight ℓ must have j 1's in the first i positions, and $\ell - j$ in the last $n-i$ positions, for some $j \in \{0, 1, \ldots, \ell\}$, and in this case $(-1)^{x\cdot\alpha} = (-1)^j$. The number of α's satisfying this condition for any particular $j \in \{0, 1, \ldots, \ell\}$ equals $\binom{i}{j}\binom{n-i}{\ell-j}$. The claim thus follows. □

Remark 11 (Krawtchouk polynomial) *The quantity $\sum_{j=0}^{\ell}(-1)^j\binom{i}{j}\binom{n-i}{\ell-j}$, denoted $K_\ell(i)$, can be regarded as the evaluation of a polynomial K_ℓ at $\mathrm{wt}(x) = i$. K_ℓ is usually called the ℓ^{th} Krawtchouk polynomial and is defined as*

$$K_\ell(X) = \sum_{j=0}^{\ell} (-1)^j \binom{X}{j}\binom{n-X}{\ell-j} .$$

(The function K_ℓ also depends on n, but we supress this dependence for notational convenience.) Note that K_ℓ is a polynomial of degree ℓ and $K_0(X) = 1$ and $K_1(X) = n - 2X$, etc.

Now, we will complete the calculation of $W_\ell^{C^\perp}$.

$$\begin{aligned} W_\ell^{C^\perp} &= \frac{2^n}{|C|}\frac{1}{2^n}\sum_x \left[1_C(x)\sum_{\alpha:\mathrm{wt}(\alpha)=\ell}(-1)^{\alpha\cdot x}\right] \\ &= \frac{1}{|C|}\sum_x 1_C(x)K_\ell(\mathrm{wt}(x)) \\ &= \frac{1}{|C|}\sum_{x\in C} K_\ell(\mathrm{wt}(x)) \\ &= \frac{1}{|C|}\sum_{i=0}^{n}\sum_{x\in C,\mathrm{wt}(x)=i} K_\ell(i) \end{aligned}$$

giving

$$W_\ell^{C^\perp} = \frac{1}{|C|}\sum_{i=0}^{n} W_i^C K_\ell(i) \tag{2}$$

for every $\ell = 0, 1, 2, \ldots, n$.

Equation 2, called the MacWilliams identity, tells us that the weight distribution of the dual code $C^\perp$ is completely determined once we are given the weight distribution of the code C.

Remark 12 *We can write the MacWilliams identitities (2) equivalently as:*

$$W_\ell^{C^\perp} = \mathbb{E}_{x\in C}\left[K_\ell(\mathrm{wt}(x))\right] \ ,$$

or as a functional equation

$$\sum_{\ell=0}^{n} W_\ell^{C^\perp} z^\ell = \frac{1}{|C|}\sum_{i=0}^{n} W_i^C (1-z)^i (1+z)^{n-i} \ .$$

Exercise 2 *Extend the MacWilliams identities to linear codes over any finite field $\mathbb{F}_q$. Specifically, if C is a q-ary linear code of block length n, and as before W_i^C (resp. $W_i^{C^\perp}$) denote the number of codewords of C (resp. $C^\perp$) of Hamming weight i, then*

$$W_\ell^{C^\perp} = \frac{1}{|C|}\sum_{i=0}^{n} W_i^C K_\ell^{(q)}(i)$$

where the q-ary Krawtchouk polynomial is defined as

$$K_\ell^{(q)}(X) = \sum_{j=0}^{\ell}(-1)^j (q-1)^{\ell-j}\binom{X}{j}\binom{n-X}{\ell-j} \ .$$

[[**Hint***: When the field size q equals a prime p, replace $(-1)^{x\cdot y}$ in the proof for the binary case by $\zeta_p^{x\cdot y}$ where $\zeta_p = e^{2\pi i/p}$ is a primitive p'th root of unity and $x\cdot y$ is, as usual, computed over the underlying field $\mathbb{F}_q$.*

When $q = p^t$ for a prime p, the role of $(-1)^{x \cdot y}$ can be played by $\zeta_p^{\mathrm{Tr}(x \cdot y)}$ where Tr *is the trace map from $\mathbb{F}_q$ to $\mathbb{F}_p = \{0, 1, 2 \ldots, p-1\}$: $\mathrm{Tr}(z) = z + z^p + \cdots + z^{p^{t-1}}$.]]*

Exercise 3 *Using the above, compute the weight distribution of the $[q^m - 1, q^m - 1 - m, 3]_q$ Hamming code.*

3 A linear program bounding $A(n, d)$

In this section, we will use the MacWilliams identity to derive a linear program that bounds the size of every code with a given minimum distance d, and thus bounds $A(n, d)$. (Recall that $A(n, d)$ is the maximum size of any binary code with block length n and minimum distance d.)

For the moment, we will focus on linear codes C. Consider the linear program:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{i=0}^{n} A_i & & & \\
\text{s.t.} \quad & A_0 & = & 1 & \\
& A_i & \geq & 0, & \quad i = 1, \ldots, n \\
& A_i & = & 0, & \quad i = 1, \ldots, d-1 \\
& \sum_{i=0}^{n} K_\ell(i) A_i & \geq & 0, & \quad \ell = 1, \ldots, n
\end{aligned}$$

We claim that for any *linear* code C of distance at least d, the assignment $A_i = W_i^C$ is a feasible solution. Indeed, the first two constraints are satisfied trivially. The constraint $A_i = 0$ for $1 \leq i < d$ enforces that the minimum distance of the code (that is, the minimum Hamming weight of any nonzero code word) is at least d. The last set of constraints follow from the MacWilliams identities for any $\ell \in \{1, 2, \ldots, n\}$,

$$\sum_{i=0}^{n} W_i^C K_\ell(i) = W_\ell^{C^\perp} \geq 0$$

For this assignment, the objective function takes the value

$$\sum_{i=0}^{n} W_i^C = |C|$$

Therefore, the optimum of the linear program upper bounds the size of any linear code C of distance at least d.

Now, we consider general codes C, and prove that they satisfy the same bound. Without loss of generality, assume that $0^n \in C$. Define:

$$A_i^C = \frac{\#\{(x, y) \in C^2 \mid \Delta(x, y) = i\}}{|C|}$$

We claim that A_i^C is a feasible solution to the linear program. The first three sets of constraints are trivially satisfied as before, whereas the last set of constraints can be verified in a straightforward manner:

$$\begin{aligned}
\sum_{i=0}^{n} A_i^C K_\ell(i) &= \frac{1}{|C|}\sum_{i=0}^{n}\sum_{(x,y)\in C^2:\Delta(x,y)=i} K_\ell(i) \\
&= \frac{1}{|C|}\sum_{i=0}^{n}\left(\sum_{(x,y)\in C^2:\Delta(x,y)=i}\left(\sum_{z:\mathrm{wt}(z)=\ell}(-1)^{(x-y)\cdot z}\right)\right) \\
&= \frac{1}{|C|}\sum_{(x,y)\in C^2}\left(\sum_{z:\mathrm{wt}(z)=\ell}(-1)^{(x-y)\cdot z}\right) \\
&= \frac{1}{|C|}\sum_{z:\mathrm{wt}(z)=\ell}\left(\sum_{(x,y)\in C^2}(-1)^{x\cdot z}(-1)^{y\cdot z}\right) \\
&= \frac{1}{|C|}\sum_{z:\mathrm{wt}(z)=\ell}\left(\sum_{x\in C}(-1)^{x\cdot z}\right)\left(\sum_{y\in C}(-1)^{y\cdot z}\right) \\
&= \frac{1}{|C|}\sum_{z:\mathrm{wt}(z)=\ell}\left(\sum_{x\in C}(-1)^{x\cdot z}\right)^2 \\
&\geq 0
\end{aligned}$$

The value of the objective function is:

$$\sum_{i=0}^{n} A_i^C = \frac{1}{|C|}\sum_{i=0}^{n}\left(\sum_{(x,y)\in C^2:\Delta(x,y)=i} 1\right) = \frac{1}{|C|}\sum_{(x,y)\in C^2} 1 = |C|$$

Therefore, the optimum value of the linear program upper bounds the size of any code with minimum distance at least d.

3.1 Dual program and the MRRW bound

Consider the dual program for the above linear program. The dual program has variables $\beta_1, \beta_2, \ldots, \beta_n$ (where $\beta_i \geq 0$). Define $\beta(X)$ to be the polynomial

$$\beta(X) = 1 + \sum_{\ell=0}^{n} \beta_\ell K_\ell(X) \ .$$

Then the dual program is given by:

$$\begin{aligned}
\text{Minimize} \quad & \beta(0) \\
\text{s.t.} \quad \beta_i &\geq 0, \qquad i = 1, 2, \ldots, n \\
\beta(j) &\leq 0, \qquad j = d, \ldots, n
\end{aligned}$$

By the weak duality theorem, the value of *any* feasible solution to the dual program upper bounds the optimum value of the linear program, and hence also upper bounds $A(n, d)$. Hence, in order to upper bound the size of the code, it suffices to exhibit a dual feasible solution with a small objective function. This was, in fact, the approach followed by MRRW, leading to the first linear programming bound. However, this involves studying several properties of Krawtchouk polynomials. In the second installment of these notes, we will prove the same bound by following a different approach based on Fourier analysis.

Introduction to Coding Theory **CMU: Spring 2010**

Notes 5.2: Proof of the first MRRW bound

February 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkat Guruswami & Srivatsan Narayanan*

1 Fourier analytic proof of the first MRRW bound

We develop a proof of the first MRRW (JPL) bound for binary *linear* codes based on a covering argument. Our main theorem (Theorem 1) states that the dual code $C^\perp$ has a small essential covering radius. From this, we conclude that the size of the dual code $|C^\perp|$ is large, and equivalently, the size of the code C is small.

Theorem 1 (Dual codes have a small "essential covering radius") *Let C be a binary linear code of distance at least d. Then,*

$$\left| \bigcup_{z \in C^\perp} B(z, r) \right| \geq \frac{2^n}{n} \tag{1}$$

for $r = \frac{1}{2}n - \sqrt{d(n-d)} + o(n)$.

Remark 2 *We say that a set S has a covering radius at most r if every point in $\{0,1\}^n$ is inside $B(z,r)$ for some $z \in S$. Theorem 1 asserts that the dual code satisfies a relaxed version of the property: for large enough r, at least a $n^{-O(1)}$ fraction of the points are inside $B(z,r)$ for some $z \in C^\perp$. This is sufficient for establishing an asymptotic bound on the rate of the codes, as shown in Corollary 3.*

Before we prove Theorem 1, we observe that it directly implies an asymptotic upper bound on the rate of codes.

Corollary 3 *Let C be a binary linear code with distance at least $d = \delta n$. Then,*

1. $|C| \leq n\mathrm{Vol}(n, r)$
2. $R(C) \leq h\left(\frac{1}{2} - \sqrt{\delta(1-\delta)}\right) + o(1)$

Proof: The covering condition (Equation 1) gives us that:

$$|C^\perp| \cdot \mathrm{Vol}(n, r) \geq \left| \bigcup_{z \in C^\perp} B(z, r) \right| \geq \frac{2^n}{n}$$

since the volume of each ball $B(z,r)$ is exactly $\mathrm{Vol}(n,d)$. But, the sizes of C and $C^\perp$ are related as follows (see Exercise 1): $|C^\perp| \cdot |C| = 2^n$. Combining the two observations, we directly obtain a bound on the code size:

$$|C| = \frac{2^n}{|C^\perp|} \leq n\mathrm{Vol}(n,r)$$

To obtain the bound on the rate, we need to translate the above bounds to asymptotic notation. Writing $d = \delta n$,

$$r = n\left(\frac{1}{2} - \sqrt{\delta(1-\delta)} + o(1)\right)$$

so that,

$$|C| \leq n\mathrm{Vol}(n,r) = n2^{nh\left(\frac{1}{2} - \sqrt{\delta(1-\delta)} + o(1)\right)}$$

Taking logarithms, and noting that $(\log n)/n = o(1)$ gives the claimed bound.

Exercise 1 *Show that* $|C| \cdot |C^\perp| = 2^n$.

Hint: What is the relation between the dimensions of a code and its dual? How many points lie on a subspace of dimension k? □

We now need to establish Theorem 1. But, at first we introduce some notation. Let A denote the adjacency matrix of the boolean hypercube; that is, for $x, y \in \{0,1\}^n$, $A_{xy} = 1$ if and only if x and y differ in exactly one coordinate. We now extend the concept of maximum eigenvalue to a set $B \subseteq \{0,1\}^n$.

Definition 4 *For* $B \subseteq \{0,1\}^n$, *define its maximum eigenvalue to be*

$$\lambda_B = \max\left\{ \frac{\langle Af, f\rangle}{\langle f, f\rangle} \;\middle|\; f : \{0,1\}^n \to \mathbb{R}, \mathrm{Supp}(f) \subseteq B \right\}$$

Proposition 5 *The maximum eigenvalue of the whole hypercube is* $\lambda_{\{0,1\}^n} = n$.

In order to establish Theorem 1, we observe that $B(z,r) = z + B(0,r)$; that is, $B(z,r)$ is really a translate of $B(0,r)$ by z. Now, we break the proof down into two parts. First, we show that for *any* $B \subseteq \{0,1\}^n$ with sufficiently large maximum eigenvalue, $\bigcup_{z \in C^\perp}(z+B)$ covers a significant (*i.e.,* $n^{-O(1)}$) fraction of the whole space. We then show that $B(0,r)$ achieves the required maximum eigenvalue, even when r is not too large.

Theorem 6 *Let* C *be a binary linear code with block length* n *and distance* d. *Suppose* $B \subseteq \{0,1\}^n$ *has maximum eigenvalue* $\lambda_B \geq n - 2d + 1$. *Then,*

$$\left| \bigcup_{z \in C^\perp} (z + B) \right| \geq \frac{2^n}{n}$$

Theorem 7 (Hamming balls have a large maximum eigenvalue)

$$\lambda_{B(0,r)} \geq 2\sqrt{r(n-r)} - o(n)$$

Now, our main theorem follows as a corollary of Theorems 6 and 7.

PROOF:(**For Theorem 1**) We just need to pick an r large enough such that

$$\lambda_{B(0,r)} \geq n - 2d + 1$$

This is satisfied provided:

$$2\sqrt{r(n-r)} - o(n) \geq n - 2d + 1$$

Neglecting $o(n)$ terms, we get:

$$2\sqrt{r(n-r)} \geq n - 2d$$

which is true for:

$$r = \frac{1}{2}n - \sqrt{d(n-d)} + o(n)$$

□

Remark 8 *We cannot hope to improve the above bound simply by employing a "better" choice for B. Hamming balls are so-called "Faber-Krahn" minimizers for the Hamming cube. That is, of all sets B with a given volume, Hamming balls have almost optimum value of the maximum eigenvalue: if $|B| = \mathrm{Vol}(n,r)$, then $\lambda_B \leq (1+o(1))\lambda_{B(0,r)}$.*

2 Proof of Theorem 5

We recall the following facts on Fourier transforms from the first installment of these notes. For $f : \{0,1\}^n \to \mathbb{R}$,

$$\begin{aligned} \hat{f}(\alpha) &= \frac{1}{2^n}\sum_x f(x)(-1)^{\alpha\cdot x} \\ f(x) &= \sum_\alpha \hat{f}(\alpha)(-1)^{\alpha\cdot x} \end{aligned}$$

The Parseval identity says that for $f, g : \{0,1\}^n \to \mathbb{R}$,

$$\mathbb{E}_x\left[f(x)g(x)\right] = \sum_\alpha \hat{f}(\alpha)\hat{g}(\alpha)\ .$$

Also, we have a natural correspondence between the Fourier transform of a code and its dual:

$$\widehat{1_C}(\alpha) = \frac{|C|}{2^n}1_{C^\perp}(\alpha) \tag{2}$$

$$\widehat{1_{C^\perp}}(\alpha) = \frac{|C^\perp|}{2^n}1_C(\alpha)$$

The following lemma is a direct consequence of Equation 2.

Lemma 9 *Let C be a binary linear code with distance at least d. Suppose that $\alpha \in \{0,1\}^n$ such that $\alpha \neq 0$ and $\mathrm{wt}(\alpha) < d$. Then,*

$$\widehat{1_{C^\perp}}(\alpha) = 0.$$

We are now equipped to prove the required claim. Let $f_B : \{0,1\}^n \to \mathbb{R}$ be a function with $\mathrm{Supp}(f_B) \subseteq B$, such that

$$\langle Af_B, f_B \rangle = \lambda_B \langle f_B, f_B \rangle$$

(In other words, f_B is a function that maximizes $\langle Af, f\rangle / \langle f, f\rangle$.) Since the set B is understood from the context, for notational ease, we simply set $\lambda = \lambda_B$ and $f = f_B$.

The following exercises make many simplifying observations.

Exercise 2 *For $g : \{0,1\}^n \to \mathbb{R}$, show that $Ag(x) = \sum_{i=1}^n g(x + e_i)$.*

Exercise 3 *Prove the following statements.*

1. *f is nonnegative: for every x, $f(x) \geq 0$.*
2. *For all $x \in B$, we have $Af(x) = \lambda f(x)$.*
3. *For all x, we have $Af(x) \geq \lambda f(x)$. Equivalently, $Af \geq \lambda f$.*

Hint for part (3): *For $x \in B$, equality holds. On the other hand, for $x \notin B$, $f(x) = 0$, and hence the inequality is trivially satisfied. (Note that all the terms in the left hand side are nonnegative.)*

For $z \in \{0,1\}^n$, define $f_z : \{0,1\}^n \to \mathbb{R}$ by $f_z(x) = f(z + x)$. Finally, define $F_B : \{0,1\}^n \to \mathbb{R}$ by:

$$F_B = \frac{1}{2^n} \sum_{z \in C^\perp} f_z$$

Again, for convenience, we set F to be F_B. Note that $\mathrm{Supp}(F) \subseteq \bigcup_{z \in C^\perp} (z + B)$.

We will sketch the high-level idea of the proof. In order to lower bound the $\bigcup_{z \in C^\perp} (z + B)$, we show that the F must have a large support. We establish this by showing that $\mathbb{E}[F]^2$ is comparable to $\mathbb{E}\left[F^2\right]$.

For this we will focus on the quantity $\langle AF, F\rangle$ and establish lower and upper bounds on it.

Lemma 10 (Lower bound on $\langle AF, F\rangle$)

$$\langle AF, F\rangle \geq \lambda \mathbb{E}\left[F^2\right]$$

Lemma 11 (Upper bound on $\langle AF, F\rangle$)

$$\langle AF, F\rangle \leq n\mathbb{E}[F]^2 + (n - 2d)\mathbb{E}\left[F^2\right]$$

Before proving the above lemmata, we will show how to obtain Theorem 6 using them.

Corollary 12 (F has a large support)

$$\left|\bigcup_{z\in C^\perp}(z+B)\right| \geq \frac{\lambda-(n-2d)}{n}2^n$$

PROOF: The lower and upper bounds (Lemmata 10 and 11) together imply

$$\lambda\mathbb{E}\left[F^2\right] \leq n\mathbb{E}\left[F\right]^2 + (n-2d)\mathbb{E}\left[F^2\right],$$

giving

$$\mathbb{E}\left[F\right]^2 \geq \frac{\lambda-(n-2d)}{n}\mathbb{E}\left[F^2\right] \tag{3}$$

On the other hand, note that F is supported on $S = \bigcup_{z\in C^\perp}(z+B)$. Therefore,

$$\mathbb{E}\left[F\right]^2 = \left(\frac{1}{2^n}\sum_{x\in S}F(x)\right)^2 = \langle 1_S, F\rangle^2 \leq \mathbb{E}\left[1_S^2\right]\mathbb{E}\left[F^2\right] = \frac{|S|}{2^n}\mathbb{E}\left[F^2\right], \tag{4}$$

using the Cauchy-Schwartz inequality.

Finally, Equations (3) and (4) together give the required bound:

$$|S| \geq \frac{\lambda-(n-2d)}{n}2^n$$

□

For $\lambda \geq n-2d+1$ (as assumed in Theorem 6), this bound simplifies to

$$\left|\bigcup_{z\in C^\perp}(z+B)\right| \geq \frac{2^n}{n}$$

2.1 Lower bound on $\langle AF, F\rangle$

In this section, we prove Lemma 10. The proof is based on the spectral property of B. Fix an $x \in \{0,1\}^n$. Then,

$$\begin{aligned} AF(x) &= \sum_{i=1}^{n}F(x+e_i) = \frac{1}{2^n}\sum_i\sum_{z\in C^\perp}f_z(x+e_i) = \frac{1}{2^n}\sum_{z\in C^\perp}\sum_i f(x+z+e_i) \\ &= \frac{1}{2^n}\sum_{z\in C^\perp}Af(x+z) \geq \frac{1}{2^n}\sum_{z\in C^\perp}\lambda f(x+z) = \frac{\lambda}{2^n}\sum_{z\in C^\perp}f_z(x) = \lambda F(x), \end{aligned}$$

Therefore,

$$\langle AF, F\rangle = \mathbb{E}_x\left[AF(x)F(x)\right] \geq \lambda\mathbb{E}_x\left[(F(x))^2\right] = \lambda\mathbb{E}\left[F^2\right]$$

which gives the claim.

2.2 Upper bound on $\langle AF, F\rangle$

In this section, we prove Lemma 11. The proof is based on the properties of the Fourier transform of F.

The following simple result is a crucial ingredient in calculating $\hat{F}$.

Lemma 13 *For $g : \{0,1\}^n \to \mathbb{R}$ and $z \in \{0,1\}^n$, define $g_z : \{0,1\}^n \to \mathbb{R}$ by $g_z(x) = g(x+z)$. Then,*

$$\widehat{g_z}(\alpha) = (-1)^{\alpha\cdot z}\hat{g}(\alpha)$$

PROOF: We have

$$\widehat{g_z}(\alpha) = \frac{1}{2^n}\sum_x g_z(x)(-1)^{\alpha\cdot x} = \frac{1}{2^n}\sum_x g(x+z)(-1)^{\alpha\cdot x} = \frac{1}{2^n}\sum_y g(y)(-1)^{\alpha\cdot(y+z)}$$

where we make the substitution $y = x + z$. Therefore,

$$\widehat{g_z}(\alpha) = (-1)^{\alpha\cdot z}\frac{1}{2^n}\sum_y g(y)(-1)^{\alpha\cdot y} = (-1)^{\alpha\cdot z}\hat{g}(\alpha)$$

□

Lemma 14 (Fourier transform of F)

$$\hat{F}(\alpha) = \widehat{1_{C^\perp}}(\alpha)\hat{f}(\alpha) = \frac{|C^\perp|}{2^n}1_C(\alpha)\hat{f}(\alpha)\ .$$

PROOF: From Lemma 13, we know that $\widehat{f_z}(\alpha) = (-1)^{\alpha\cdot z}\hat{f}(\alpha)$. Therefore,

$$\begin{aligned}\hat{F}(\alpha) &= \frac{1}{2^n}\sum_{z\in C^\perp}\widehat{f_z}(\alpha) = \frac{1}{2^n}\sum_{z\in C^\perp}(-1)^{\alpha\cdot z}\hat{f}(\alpha) = \frac{1}{2^n}\hat{f}(\alpha)\sum_{z\in C^\perp}(-1)^{\alpha\cdot z}\\ &= \hat{f}(\alpha)\mathbb{E}_z[1_{C^\perp}(z)(-1)^{\alpha\cdot z}] = \hat{f}(\alpha)\widehat{1_{C^\perp}}(\alpha)\ .\end{aligned}$$

The full claim follows since $\widehat{1_{C^\perp}} = \frac{|C^\perp|}{2^n}1_C$. □

Remark 15 *Lemma 14 can be directly obtained as follows. Note that*

$$F(x) = \sum_{z\in C^\perp} f(x+z) = \sum_{z\in\{0,1\}^n} 1_{C^\perp}(z)f(x+z) = \left(1_{C^\perp} * f\right)(x)$$

where $*$ represents the 'convolution' operation. It is a well known property that the Fourier transform of a convolution is the product of the Fourier transforms. Formally, for $f, g : \{0,1\}^n \to \mathbb{R}$, we have $\widehat{f * g} = \hat{f}\cdot\hat{g}$. This establishes the claim.

Corollary 16 *Let C be a binary linear code of distance at least d. Suppose $\alpha \in \{0,1\}^n$ is such that $\alpha \neq \mathbf{0}$ and $\mathrm{wt}(\alpha) < d$. Then, $\hat{F}(\alpha) = 0$.*

PROOF: Since $\mathrm{wt}(\alpha) < d$, it follows that $\alpha \notin C$. Therefore, from Lemma 14, $\hat{F}(\alpha) = 0$. □

Lemma 17 (Fourier transform of AF) *For $g : \{0,1\}^n \to \mathbb{R}$,*

$$\widehat{Ag}(\alpha) = \hat{g}(\alpha)(n - 2\mathrm{wt}(\alpha))$$

PROOF: We know that

$$Ag = \sum_i g_{e_i}$$

Therefore,

$$\widehat{Ag}(\alpha) = \sum_i \widehat{g_{e_i}}(\alpha) = \hat{g}(\alpha) \sum_i (-1)^{\alpha \cdot e_i} = \hat{g}(\alpha) \sum_i (-1)^{\alpha_i}$$

where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$. It is easy to check that for $\alpha_i \in \{0,1\}$, $(-1)^{\alpha_i} = 1 - 2\alpha_i$. Plugging this in the previous equation, we get

$$\widehat{Ag}(\alpha) = \hat{g}(\alpha) \sum_i (1 - 2\alpha_i) = \hat{g}(\alpha)(n - 2\mathrm{wt}(\alpha)).$$

□

With these claims in place, we can establish an upper bound on $\langle AF, F \rangle$.

$$\begin{aligned}
\langle AF, F \rangle &= \sum_\alpha \widehat{AF}(\alpha)\hat{F}(\alpha) \\
&= \sum_\alpha \hat{F}(\alpha)^2(n - 2\mathrm{wt}(\alpha)) \\
&= n\hat{F}(0)^2 + \sum_{\alpha : \mathrm{wt}(\alpha) \geq d} \hat{F}(\alpha)^2(n - 2\mathrm{wt}(\alpha))
\end{aligned}$$

using Corollary 16. We now complete the upper bound.

$$\begin{aligned}
\langle AF, F \rangle &\leq n\hat{F}(0)^2 + (n - 2d) \sum_{\alpha : \mathrm{wt}(\alpha) \geq d} \hat{F}(\alpha)^2 \\
&\leq n\hat{F}(0)^2 + (n - 2d) \sum_\alpha \hat{F}(\alpha)^2 \\
&= n\mathbb{E}[F]^2 + (n - 2d)\mathbb{E}\left[F^2\right],
\end{aligned}$$

using $\hat{F}(0) = \mathbb{E}[F]$.

3 Lower bound on the maximum eigenvalue of Hamming balls

We are interested in lower bounding the maximum eigenvalue of the Hamming ball $B(0,r)$, where $r = \gamma n$ for $\gamma < 1/2$. We will restrict ourselves to an $f : \{0,1\}^n \to \mathbb{R}$, such that $f(x)$ depends on *only* on the weight of x, and has support $\{x : r - M \leq \mathrm{wt}(x) \leq r\} \subseteq B(0,r)$, for some $M = o(n)$. (For instance, we could choose $M = n^{3/4}$.) Define f as follows:

$$f(x) = \begin{cases} \frac{1}{\sqrt{\binom{n}{i}}}, & \text{if } \mathrm{wt}(x) = i \in [r-M, r], \\ 0, & \text{otherwise.} \end{cases}$$

For convenience, we will denote by $f(i)$ the evaluation of f at any x with weight i. Let us now compute $\langle f, f\rangle$ and $\langle Af, f\rangle$ respectively.

$$2^n\langle f, f\rangle = \sum_{i=r-M}^{r} \sum_{x:\mathrm{wt}(x)=i} f(x)^2 = \sum_{i=r-M}^{r} \binom{n}{i} f(i)^2 = M + 1 \leq M(1 + o(1)) \tag{5}$$

On the other hand,

$$2^n\langle Af, f\rangle = \sum_x Af(x)f(x) = \sum_i \sum_{x:\mathrm{wt}(x)=i} Af(x)f(x)$$

Fix an $x \in \{0,1\}^n$ of weight i. Therefore, of the n neighbors of x, i have a weight $i-1$, and the remaining $n-i$ have a weight $i+1$. Therefore,

$$Af(x)f(x) = f(x)\sum_{j=1}^{n} f(x + e_j) = f(i)\left(if(i-1) + (n-i)f(i+1)\right)$$

Therefore,

$$\begin{aligned} 2^n\langle Af, f\rangle &= \sum_{i=1}^{n} \left(i\binom{n}{i} f(i)f(i-1) + (n-i)\binom{n}{i} f(i)f(i+1) \right) \\ &= \sum_{i=r-M+1}^{r} i\sqrt{\frac{\binom{n}{i}}{\binom{n}{i-1}}} + \sum_{i=r-M}^{r-1} (n-i)\sqrt{\frac{\binom{n}{i}}{\binom{n}{i+1}}} \\ &= \sum_{i=r-M+1}^{r} \sqrt{i(n-i+1)} + \sum_{i=r-M}^{r-1} \sqrt{(n-i)(i+1)} \\ &= 2\sum_{i=r-M+1}^{r} \sqrt{i(n-i+1)} \end{aligned}$$

For the values of r and i we are interested in, it is easy to see that

$$i(n-i+1) \geq (r-M+1)(n-r+M) \geq r(n-r) - o(n^2).$$

Hence,

$$2^n\langle Af, f\rangle \geq M\left(2\sqrt{r(n-r)} - o(n)\right) \tag{6}$$

Combining Equations (6) and (5), we get

$$\lambda_{B(0,r)} \geq \frac{\langle Af, f\rangle}{\langle f, f\rangle} \geq \frac{2\sqrt{r(n-r)} - o(n)}{1 + o(1)} = 2\sqrt{r(n-r)} - o(n)$$

For $r = \gamma n$, this gives the bound

$$\lambda_{B(0,r)} \geq 2n\sqrt{\gamma(1-\gamma)} - o(n)$$

4 Some remarks and the second MRRW bound

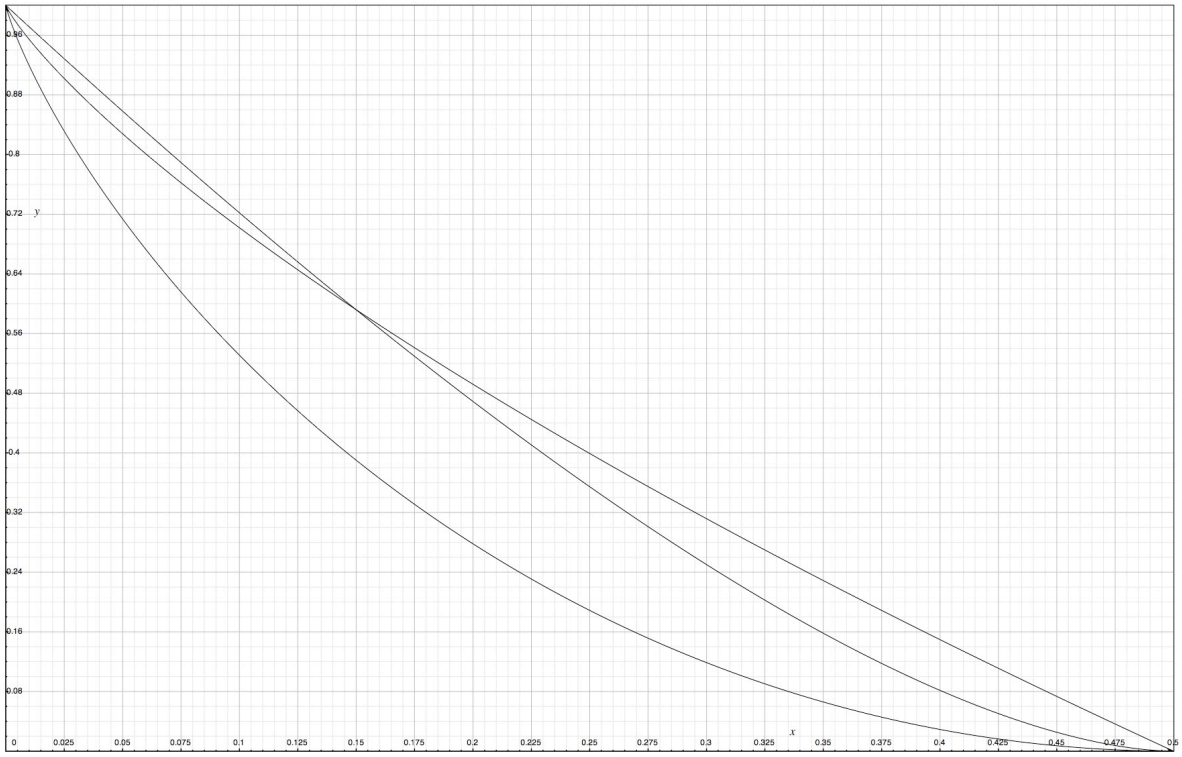
Though we proved the bound only for linear codes, the bound holds also for general codes, and in fact can be proved within the same Fourier analytic framework; see the final section of the paper by Navon and Samorodnitsky for the details.

The first MRRW bound can also be generalized to larger alphabets, giving the following statement.

Theorem 18 (First MRRW bound for larger alphabets) *The rate of a q-ary code of relative distance δ, $0 < \delta < 1 - 1/q$, is at most*

$$h_q\Big(\frac{1}{q}\big(q - 1 - (q-2)\delta - 2\sqrt{(q-1)\delta(1-\delta)}\big)\Big) + o(1) \ .$$

Below is a plot of the best bounds on the best possible rate $R(\delta)$ as a function of relative distance δ we have seen so far for binary codes: the Gilbert-Varshamov lower bound $R(\delta) \geq 1 - h(\delta)$, the Elias-Bassalygo bound $R(\delta) \leq 1 - h(\frac{1-\sqrt{1-2\delta}}{2})$, and the first MRRW bound $R(\delta) \leq h(\frac{1}{2} - \sqrt{\delta(1-\delta)})$.



Note that the first MRRW bound is weaker than the Elias-Bassalygo bound for δ smaller than about 0.15 (in fact it is even weaker than the Hamming bound for $\delta \lesssim 0.11$). There is a strenghtening of the bound, called the second MRRW bound, which uses the inequality

$$A(n,d) \leq \frac{2^n}{\binom{n}{w}} A(n,d,w)$$

together with an upper bound on the size $A(n,d,w)$ of constant-weight codes via Delsarte's linear programming approach applied to the "Johnson" association scheme. We state the bound here without proof. This bound beats the Elias-Bassalygo bound for the entire range $\delta \in [0,1/2]$. The bound coincides with the first MRRW bound for $\delta > 0.273$. The second MRRW bound gives the best upper bound on $R(\delta)$ for the entire range of δ and has not been improved upon in over three decades!

Theorem 19 (Second MRRW bound for binary codes) *Let $0 < \delta < 1/2$. The largest rate of a binary code of relative distance δ is at most* $\mathrm{MRRW}^{(2)}(\delta) + o(1)$ *where*

$$\mathrm{MRRW}^{(2)}(\delta) = \min_{\delta/2 \leq \xi \leq 1/2} \left\{ 1 - h(\xi) + R_{\mathrm{cw}}(\xi, \delta) \right\} \tag{7}$$

with

$$R_{\mathrm{cw}}(\xi,\delta) = \begin{cases} h\left(\frac{1}{2}\left(1 - \sqrt{1 - \left(\sqrt{4\xi(1-\xi) - 2\delta + \delta^2} - \delta\right)^2}\right)\right) & \text{if} \quad \delta \leq 2\xi(1-\xi) \\ 0 & \text{otherwise.} \end{cases}$$

The above bound encompasses the Hamming, Elias-Bassalygo, and first MRRW bounds.

Exercise 4 *1. Verify that the choice $\xi = 1/2$ in the the minimization in (7) yields the first MRRW bound.*

2. Verify that the choice $\xi = \delta/2$ in the minimization in (7) yields the Hamming bound.

3. Verify that picking ξ so that $2\xi(1-\xi) = \delta$ in the minimization in (7) yields the Elias-Bassalygo bound.

As far as I am aware the second MRRW bound only applies for binary codes and has not been extended to larger alphabets.

Introduction to Coding Theory **CMU: Spring 2010**

Notes 6: Reed-Solomon, BCH, Reed-Muller, and concatenated codes

February 2010

Lecturer: Venkatesan Guruswami *Scribe: Eric Blais & Venkat Guruswami*

In this lecture, we begin the algorithmic component of the course by introducing some explicit families of good algebraic codes. We begin by looking at Reed-Solomon codes.

1 Reed-Solomon codes

Reed-Solomon codes are a family of codes defined over large fields as follows.

Definition 1 (Reed-Solomon codes) *For integers $1 \leq k < n$, a field $\mathbb{F}$ of size $|\mathbb{F}| \geq n$, and a set $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$, we define the Reed-Solomon code*

$$\mathrm{RS}_{\mathbb{F},S}[n,k] = \{\, (p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_n)) \in \mathbb{F}^n \mid p \in \mathbb{F}[X] \text{ is a polynomial of degree } \leq k-1 \,\}.$$

A natural interpretation of the $\mathrm{RS}_{\mathbb{F},S}[n,k]$ code is via its encoding map. To encode a message $m = (m_0, m_1, \ldots, m_{k-1}) \in \mathbb{F}^k$, we interpret the message as the polynomial

$$p(X) = m_0 + m_1 X + \cdots + m_{k-1} X^{k-1} \in \mathbb{F}[X].$$

We then evaluate the polynomial p at the points $\alpha_1, \alpha_2, \ldots, \alpha_n$ to get the codeword corresponding to m.

To evaluate the polynomial p on the points $\alpha_1, \alpha_2, \ldots, \alpha_n$, we multiply the message vector m on the left by the $n \times k$ Vandermonde matrix

$$G = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^{k-1} \end{pmatrix}.$$

The matrix G is a generator matrix for $\mathrm{RS}_{\mathbb{F},S}[n,k]$, so we immediately obtain that Reed-Solomon codes are linear codes over $\mathbb{F}$.

1.1 Properties of the code

Let's now examine the parameters of the above Reed-Solomon code. The block length of the code is clearly n. As we will see, the code $\mathrm{RS}_{\mathbb{F},S}[n,k]$ has minimum distance $n-k+1$. This also means that the encoding map is injective and therefore the code has dimension equal to k.

The key to establishing the minimum distance of Reed-Solomon codes is the 'degree mantra' that we saw in the previous lecture: *A non-zero polynomial of degree d with coefficients from a field $\mathbb{F}$ has at most d roots in $\mathbb{F}$.*

Theorem 2 *The Reed-Solomon code $\mathrm{RS}_{\mathbb{F},S}[n,k]$ has distance $n-k+1$.*

PROOF: Since $\mathrm{RS}_{\mathbb{F},S}[n,k]$ is a linear code, to prove the theorem it suffices to show that any non-zero codeword has Hamming weight at least $n-k+1$.

Let $(m_0, m_1, \ldots, m_{k-1}) \neq 0$. The polynomial $p(X) = m_0 + m_1 X + \cdots + m_{k-1}X^{k-1}$ is a non-zero polynomial of degree at most $k-1$. So by our degree mantra, p has at most $k-1$ roots, which implies that $(p(\alpha_1), \ldots, p(\alpha_n))$ has at most $k-1$ zeros.

By the Singleton bound, the distance cannot exceed $n-k+1$, and therefore must equal $n-k+1$. The upper bound on distance can also be seen by noting that the codeword corresponding to the polynomial $\prod_{i=1}^{k-1}(X-\alpha_i)$ has Hamming weight exactly $n-k+1$. □

Note that the minimum distance of Reed-Solomon codes meets the Singleton bound. This is quite interesting: Reed-Solomon codes are a simple, natural family of codes based only on univariate polynomials, and yet their rate is optimal.

In our definition above, we have presented Reed-Solomon codes in the most general setting, where S can be any arbitrary subset of $\mathbb{F}$ of size n. This presentation highlights the flexibility of Reed-Solomon codes. In practice, however, there are two common choices of S used to instantiate Reed-Solomon codes:

1. Take $S = \mathbb{F}$, or
2. Take $S = \mathbb{F}^*$ to be the set of non-zero elements in $\mathbb{F}$.

These two choices attain the best possible trade-off between the field size and the block length.

1.2 Alternative characterization

We presented Reed-Solomon codes from an encoding point of view. It is also possible to look at these codes from the "parity-check" point of view. This approach is used in many textbooks, and leads to the following characterization of Reed-Solomon codes.

Theorem 3 (Parity-check characterization) *For integers $1 \leq k < n$, a field $\mathbb{F}$ of size $|\mathbb{F}| = q = n+1$, a primitive element $\alpha \in \mathbb{F}^*$, and the set $S = \{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, the Reed-Solomon code over $\mathbb{F}$ with evaluation set S is given by*

$$\mathrm{RS}_{\mathbb{F},S}[n,k] = \{\, (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}^n \mid c(X) = c_0 + c_1 X + \cdots + c_{n-1}X^{n-1} \text{ satisfies } c(\alpha) = c(\alpha^2) = \cdots = c(\alpha^{n-k}) = 0 \,\}. \quad (1)$$

In other words, Theorem 3 states that the codewords of the Reed-Solomon code with evaluation points $1, \alpha, \ldots, \alpha^{n-1}$ correspond to the polynomials of degree $n-1$ that vanish at the points $\alpha, \alpha^2, \ldots, \alpha^{n-k}$.

The characterization of Reed-Solomon codes in Theorem 3 has the same dimension as the code obtained with our original definition; to complete the proof of Theorem 3, we only need to check that every codeword obtained in Definition 1 satisfies the parity-check condition (1).

Exercise 1 *Complete the proof of Theorem 3. (Hint: The proof uses the fact that for every $x \neq 1$ in $\mathbb{F}^*$, $\sum_{\ell=0}^{n-1} x^\ell = \frac{1-x^n}{1-x} = 0$.)*

1.3 Applications

Reed-Solomon codes were originally introduced by Reed and Solomon in 1960 [6]. There have been many other codes introduced since – we will see some of those more recent codes soon – and yet Reed-Solomon codes continue to be used in many applications. Most notably, they are extensively used in storage devices like CDs, DVDs, and hard-drives.

Why are Reed-Solomon codes still so popular? One important reason is because they are optimal codes. But they do have one downside: Reed-Solomon codes require a large alphabet size. In a way, that is unavoidable; as we saw in Notes 4, any code that achieves the Singleton bound must be defined over a large alphabet.

The large alphabet brings to the fore an important issue: if we operate on bits, how do we convert the codewords over the large field in the binary alphabet? There is one obvious method. Say, for example, that we have a code defined over $\mathbb{F}_{256}$. Then we can write an element in this field as an 8-bit vector.

More precisely: if we have a message that corresponds to the polynomial $p(X)$ in $\mathbb{F}[X]$, its encoding in the Reed-Solomon code is the set of values $p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_m)$. We can simply express these values in a binary alphabet with $\log|\mathbb{F}|$ bits each. So provided that the Reed-Solomon code is defined over a field that is an extension field of $\mathbb{F}_2$, then this simple transformation yields a code over $\mathbb{F}_2$. In fact, there is way to represent field elements as bit vectors so that the resulting code is a binary *linear* code.

This method is in fact what is done in practice. But then it leads to the natural question: What are the error correction capabilities of the resulting binary code?

Let's look at an example: say we have a Reed-Solomon code with $n = 256$ and $k = 230$. The distance of this code is $d = 27$, so the code can correct 13 errors. The transformation to a binary code yields a binary code where $n' = 256 \cdot 8$ and $k' = 230 \cdot 8$, since all we have done in the transformation is scale everything. And at worst the distance of the resulting binary code is $d' \geq d = 27$, so the binary code can also correct at least 13 errors.

Let us now generalize the example. If we have a $[N, K, D]_{\mathbb{F}}$ code where $|\mathbb{F}| = N$ and N is a power of 2, then the transformation described above yields a $[N \log N, K \log N, D']_2$ binary linear code, where $D' \geq D$. Writing $n = N \log N$ and considering the case where $K = N - D + 1$, we observe that the transformation of a Reed-Solomon code to a binary code results in a $[n, n-(D-1)\log N, \geq D]_2$ code.

The resulting binary code has a decent rate, but it is not optimal: *BCH codes* are even better, as they are $[n, n - \lceil \frac{D-1}{2} \rceil \log(n+1), \geq D]_2$ codes. BCH codes are very interesting in their own right, and we will examine them in the next section. But first we return to the question that we posed at

the beginning of this section: why are Reed-Solomon codes still so popular? If BCH codes have the same distance guarantees as Reed-Solomon codes and a better rate, one would expect these codes to have completely replaced Reed-Solomon codes.

The main reason that Reed-Solomon are still frequently used is that in many applications – and in particular in storage device applications – errors often occur in bursts. Reed-Solomon codes have the nice property that bursts of consecutive errors affect bits that correspond to a much smaller number of elements in the field on which the Reed-Solomon code is defined. For example, if a binary code constructed from the $\mathrm{RS}_{\mathbb{F}_{256}}[256, 230]$ code is hit with 30 consecutive errors, these errors affect at most 5 elements in the field $\mathbb{F}_{256}$ and this error is easily corrected.

2 BCH codes

BCH codes were discovered by independently by Bose and Ray-Chaudhuri [1] and by Hocquenghem [3] in the late 1950s. As we saw in the previous section, BCH codes have better rate than binary codes constructed from Reed-Solomon codes. In fact, as we will see later in the section, the rate of BCH codes is optimal, up to lower order terms.

BCH codes can be defined over any field, but for today's lecture we will focus on binary BCH codes:

Definition 4 (Binary BCH codes) *For a length $n = 2^m - 1$, a distance D, and a primitive element $\alpha \in \mathbb{F}_{2^m}^*$, we define the binary BCH code*

$$\mathrm{BCH}[n, D] = \{\, (c_0, \ldots, c_{n-1}) \in \mathbb{F}_2^n \mid c(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1} \text{ satisfies } c(\alpha) = c(\alpha^2) = \cdots = c(\alpha^{D-1}) = 0 \,\}.$$

This definition should look familiar: it is almost exactly the same as the alternative characterization of Reed-Solomon codes in Theorem 3. There is one important difference: in Theorem 3, the coefficients $c_0, \ldots, c_{n-1}$ could take any value in the extension field, whereas here we restrict the coefficients to take values only from the base field (i.e., the coefficients each take values from $\mathbb{F}_2$ instead of $\mathbb{F}_{2^m}$).

The BCH codes form linear spaces. The definition gives the parity-check view of the linear space, as it defines the constraints over the elements. The constraint $c(\alpha) = 0$ is a constraint over the extension field $\mathbb{F}_{2^m}$, but it can also be viewed as a set of m linear constraints over $\mathbb{F}_2$.

The last statement deserves some justification. That each constraint over $\mathbb{F}_{2^m}$ corresponds to m constraints over $\mathbb{F}_2$ is clear from the vector space view of extension fields. That the resulting constraints are linear is not as obvious but follows from the argument below.

Consider the (multiplication) transformation $\mathrm{Mult}_\alpha : x \mapsto \alpha x$ defined on $\mathbb{F}_{2^m}$. This map is $\mathbb{F}_2$-linear, since $\alpha(x + y) = \alpha x + \alpha y$. Using the additive vector space structure of $\mathbb{F}_{2^m}$, we can pick a basis $\{\beta_1 = 1, \beta_2, \ldots, \beta_m\} \subset \mathbb{F}_{2^m}$ of $\mathbb{F}_{2^m}$ over $\mathbb{F}$, and represent each element $x \in \mathbb{F}_{2^m}$ as the (column) vector $(x_1, x_2, \ldots, x_m)^T \in \mathbb{F}_2^m$ where $x = x_1\beta_1 + x_2\beta_2 + \cdots + x_m\beta_m$. The $\mathbb{F}_2$-linear multiplication map Mult_α then corresponds to a linear transformation of this vector representation, mapping $x = (x_1, \ldots, x_m)^T$ to $M_\alpha x$ for a matrix $M_\alpha \in \mathbb{F}_2^{m \times m}$. And the coefficients $c_i \in \mathbb{F}_2$ correspond to vectors $(c_i, 0, \ldots, 0)^T \in \mathbb{F}_2^m$ so the constraint $c(\alpha) = c_0 + c_1\alpha + c_2\alpha^2 + \cdots + c_{n-1}\alpha^{n-1} = 0$ is

equivalent to the constraint

$$\begin{pmatrix} c_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + M_\alpha \begin{pmatrix} c_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \cdots + M_\alpha^{n-1} \begin{pmatrix} c_{n-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

which yields m linear constraints over $\mathbb{F}_2$.

2.1 Parameters of BCH codes

The block length of the BCH$[n, D]$ code is n, and its distance is at least D. The latter statement is seen most easily by noting that the BCH code is a subcode of Reed-Solomon codes (i.e., the codewords of the BCH code form a subset of the codewords of the corresponding Reed-Solomon code), so the distance of the BCH code is bounded below by the distance of the Reed-Solomon code.

The dimension of the BCH$[n, D]$ code is a bit more interesting. The dimension of the code is at least $n - (D-1)\log(n+1)$, since in our definition we have $D-1$ constraints on the extension field that each generate $m = \log(n+1)$ constraints in the base field. But this bound on the dimension is not useful: it is (almost) identical to the dimension of Reed-Solomon codes converted to the binary alphabet (for a similar distance and block length), so if this bound were tight we would have no reason for studying BCH codes. This bound, however, can be tightened, as the following more careful analysis shows.

Lemma 5 *For a length $n = 2^m - 1$ and a distance D, the dimension of the* BCH$[n, D]$ *code is at least $n - \lceil \frac{D-1}{2} \rceil \log(n+1)$.*

Proof: In order to establish the tighter bound on the dimension of BCH codes, we want to show that some of the constraints in Definition 4 are redundant. We do so by showing that for any polynomial $c(X) \in \mathbb{F}_2[X]$ and any element $\gamma \in \mathbb{F}_2$, if we have $c(\gamma) = 0$, then we must also have $c(\gamma^2) = 0$. We establish this fact below.

Let $c(X) \in \mathbb{F}_2[X]$ and $\gamma \in \mathbb{F}_2$ be such that $c(\gamma) = 0$. Then we also have $c(\gamma)^2 = 0$, so

$$(c_0 + c_1\gamma + c_2\gamma^2 + \cdots + c_{n-1}\gamma^{n-1})^2 = 0.$$

For any two elements $\alpha, \beta \in \mathbb{F}_{2^m}$, $(\alpha+\beta)^2 = \alpha^2 + \beta^2$, so $c(\gamma) = 0$ also implies

$$\begin{aligned} & c_0^2 + (c_1\gamma)^2 + (c_2\gamma^2)^2 + \cdots + (c_{n-1}\gamma^{n-1})^2 &&= 0 \\ \Leftrightarrow \quad & c_0^2 + c_1^2\gamma^2 + c_2^2(\gamma^2)^2 + \cdots + c_{n-1}^2(\gamma^2)^{n-1} &&= 0. \end{aligned}$$

Since the coefficients $c_0, c_1, \ldots, c_{n-1}$ are in $\mathbb{F}_2$, $c_i^2 = c_i$ for all $i = 0, 1, \ldots, n-1$. Therefore, $c(\gamma) = 0$ implies that

$$c_0 + c_1\gamma^2 + c_2(\gamma^2)^2 + \cdots + c_{n-1}(\gamma^2)^{n-1} = c(\gamma^2) = 0,$$

which is what we wanted to show.

To complete the proof, we now observe that the fact we just proved implies that the constraints $c(\alpha^{2j}) = 0$ for $j = 1, 2, \ldots, \lfloor \frac{D-1}{2} \rfloor$ are all redundant (and implies by $c(\alpha^j) = 0$); we can remove these constraints from the definition without changing the set of codewords in a BCH code. Doing this operation leaves $\lceil \frac{D-1}{2} \rceil m = \lceil \frac{D-1}{2} \rceil \log(n+1)$ constraints. $\square$

Remark 6 *The bound in Lemma 5 is asymptotically tight; the 2 can not be improved to $2 - \epsilon$ for any $\epsilon > 0$.*

The asymptotic tightness of the bound in Lemma 5 follows from the Hamming bound.

2.2 Alternative characterization

Another interpretation of the BCH$[n, D]$ code is that it is equivalent to taking the definition of Reed-Solomon codes, and modifying it to keep only the polynomials where all the evaluations lie in the base field. In fact, this interpretation leads to the following corollary to Theorem 3. The proof follows immediately from Theorem 3 and Definition 4 of BCH codes.

Corollary 7 *BCH codes are subfield subcodes of Reed-Solomon codes. Specifically,*

$$\text{BCH}[n, D] = \text{RS}_{\mathbb{F}, \mathbb{F}^*}[n, n - D + 1] \cap \mathbb{F}_2^n.$$

An important implication of Corollary 7 is that any decoder for Reed-Solomon codes also yields a decoder for BCH codes. Therefore, in later lectures, we will only concentrate on devising efficient algorithms for decoding Reed-Solomon codes; those same algorithms will immediately also give us efficient decoding of BCH codes.

2.3 Analysis and applications

Just like Hamming codes, BCH codes have a very good rate but are only useful when we require a code with small distance (in this case, BCH codes are only useful when $D \leq \frac{n}{\log n}$). In fact, there is a much closer connection between Hamming codes and BCH codes:

Exercise 2 *For $n = 2^m - 1$, show that the* BCH$[n, 3]$ *code is the same (after perhaps some coordinate permutation) as the Hamming code* $[2^m - 1, 2^m - 1 - m, 3]_2$.

As we mentioned above, we are not particularly interested in BCH codes from an algorithmic point of view, since efficient decoding of Reed-Solomon codes also implies efficient decoding of BCH codes. But there are some applications where the improved bound in the dimension of BCH code is crucial.

In particular, one interesting application of BCH codes is in the generation of k-wise independent distributions. A distribution over n-bit strings is k-wise independent if the strings generated by this distribution look completely random if you look only at k positions of the strings. The simplest way to generate a k-wise independent distribution is to generate strings by the uniform distribution. But this method requires a sample space with 2^n points. Using BCH codes, it is possible to generate k-wise independent distributions with a sample space of only $\approx n^{k/2}$ points.

3 Reed-Muller codes

The BCH codes we introduced were a generalization of Hamming codes. We now generalize the dual of Hamming codes – Hadamard codes. The result is another old family of algebraic codes called Reed-Muller codes. We saw in Notes 1 that Hadamard codes were related to first-order Reed-Muller codes; we now obtain the full class of Reed-Muller codes by considering polynomials of larger degree.

Reed-Muller codes were first introduced by Muller in 1954 [4]. Shortly afterwards, Reed provided the first efficient decoding algorithm for these codes [5]. Originally, only binary Reed-Muller codes were considered, but we will describe the codes in the more general case. The non-binary setting is particularly important: in many applications of codes in computational complexity, Reed-Muller codes over non-binary fields have been used to obtain results that we are still unable to achieve with any other family of codes. We saw one such example, of hardness amplification using Reed-Muller codes, in the Introduction to Computational Complexity class last year.

Definition 8 (Reed-Muller codes) *Given a field size q, a number m of variables, and a total degree bound r, the $\mathrm{RM}_q[m,r]$ code is the linear code over $\mathbb{F}_q$ defined by the encoding map*

$$f(X_1,\ldots,X_m) \to \langle f(\alpha)\rangle\,|_{\alpha\in\mathbb{F}_q^m}$$

applies to the domain of all polynomials in $\mathbb{F}_q[X_1,X_2,\ldots,X_m]$ of total degree $\deg(f)\le r$.

Reed-Muller codes form a strict generalization of Reed-Solomon codes: the latter were defined based on univariate polynomials, while we now consider polynomials over many variables.

There is one term in the definition of Reed-Muller codes that we have not yet defined formally: the *total degree* of polynomials. We do so now: the total degree of the monomial $X_1^{k_1}X_2^{k_2}\cdots X_m^{k_m}$ is $k_1+k_2+\cdots+k_m$, and the total degree of a polynomial is the maximum total degree over all its monomials that have a nonzero coefficient.

3.1 Properties of the code

The block length of the $\mathrm{RM}_q[m,r]$ code is q^m, and the dimension of the code is the number of polynomials in $\mathbb{F}_q[X_1,X_2,\ldots,X_m]$ of degree at most r.

When $q=2$, the size of the $\mathrm{RM}_2[m,r]$ can be computed explicitly: there are $\binom{m}{0}+\binom{m}{1}+\cdots+\binom{m}{r}$ ($\approx m^r$) such polynomials.

In general, for any $q\ge 2$ the number of polynomials on m variables of total degree at most r is

$$\left|\left\{(i_1,\ldots,i_m)\mid 0\le i_j\le q-1,\sum_{j=1}^{m} i_j\le r\right\}\right|.$$

When $q>2$ this count does not have a simple expression.

As with Reed-Solomon codes, the interesting parameter of Reed-Muller codes is their distance. To compute the distance parameter, we look for the minimum number of zeros of any non-zero

polynomial. Since $\alpha^q = \alpha$ for $\alpha \in \mathbb{F}_q$, when considering m-variate polynomials over $\mathbb{F}_q$ which will be evaluated at points in $\mathbb{F}_q^m$, we can restrict the degree in each variable X_i to be at most $q-1$. The distance property of Reed-Solomon codes was a consequence of the following fundamental result: a non-zero univariate polynomial of degree at most d over a field has at most d roots. The Schwartz-Zippel Lemma extends the degree mantra to give a bound on the number of roots of multi-variate polynomials.

Theorem 9 (Number of zeroes of multivariate polynomials) *Let $f \in \mathbb{F}_q[X_1, \ldots, X_m] \neq 0$ be a polynomial of total degree r, with the maximum individual degree in the X_i's bounded by $q-1$. Then*

$$\Pr_{\alpha \in \mathbb{F}_q^m}[f(\alpha) \neq 0] \geq \frac{1}{q^a}\left(1 - \frac{b}{q}\right)$$

where $r = a(q-1) + b$, $0 \leq b < q-1$.

The proof of the Schwartz-Zippel Lemma follows from two slightly simpler lemmas. The first lemma provides a good bound on the number of roots of a multi-variate polynomial when its total degree is smaller than the degree of the underlying field.

Lemma 10 (Schwartz [7]) *Let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a non-zero polynomial of total degree at most $\ell < q$. Then*

$$\Pr_{(a_1,\ldots,a_m) \in \mathbb{F}_q^m}[f(a_1, \ldots, a_m) = 0] \leq \frac{\ell}{q}.$$

Proof: The proof of Lemma 10 is by induction on the number of variables in the polynomial. In the base case, when f is a univariate polynomial, the lemma follows directly from the degree mantra.

For the inductive step, consider the decomposition

$$\begin{aligned} f(X_1, \ldots, X_m) = X_m^{d_m}\, g_m(X_1, \ldots, X_{m-1}) + \cdots \\ + X_m\, g_1(X_1, \ldots, X_{m-1}) + g_0(X_1, \ldots, X_{m-1}) \end{aligned}$$

where d_m is the degree of f in X_m. Then g_m is a non-zero polynomial of total degree at most $\ell - d_m$. By the induction hypothesis,

$$\Pr_{(\alpha_1,\ldots,\alpha_{m-1}) \in \mathbb{F}_q^{m-1}}[g_m(\alpha_1, \ldots, \alpha_{m-1}) = 0] \leq \frac{\ell - d_m}{q}. \tag{2}$$

Also, when $g_m(\alpha_1, \ldots, \alpha_{m-1}) \neq 0$, then $f(\alpha_1, \ldots, \alpha_{m-1}, X_m)$ is a non-zero univariate polynomial of degree at most d_m, so we have

$$\Pr_{(\alpha_1,\ldots,\alpha_m) \in \mathbb{F}_q^m}[f(\alpha_1, \ldots, \alpha_m) = 0 \mid g_m(\alpha_1, \ldots, \alpha_{m-1}) \neq 0] \leq \frac{d_m}{q}. \tag{3}$$

Therefore,

$$\begin{aligned} \Pr_{(a_1,\ldots,a_m) \in \mathbb{F}_q^m}[f(a_1, \ldots, a_m) = 0] &\leq \Pr_{(\alpha_1,\ldots,\alpha_{m-1}) \in \mathbb{F}_q^{m-1}}[g_m(\alpha_1, \ldots, \alpha_{m-1}) = 0] \\ &\quad + \Pr_{(\alpha_1,\ldots,\alpha_m) \in \mathbb{F}_q^m}[f(\alpha_1, \ldots, \alpha_m) = 0 \mid g_m(\alpha_1, \ldots, \alpha_{m-1}) \neq 0] \\ &\leq \frac{\ell - d_m}{q} + \frac{d_m}{q} = \frac{\ell}{q}. \end{aligned}$$

□

Remark 11 *A version of Lemma 10 can also be stated for infinite fields (or integral domains). Specifically, the same proof shows that for any field $\mathbb{F}$ and any subset $S \subseteq \mathbb{F}$, the probability that a non-zero polynomial of total degree ℓ is zero is at most $\frac{\ell}{|S|}$ when the values of the variables are chosen independently and uniformly at random from S.*

In many computer science applications, the field size q is very large, and the bound of Lemma 10 is sufficient. As a result, that lemma is often presented as the Schwartz-Zippel Lemma. For our analysis of Reed-Muller codes, however, we also need to bound the probability that a multi-variate polynomial is zero when the degree of the underlying field is small. The following lemma gives us a good bound in this setting.

Lemma 12 (Zippel [8]) *Let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a non-zero polynomial with maximum degree $\deg_{X_i}(f) \leq d_i$ for $i = 1, \ldots, m$. Then*

$$\Pr_{(a_1,\ldots,a_m)\in\mathbb{F}_q^m}[f(a_1, \ldots, a_m) \neq 0] \geq \frac{\prod_{i=1}^{m}(q - d_i)}{q^m}.$$

PROOF: We again proceed with a proof by induction on the number of variables. When f is univariate, the lemma follows since a degree d polynomial has at most d zeroes.

For the inductive step, consider again the decomposition

$$f(X_1, \ldots, X_m) = X_m^{d_m}\, g_m(X_1, \ldots, X_{m-1}) + \cdots + X_m\, g_1(X_1, \ldots, X_{m-1}) + g_0(X_1, \ldots, X_{m-1}).$$

The decomposition say that we can think of the (multi-variate) polynomial f as a univariate polynomial in $\mathbb{F}_q[X_1, \ldots, X_{m-1}][X_m]$. That is, f can be viewed as a polynomial on the variable X_m with coefficients coming from $K = \mathbb{F}_q(X_1, \ldots, X_{m-1})$, the field of rational functions in variables $X_1, X_2, \ldots, X_{m-1}$. By the degree mantra for univariate polynomials, we get that there are at most d_m values $\beta \in K$ for which $f(X_1, X_2, \ldots, X_{m-1}, \beta) = 0$ (in the field K). Thus there are certainly at least $q - d_m$ values $\alpha \in \mathbb{F}_q$ that can be assigned to X_m such that $f(X_1, \ldots, X_{m-1}, \alpha)$ is a non-zero polynomial (on $m - 1$ variables). Applying the induction hypothesis to this polynomial completes the proof of the lemma. □

To complete the proof of Theorem 9, we can apply Lemma 12 repeatedly to a polynomial, removing one variable at a time, until the total degree ℓ of the polynomial on the remaining variables satisfies $\ell < q$, and then we can apply Lemma 10. We leave the details of the proof to the reader.

It is reasonable to ask if the bound of Theorem 9 could be improved. In general, it can't. Consider the polynomial

$$f(X_1, \ldots, X_{a+1}) = \prod_{i=1}^{a} \prod_{\alpha\in\mathbb{F}_q^*} (X_i - \alpha) \prod_{\beta\in\{\beta_1,\ldots,\beta_b\}\subset\mathbb{F}_q^*} (X_{a+1} - \beta).$$

The polynomial $f(X_1, \ldots, X_{a+1})$ has total degree $r = a(q - 1) + b$ and maximum degree $q - 1$. The value of $f(X_1, \ldots, X_{a+1})$ is non-zero only when $X_1 = \cdots = X_a = 0$ and $X_{a+1} \notin \{\beta_1, \ldots, \beta_b\}$. The first condition is satisfied with probability $\frac{1}{q^a}$ and the second with probability $(1 - \frac{b}{q})$. So the bound of Theorem 9 is tight.

4 Reed-Muller codes

We can now use the Schwartz-Zippel Lemma to establish the distance parameter of binary Reed-Muller codes.

Recall that the $\mathrm{RM}(m, r)$ binary Reed-Muller code is defined by

$$\mathrm{RM}(m, r) = \{\langle f(\alpha)\rangle_{\alpha \in \mathbb{F}_2^m} \mid f \text{ has total degree } \leq r\}.$$

The block length of this code is $n = 2^m$, and the dimension of this code is

$$k = \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r},$$

which is can be roughly approximated by $k \approx m^r$.

Applying Theorem 9 (or Lemma 12) to $q = 2$, we can conclude that the distance of $\mathrm{RM}(m, r)$ is at least 2^{m-r}. We will reprove with a more specialized argument and also show that the distance is exactly 2^{m-r}.

4.1 Decoding Reed-Muller codes

The Reed-Muller codes were first introduced by Muller in 1954 [4]. Muller showed that the family of codes he introduced had good distance parameters, but he did not study the problem of decoding these codes efficiently.

The naïve method of decoding the $\mathrm{RM}(m, r)$ code is to enumerate all the codewords, compute their distance to the received word and to output the one with the minimum distance. This algorithm runs in time $2^k \approx 2^{m^r} = 2^{(\log n)^r}$. The running time of the naïve decoding algorithm is therefore quasi-polynomial (but not polynomial!) in the block length n.

Reed introduced the first efficient algorithm for decoding Reed-Muller codes [5] shortly after the codes were introduced by Muller. Reed's algorithm also corrects up to half the minimum distance (i.e., up to $2^{m-r-1} - 1$ errors) and further runs in time polynomial in the block length n.

We will not cover Reed's decoding algorithm for Reed-Muller codes in this class. At a very high level, the idea of the algorithm is to apply a majority logic decoding scheme. The algorithm was covered in previous iterations of this class; interested readers are encouraged to consult those notes for more details on the algorithm.

4.2 Distance of Reed-Muller codes

Let us now give a self-contained argument proving that the distance of the $\mathrm{RM}(m, r)$ code is 2^{m-r}.

We begin by showing that the distance of binary Reed-Muller codes is at most 2^{m-r}. Since Reed-Muller codes are linear codes, we can do so by exhibiting a non-zero codeword of $\mathrm{RM}(m, r)$ with weight 2^{m-r}. Consider the polynomial

$$f(X_1, \ldots, X_m) = X_1 X_2 \cdots X_r.$$

The polynomial $f \in \mathbb{F}_2[X_1, \ldots, X_m]$ is a non-zero polynomial of degree r, and clearly $f(\alpha_1, \ldots, \alpha_m) \neq 0$ only when $\alpha_1 = \alpha_2 = \cdots = \alpha_r = 1$. There are 2^{m-r} choices of $\alpha \in \mathbb{F}_2^m$ that satisfy this condition, so $\mathrm{wt}(\langle f(\alpha) \rangle_{\alpha \in \mathbb{F}_2^m}) = 2^{m-r}$.

Let us now show that the distance of binary Reed-Muller codes is at least 2^{m-r} by showing that the weight of any non-zero codewords in $\mathrm{RM}(m, r)$ is at least 2^{m-r}. Consider any non-zero polynomial $f(X_1, \ldots, X_m)$ of total degree at most r. We can write f as

$$f(X_1, \ldots, X_m) = X_1 X_2 \cdots X_s + g(X_1, \ldots, X_m)$$

where $X_1 X_2 \cdots X_s$ is a maximum degree term in f and $s \leq r$. Consider any assignment of values to the variables $X_{s+1}, \ldots, X_m$. After this assignment, the resulting polynomial on $X_1, \ldots, X_s$ is a non-zero polynomial, since the term $X_1 X_2 \cdots X_s$ cannot be cancelled. Therefore, for each of the 2^{m-s} possible assignment of values to the variables $X_{s+1}, \ldots, X_m$, the resulting polynomial is a non-zero polynomial.

When you have a non-zero polynomial, then there is always at least one assignment of values to its variables such that the polynomial does not evaluate to 0. Therefore, for each assignment $\alpha_{s+1}, \ldots, \alpha_m$ to the variables $X_{s+1}, \ldots, X_m$, there exists at least one assignment of values $\alpha_1, \ldots, \alpha_s$ to $X_1, \ldots, X_s$ such that $f(\alpha_1, \ldots, \alpha_m) \neq 0$. This implies that $\mathrm{wt}(\langle f(\alpha) \rangle_{\alpha \in \mathbb{F}_2^m}) = 2^{m-s} \geq 2^{m-r}$.

In summary, when the maximum degree r is constant, binary Reed-Muller codes have good distance, but a poor rate ($\approx m^r 2^{-m} \to 0$ for large m). Increasing the parameter r increases the rate of the code but also decreases the distance of the code at a faster pace. So there is no setting of r that yields a code with constant rate and constant distance.

In the following section, we introduce a family of binary codes that can be constructed efficiently and has both a good rate and a good distance simultaneously.

5 Concatenated codes

Concatenated codes were introduced by Forney in his doctoral thesis in 1966 [2]. In fact, Forney proved many wonderful properties about these codes; in this lecture we only give a brief overview of the definitions and key properties of concatenated codes.

The starting point in our search for binary codes with good rate and good distance is the idea that we have already seen codes with good rate and good distance when we have large alphabets: the distance of Reed-Solomon codes meets the Singleton bound, so they in fact are optimal codes. So let's start with Reed-Solomon codes and see if we can use them to construct a family of good binary codes.

We already saw in the last lecture a simple transformation for converting Reed-Solomon codes to binary codes. In this transformation, we started with a polynomial f of degree $k-1$ and evaluated it over $\alpha_1, \ldots, \alpha_m$ to obtain the values $f(\alpha_1), \ldots, f(\alpha_m) \in \mathbb{F}_{2^m}^n$. We then encoded each of the values $f(\alpha_i)$ in the binary alphabet with m bits.

The binary code obtained with the simple transformation has block length $N = nm$ and distance $D \geq d = n - k + 1$. This distance is not very good, since the lower bound on the relative distance $\frac{D}{N} \geq \frac{n-k+1}{nm}$ is quite weak. Still, the lower bound $D \geq d$ follows from a very simple analysis; one

may hope that a better bound – ideally of the form $D \geq \Omega(dm)$ – might be obtained with a more sophisticated analysis or by applying some neat trick (like, say, by encoding the bits in some clever basis). Unfortunately, that hope is not realizable: there is a nearly tight upper bound showing that with the simple transformation, the distance of the resulting binary code is at most $D \leq 2d$.

So if we hope to obtain a binary code with good distance from the Reed-Solomon code, we need to introduce a new idea to the transformation. One promising idea is to look closely at the step where we took the values from the field $\mathbb{F}_{2^m}$ and encoded them with m bits in the binary alphabet: instead of using the minimum number of bits to encode those elements in the binary alphabet, we could use more bits – say $2m$ bits – and use an encoding that adds more distance to the final code. That is indeed the idea used to obtain concatenated codes.

5.1 Binary concatenated codes

The concatenated code $C = C_{out} \diamond C_{in}$ is defined by two codes. The *outer* code $C_{out} \subset \Sigma_1^{n_1}$ converts the input message to a codeword over a large alphabet Σ_1, and the *inner* code $C_{in} \subset \Sigma_2^{n_2}$ is a much smaller code that converts symbols from Σ_1 to codewords over Σ_2. When $\Sigma_2 = \{0, 1\}$, the code C is a binary concatenated code.

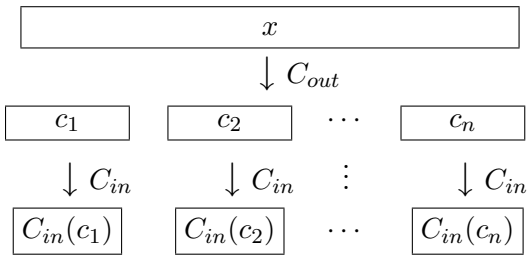


Figure 1: Binary concatenated codes.

A key observation in the definition of concatenated codes is that the inner code C_{in} is a small code, in the sense that it only needs one codeword for each symbol in Σ_1. The size of the alphabet Σ_1 is (typically) much smaller than the total number of codewords encoded by C, and will let us do a brute-force search for good inner codes in our construction of good concatenated codes. But first, let us examine the rate and distance parameters of general concatenated codes.

5.2 Facts of concatenated codes

The rate of the concatenated code $C = C_{out} \diamond C_{in}$ is

$$R(C) = \frac{\log |C_{out}|}{n_1 n_2 \log |\Sigma_2|} = \frac{\log |C_{out}|}{n_1 \log |\Sigma_1|} \cdot \frac{\log |\Sigma_1|}{n_2 \log |\Sigma_2|} = R(C_{out}) \cdot R(C_{in}),$$

where the last equality uses the fact that $|C_{in}| = |\Sigma_1|$.

The simple transformation of Reed-Solomon codes to binary codes used an inner code C_{in} with rate 1 (which did not add any redundancy). The rate equation $R(C) = R(C_{out}) \cdot R(C_{in})$ says that we can replace the trivial inner code with any other code and incur a rate cost proportional to the rate of C_{in}.

Let's now look at the distance of concatenated code. We do not get an exact formula for the distance of these codes, but a simple argument does give us a lower bound that will be sufficient to construct concatenated codes with good distance:

Proposition 13 *The distance of the concatenated code $C = C_{out} \diamond C_{in}$ satisfies*

$$\Delta(C) \geq \Delta(C_{out}) \cdot \Delta(C_{in}).$$

PROOF: Let x and y be two distinct messages. The distance property of the outer code guarantees that the encodings $C_{out}(x)$ and $C_{out}(y)$ will differ in at least $\Delta(C_{out})$ symbols. For each of the symbols where they differ, then the inner code will encode the symbols into codewords that differ in at least $\Delta(C_{in})$ places. □

The lower bound of Proposition 13 is not tight, and in general the distance of concatenated codes can be much larger. This may seem counter-intuitive at first: at the outer level, we can certainly have two codewords that differ in only $\Delta(C_{out})$ places, and at the inner level we can also have two different symbols in Σ_1 whose encodings under C_{in} differ in only $\Delta(C_{in})$ places. But the two events are not necessarily independent – it could be that when there are two codewords at the outer level that differ at only $\Delta(C_{out})$ symbols, then they must differ in a pattern that the inner code can take advantage of so that for those cases, the inner code does much better than its worst case.

In fact, a probabilistic argument shows that when the outer code is a Reed-Solomon code and the inner codes are "random projections" obtained by mapping the symbols of Σ_1 to codewords in Σ_2 with independently chosen random bases, then the resulting concatenated code reaches the Gilbert-Varshamov bound with high probability. (And thus has distance much larger than the lower bound suggested by Proposition 13.) This construction is randomized; it is an interesting problem to give a family of *explicit* codes for which the inequality of Proposition 13 is far from tight. (There are some codes called *multilevel concatenated codes* where the Zyablov bound can be improved, but this still falls well short of the GV bound.)

5.3 Constructing good concatenated codes

In this section, we construct a family of binary concatenated codes with good rate and good distance. Fix $0 < R < 1$ to be our target rate. We will build a code with rate R and distance as large as possible.

For our construction, take $C_{out} = [n, k, n-k+1]_{2^m}$ to be the Reed-Solomon code with block length $n = 2^m$. The rate of this outer code is $R_{out} = \frac{k}{n}$ and the relative distance of this code is $\delta_{out} = \frac{n-k+1}{n} \geq 1 - R_{out}$. Take C_{in} to be a binary linear code with parameters $[\frac{m}{r}, m, d]_2$, so that the rate of the inner code is $R(C_{in}) = r$. The rate of the concatenated code $C = C_{out} \diamond C_{in}$ is $R = R_{out} \cdot r$, so $R_{out} = \frac{R}{r}$.

We now have a partial construction. The outer code is the Reed-Solomon code, which we know is optimal so we're done with this part of the construction. The inner code, however, is not yet defined: we have only specified that we want C_{in} to be a linear code with rate r. For our concatenated code C to have good distance, we want the distance of C_{in} to be as large as possible.

The asymptotic Gilbert-Varshamov bound guarantees that there exists a linear code C_{in} with rate $r \geq 1 - h(\delta_{in})$. Rearranging the terms, this means that there is a code with rate r and distance $\delta_{in} \geq h^{-1}(1-r)$. So if we find an inner code that matches this distance bound, we obtain a concatenated code C with distance

$$\delta(C) \geq \delta_{out} \cdot \delta_{in} \geq (1 - R_{out}) \cdot h^{-1}(1-r) = \left(1 - \tfrac{R}{r}\right) \cdot h^{-1}(1-r).$$

The question remains: how can we find an inner code C_{in} with minimum distance $\delta_{in} \geq h^{-1}(1-r)$? Since C_{in} is a small code, so we can do a brute force search over the linear codes to find one with large distance.

We have to be a little careful in the algorithm that we use to search for C_{in}. A naïve searching algorithm simply enumerates all the possible generator matrices for C_{in} and checks the distance of each corresponding code. But there are $2^{m \times m/r} = n^{\log(n)/r}$ possible generator matrices G, so this search does not run in time polynomial in the block length of the code.

There is a more efficient algorithm for finding an inner code C_{in} with minimum distance d. The algorithm uses the greedy method to build a parity check matrix H such that every set of $d-1$ columns in H is linearly independent: Enumerate all the possible columns. If the current column is not contained by the linear span of any $d-2$ columns already in H, add it to H.

The greedy algorithm examines $2^{m/r-m} = n^{1/r-1}$ columns, and as long as $2^{m/r-m} > \sum_{i=0}^{d-2} \binom{n-1}{i}$, the process is also guaranteed to find a parity-check matrix H of distance d. So this method can be used to find a linear code that meets the Gilbert-Varshamov bound in time polynomial in the block length.

This completes our construction of a binary concatenated code with good rate and good distance. In the next section, we examine the best rate-distance trade-off obtained by optimizing the parameters of the concatenated code. But first, we mention one more useful property of the code we have constructed: it is a linear code.

Exercise 3 *Prove that the concatenated code C is linear over $\mathbb{F}_2$.*

5.4 Zyablov radius

In our construction of good concatenated codes, we are free to set the rate r of the inner code. Optimizing the value of r over all the choices that guarantee an overall rate of R for the concatenated code yields the following result.

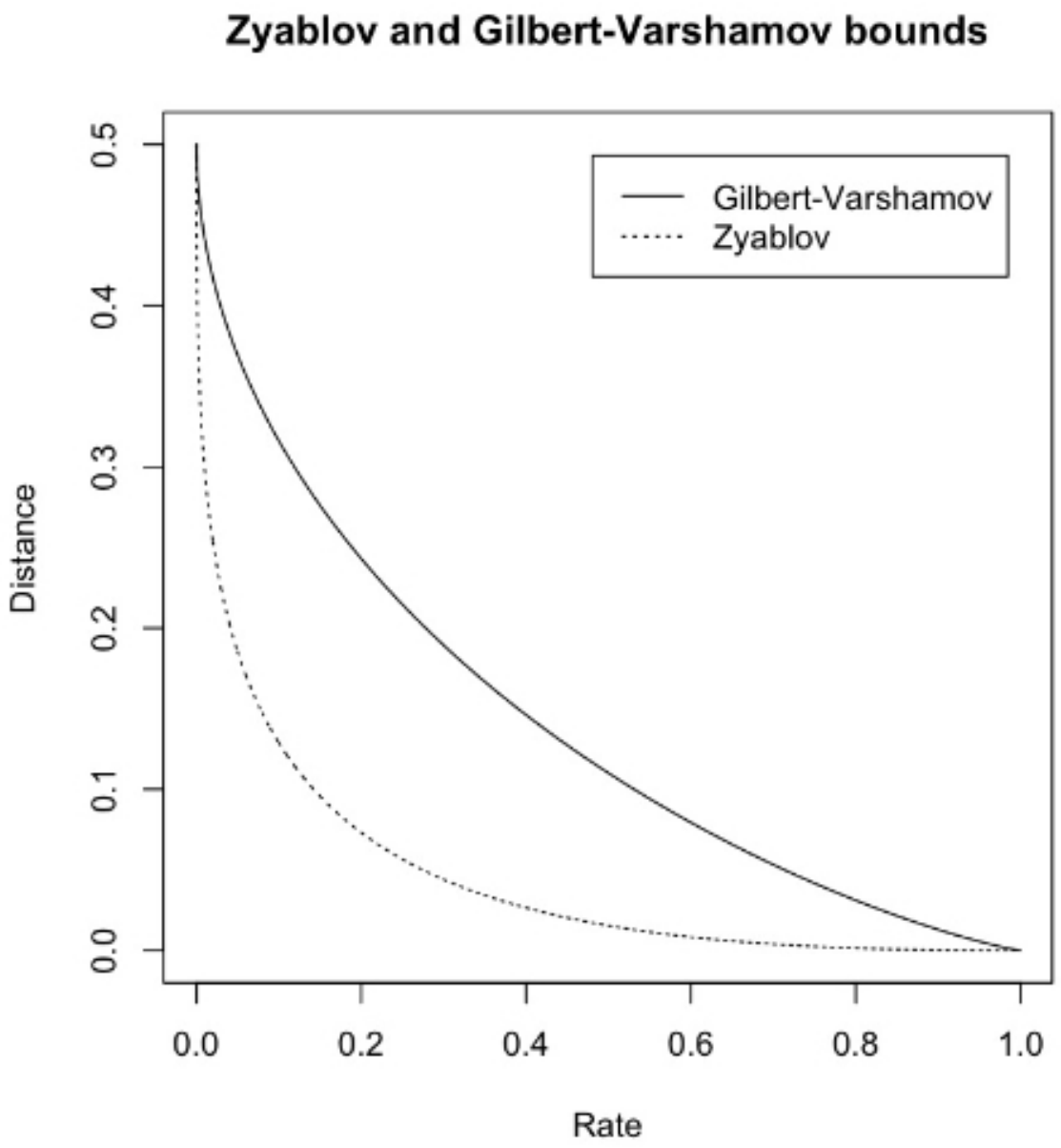
Theorem 14 *Let $R \in (0,1)$. Then it is possible to efficiently construct a code of rate R and distance*

$$\delta_{Zyablov}(R) = \max_{R \leq r \leq 1} (1 - \tfrac{R}{r}) h^{-1}(1-r).$$

The function $\delta_{Zyablov}$ is called the Zyablov trade-off curve, or sometimes the Zyablov bound, and is named after Zyablov, who first observed it in 1971 [9]. For any value of $R \in (0,1)$, the value of $\delta_{Zyablov}(R)$ is bounded away from 0, so we get the following corollary.

Corollary 15 *Asymptotically good codes of any desired rate $R \in (0,1)$ can be constructed in polynomial time.*

So how good is the resulting bound? Quite a bit weaker than the Gilbert-Varshamov bound, as the figure shows.



Another aspect of our construction of concatenated codes that is somewhat unsatisfactory is that whilt it it is constructed in polynomial time, it involves brute-force search for a code of logarithmic

block length. It would be nice to have an explicit formula or description of how the code looks like. From a complexity view point, we might want a linear code the entries of whose generator matrix we can compute in polylogarithmic time.

In the next lecture, we will see an asymptotically code that is constructedexplicitly without any brute-force search of smaller codes, and which further achievs the Zyablov trade-off between rate and relative distance for rates more than 0.31.

References

[1] Ray C. Bose and Dwijendra K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960. 4

[2] G. David Forney. *Concatenated codes*. MIT Press, 1966. 11

[3] Alexis Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2:147–156, 1959. 4

[4] David E. Muller. Application of boolean algebra to switching circuit design and to error detection. *IEEE Trans. on Computers*, 3:6–12, 1954. 7, 10

[5] Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IEEE Trans. on Information Theory*, 4:38–49, 1954. 7, 10

[6] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *J. Soc. Indust. Appl. Math.*, 8(2):300–304, 1960. 3

[7] Jack T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. 8

[8] Richard Zippel. Probabilistic algorithms for sparse polynomials. *Symbolic and Algebraic Computation*, Springer LNCS 72:216–226, 1979. 9

[9] Victor V. Zyablov. An estimate of the complexity of constructing binary linear cascade codes. *Probl. Peredachi Inf.*, 7(1):5–13, 1971. 14

Introduction to Coding Theory **CMU: Spring 2010**

Notes 7: Justesen codes, Reed-Solomon and concatenated codes decoding

March 2010

Lecturer: V. Guruswami *Scribe: Venkat Guruswami & Balakrishnan Narayanaswamy*

1 Review - Concatenated codes and Zyablov's tradeoff

In the last class we saw (Theorem 14) that it is possible to *efficiently* construct an (asymptotically good) concatenated code of rate R with distance meeting the Zyablov trade-off between relative distance δ and rate R:

$$\delta_{\text{Zyablov}}(R) = \max_{R \le r \le 1} (1 - \frac{R}{r}) h^{-1}(1-r) \tag{1}$$

Given a specified rate target R, the construction involved a brute force search for the inner code C_{in} of rate r that met the Gilbert-Varshamov bound and had relative distance $h^{-1}(1-r)$. Once such an inner code was found, it was used to encode each symbol of the outer Reed-Solomon code.

This use of a search step left open the question of constructing fully explicit codes, with no brute-force search for a smaller code, with similar trade-offs.

2 Justesen's code

Justesen [9] provided an explicit concatenated code construction that achieved the Zyablov tradeoff over a range of rates (approximately rates $R \ge .31$) and was aymptotically good for any desired rate $R \in (0,1)$. The construction was based on the following insights:

1. A inner codes C_{in}^i do not have to be the same
2. It is sufficient if most (a fraction $1 - o(1)$) of the inner codes meet the Gilbert-Varshamov bound.

How can we exploit these insights ? We fix the outer code C_{out} as the $[n = 2^m - 1, k, n-k+1]_{2^m}$ Reed Solomon code $\text{RS}_{\mathbb{F},\mathbb{F}^*}[n,k]$ over the field $\mathbb{F} = \mathbb{F}_{2^m}$. Each symbol in the resulting code can be mapped (bijectively) into a binary sequence of length m using an $\mathbb{F}_2$-linear map $\sigma : \mathbb{F}_{2^m} \to \mathbb{F}_2^m$. We code each sequence i (for $i = 1, \ldots, n$) with a different binary rate half inner code C_{in}^i. That is each C_{in}^i maps the binary sequence of length m to a binary sequence of length $2m$. Suppose that most (at least $(1 - o(1))n$) of the inner codes C_{in}^i have a relative distance $\delta_i \ge \delta_g$.

This concatenated code has rate $\frac{R}{2} = \frac{k}{2n}$ and relative distance at least $(1 - 2R - o(1))\delta_g$. If most of the inner codes (almost) meet the GV bound for rate 1/2 codes, so that

$$\delta_g = \delta_{GV}(1/2) - o(1) = h^{-1}(1/2) - o(1) \ , \tag{2}$$

then the overall code has relative distance given by

$$\delta(R) = (1-2R)h^{-1}(1/2) - o(1) \ . \tag{3}$$

In the Exercise 1 below, we will construct a family of codes such that all but a small fraction of them asymptotically meet the GV bound.

Exercise 1 *Show that there is a family $\mathcal{F}$ of $[2m, m]_2$ binary linear codes such that the following hold*

1. $|\mathcal{F}| = 2^m - 1$
2. *Most of the codes $C \in \mathcal{F}$ have relative distance at least $h^{-1}(1/2) - o(1)$*

*(***Hint***: Consider the code family from HW 1, Question 5.b. For $\alpha \in \mathbb{F}_{2^m}$, $\alpha \neq 0$, consider the map $L_\alpha : \mathbb{F}_2^m \to F_2^{2m}$ defined as*

$$L_\alpha(\mathbf{x}) = (\mathbf{x}; \sigma(\alpha \bullet \sigma^{-1}(\mathbf{x}))) \tag{4}$$

and the family of codes $\mathcal{F} = \{L_\alpha : \alpha \neq 0\}$. We showed that there exist codes in this family that asymptotically meet the GV bound. The same argument can actually show that this is true for most codes in the family.)

Using these codes as the different inner codes C_{in}^i's with outer code $\mathrm{RS}_{\mathbb{F},\mathbb{F}^*}[n,k]$ gives us the following result:

Lemma 1 *There are explicit binary linear codes of rate $R < 1/2$ and relative distance*

$$(1-2R)h^{-1}(\frac{1}{2}) - o(1) \ . \tag{5}$$

Remark 2 *In the above concatenated code, the message is a polynomial $f \in \mathbb{F}_{2^m}[X]$ of degree at most $k-1$ and it is encoded by the evaluations of both $f(X)$ and $Xf(X)$ at the nonzero elements of the field, which are then expressed as elements of $\mathbb{F}_2^m$ as per some basis.*

We see that the above only gives codes of overall rate less than $1/2$ (since the inner code itself has rate only $1/2$). For larger rates we modify the construction in Exercise 1 to construct a small family of codes of any desired rate $r \in (1/2, 1)$ that meet the GV bound.

Exercise 2 *For any $1 \leq s \leq m$, show that there is a family $\mathcal{F}$ of $[m+s, m]_2$ binary linear codes such that the following hold*

1. $|\mathcal{F}| = 2^m - 1$
2. *Most of the codes $C \in \mathcal{F}$ have relative distance at least $h^{-1}(s/(m+s)) - o(1)$*

(**Hint**: *Modify the code family from Exercise 1 as follows. Define* $\sigma' : \mathbb{F}_{2^m} \to F_2^s$ *such that* $\sigma'(\mathbf{x}) =$ *first s bits of* $\sigma(\mathbf{x})$. *For* $\alpha \in \mathbb{F}_{2^m}$, $\alpha \neq 0$, *consider the map* $L'_\alpha : \mathbb{F}_2^m \to F_2^{m+s}$ *defined as*

$$L'_\alpha(\mathbf{x}) = (\mathbf{x}; \sigma'(\alpha \bullet \sigma^{-1}(\mathbf{x}))) \tag{6}$$

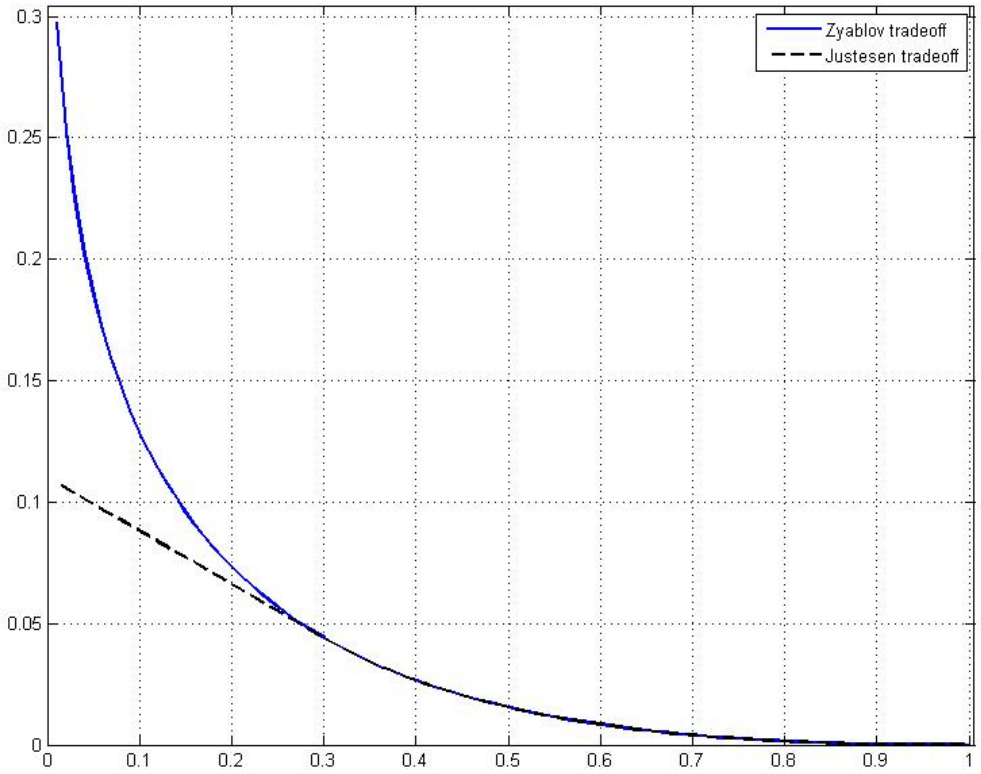
and the family of codes $\mathcal{F} = \left\{ L'_\alpha : \alpha \neq 0 \right\}$. *Use volume arguments to show that most codes in this family have relative distance* $\geq h^{-1}(\frac{s}{s+m}) - o(1)$.

By concatenating the outer Reed-Solomon code of length $2^m - 1$ with all the different codes in the above family, we get the following:

Theorem 3 *For any* $R \in (0, 1)$, *there are explicit binary lineat codes of rate at least* R *and relative distance at least*

$$\delta_{\text{Justesen}}(R) = \max_{r \geq \max(\frac{1}{2}, R)} (1 - \frac{R}{r}) h^{-1}(1 - r) - o(1) \tag{7}$$

We compare the two bounds constructive results: the Zyablov tradeoff (Eq. 1) and the explicit construction Justesen tradeoff (Eq. 7) in the figure below.



Exercise 3 *Show that the* $\delta_{Zyablov}(R) = \delta_{Justesen}(R)$ *for* $R \geq .31$.

Hint: *Differentiating the expression* $(1 - R/r)h^{-1}(1 - r)$ *w.r.t* r*, one finds that the* r *maximizing this expression (and thus leading to the Zyablov trade-off in (1)) satisfies*

$$R = \frac{r^2}{1 + \log_2 (1 - h^{-1}(1 - r))} \ .$$

For $R \geq 0.31$*, the solution* r *to the above equation lies in the range* $[1/2, 1]$*, and thus the Zyablov bound can be met by the Justesen construction.*

2.1 Meeting the Zyablov trade-off at lower rates

The Justesen construction in the previous section used a good ensemble of codes of rate $r = \frac{1}{2}$ for the inner code, with the size of the ensemble $|\mathcal{F}| < 2^m$. The reason that the same construction does not work for rates $R < .31$ is that we do not know small enough inner code ensembles with rate $r < \frac{1}{2}$ where most of the codes are meet the GV bound.

For example, for $R \approx 0.15$, the optimal choice of the inner rate r in the Zyablov bound is $\approx 1/3$. Consider the following map, similar to the ones used in Exercises 1 and 2,

$$L_{\alpha_1,\alpha_2}(\mathbf{x}) = (\mathbf{x}; \sigma(\alpha_1 \bullet \sigma^{-1}(\mathbf{x})); \sigma(\alpha_2 \bullet \sigma^{-1}(\mathbf{x}))) \tag{8}$$

and the family of codes defined by this map for all pairs of nonzero field elements α_1, α_2. One can show, via a similar counting argumemt, that this is a family of at most 2^{2m} codes where most of the codes meet the GV bound and have relative distance $\delta = h^{-1}(\frac{2}{3}) - o(1)$.

To use these rate $1/3$ codes in a Justesen-like construction, we need an outer code over alphabet $\mathbb{F}_{2^m}$ that has block length about 2^{2m}, and which nearly meets the Singleton bound. Reed-Solomon codes are limited to a block length of 2^m and thus are not long enough. The solution around this predicament is provided by algebraic-geometric (AG) codes. As we briefly mentioned earlier, AG codes over $\mathbb{F}_q$ are a generalization of RS curves based on evaluation of functions with few "poles" at the rational points of an appropriately chosen algebraic curve which has $\gg q$ points with coordinates in $\mathbb{F}_q$. Shen [12] proposed an explicit family of AG codes over $\mathbb{F}_q$ which can have block length at least q^c for any desired c, and whose relative distance as a function of the rate R is $1 - R - o_q(1)$. Using these codes as outer codes in place of RS codes and the appropriate extension of the above rate $1/3$ code ensemble in a Justesen-like construction, he was able to give an explicit construction achieving the Zyablov trade-off for the entire range of rates $R \in (0, 1)$. Discussing the details of these AG codes are beyond the scope of this course, and the interested reader can find the details in the original paper [12].

3 Decoding algorithms

We now turn to algorithmic aspects of error-correction. We first consider the (relatively benign) erasure channel and then move on to channels that arbitrarily corrupt a constant fraction of the transmitted codeword symbols. We will see that Reed-Solomon codes admit efficient decoding algorithms matching the combinatorial bounds possible by virtue of its minimum distance.

3.1 Erasure decoding of linear codes

Consider a $[n, k, d]_q$ code with message $x \in \mathbb{F}_q^k$ and corresponding codeword $y = Gx \in \mathbb{F}_q^n$. Suppose a subset $S \subseteq \{1, \ldots, n\}$ is received (uncorrupted) and the rest of the positions are erased. The location of the erasures are known at the receiving end. Decoding the message involves solving the linear system

$$G_S x = y_S \tag{9}$$

We reorder the indices so that the first $|S|$ entries in y are the uncorrupted entries, resulting in the following matrix system,

$$\left[\begin{array}{c} G_S \\ \hline G_{\overline{S}} \end{array}\right] \left[\begin{array}{c} \\ x \\ \\ \end{array}\right] = \left[\begin{array}{c} y_S \\ \hline y_{\overline{S}} \end{array}\right] \tag{10}$$

As long as the rank of G_S is k, the solution to $G_S x = y_S$ is uniquely determined. When the number of erasures is less than d, i.e., $|S| > n - d$, the distance property of the code implies that G_S has full rank. Thus one can correct any pattern of $d - 1$ or fewer erasures by solving the linear system $G_S x = y_S$ in $O(n^3)$ time. However, specific structured G matrices can allow much faster solution of the linear system (even linear time in certain cases as we will see later on when discussing expander codes), leading to faster erasure recovery algorithms.

3.2 Erasure decoding of Reed Solomon Codes

We recall the interpretation of Reed Solomon codes from the previous lecture,

Definition 4 *Reed-Solomon codes*
For integers $1 < k < n$, field $\mathbb{F}$ of size $|\mathbb{F}| > n$, and a set $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$, we define the Reed-Solomon code

$$RS_{\mathbb{F},S}[n, k] \quad = \quad \{(p(\alpha_1), \ldots, p(\alpha_n)) \in \mathbb{F}^n | p \in \mathbb{F}[X] \text{ a polynomial of degree} \leq k - 1\} \tag{11}$$

To encode a message $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}^k$, we interpret the polynomial as

$$p(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-1} \in \mathbb{F}[X] \tag{12}$$

Suppose a codeword from a Reed Solomon code is transmitted over an erasure channel and all but t symbols are erased. Then the decoder must reconstruct the message m from t pairs of values $\{(\alpha_1, f(\alpha_1)), \ldots, (\alpha_t, f(\alpha_t))\}$. Since the polynomial $p(X)$ is a degree $k-1$ polynomial it is uniquely determined by its value at any $t \geq k$ points. This can be done using FFTs in $n \log^{O(1)} n$ time or using polynomial interpolation in $O(n^2)$ time. Suppose $t = k$ and nonerased locations correspond to $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. Note that if we define the polynomials p_j for $1 \leq j \leq k$

$$p_j(X) = \prod_{i=1, i \neq j}^{k} \frac{X - \alpha_i}{\alpha_j - \alpha_i} \ , \tag{13}$$

the interpolated polynomial is then

$$f(X) = \sum_{j=1}^{k} f(\alpha_j) p_j(X) \ . \tag{14}$$

The number $n-k$ of erasures corrected by RS codes is optimal, since to have any hope of recovering the k message symbols, one must receive at least k symbols at the receiving end.

3.3 Decoding Reed-Solomon codes from errors

We now turn to the more challenging problem of decoding Reed-Solomon codes from worst-case errors. Specifically, we would like to decode the RS code RS$[n, k]$ up to $\tau = \lfloor \frac{n-k}{2} \rfloor$ errors. (Recall that the code has distance $n - k + 1$, so the correct codeword is uniquely determined as the closest codeword to the received word if up to τ errors occur.)

Suppose a polynomial $f \in \mathbb{F}_q[X]$ of degree $k-1$ is encoded as the RS codeword $(f(\alpha_1), \ldots, f(\alpha_n))$ and transmitted, but it is received as the noisy word $y = (y_1, \ldots, y_n)$ satisfying $y_i \neq f(\alpha_i)$ for at most τ values of i. The goal is to recover the polynomial $f(X)$ from y.

We now discuss an algorithm due to Welch and Berlekamp [15] for solving this problem. (The streamlined and simplified presentation discussed here is due to Gemmell and Sudan [7].) Note that if we knew the location of the errors, i.e., the set $E = \{i \mid y_i \neq f(\alpha_i)\}$, then the decoding is easy, as we can erase the erroneous and interpolate the polynomial on the rest of the locations.

To this end, let us define the error locator polynomial (which is unknown to the decoder):

$$E(X) = \prod_{f(\alpha_i) \neq y_i} (X - \alpha_i) \tag{15}$$

The degree of $E(X)$ is $\leq \tau$. Clearly $E(X)$ has the property that for $1 \leq i \leq n$, $E(\alpha_i) y_i = E(\alpha_i) f(\alpha_i)$. Define the polynomial

$$N(X) = E(X) F(X) \ , \tag{16}$$

which has degree at most $\tau + k - 1$. Now the bivariate polynomial

$$P(X, Y) = E(X) Y - N(X) \tag{17}$$

satisfies $P(\alpha_i, y_i) = 0, \forall i$. We will use the existence of such a P to find a similar bivariate polynomial from which we can find $f(X)$.

Formally, the algorithms proceeds in two steps.

Step 1 : Find a non-zero polynomial $Q(X, Y)$ such that,

1. $Q(X, Y) = E_1(X) Y - N_1(X)$
2. $deg\ E_1 \leq \tau$ and $deg\ N_1 \leq \tau + k - 1$
3. $Q(\alpha_i, y_i) = 0, \forall i$

Step 2 : Output $\frac{N_1(X)}{E_1(X)}$ as $f(X)$

Proposition 5 *A non-zero solution Q to Step 1 exists.*

PROOF: Take $E_1 = E$, $N_1 = N$. $\square$

Proposition 6 *Any solution (E_1, N_1) must satisfy $\frac{N_1}{E_1} = f$.*

PROOF: Define the polynomial

$$R(X) = E_1(X)f(X) - N_1(X) \tag{18}$$

Fact 1: *deg* $R \leq \tau + k - 1$. This follows immediately from the conditions imposed on the degree of E_1 and N_1.
Fact 2: R has at least $n - \tau$ roots. Indeed, for each locations i that is not in error, i.e., $f(\alpha_i) = y_i$, we have $R(\alpha_i) = Q(\alpha_i, y_i) = 0$.

Using the above two facts, we can conclude that if $n - \tau > \tau + k - 1$, then R is identically 0, which means that $f(X) = N_(X)/E_1(X)$. Since $\tau = \lfloor \frac{n-k}{2} \rfloor$, this condition on τ is met, and we conclude that the algorithm successfully recovers $f(X)$. $\square$

We now argue that the above algorithm can be implemented in polynomial time. Clearly the second step is easy. For the first interpolation step, note that it can be solved by finding a non-zero solution to a homogeneous linear system with unknowns being the coefficients of the polynomials N_1, E_1, and n linear constraints $Q(\alpha_i, y_i) = 0$. Since we guaranteed the existence of a nonzero solution, one can find some nonzero solution by Gaussian elimination in $O(n^3)$ field operations.

The interpolation step is really rational function interpolation and near-linear time algorithms are known for it. Also one can do fast polynomial division in $n \log^{O(1)} n$ field operations. Thus overall the algorithm can also be implemented in near-linear time.

We conclude by recording the main result concerning decoding RS codes up to half the distance.

Theorem 7 *There is a polynomial time decoding algorithm for an $[n, k]_q$ Reed-Solomon code that can correct up to $\lceil \frac{n-k}{2} \rceil$ worst-case errors.*

Thus, for a given rate R we can correct a $\frac{1-R}{2}$ fraction of errors (using RS codes and the above algorithm) which is the best possible by the Singleton bound. Later in the course we will look at list decoding algorithms that can improve on these parameters by allowing the decoder to output a (small) list of candidate codewords.

The primary disadvantage of RS codes are that they are defined over very large alphabets (of size at least the codeword length). We will soon see efficiently decodable binary codes constructed via code concatenation. But next we see a different algorithm for decoding RS codes up to half the distance, which was historically the first such algorithm.

4 Peterson Algorithm for decoding RS codes

The Peterson Algorithm from 1960 [11] is another algorithm to decode Reed Solomon codes up to half the minimum distance. One interesting feature of its discovery is that it is a non-trivial polynomial time algorithm for a non-trivial problem proposed *before* polynomial time was formalized as the theoretical standard for efficient computation!

The Peterson Algorithm works with the parity check view of RS codes which we recall here

Definition 8 (Parity-check characterization) *For integers $1 \leq k < n$, a field $\mathbb{F}$ of size $|\mathbb{F}| = q = n + 1$, a primitive element $\alpha \in \mathbb{F}_q^*$, and the set $S = \{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, the $[n, k, n-k+1]_q$ Reed-Solomon code over $\mathbb{F}$ with evaluation set S is given by*

$$\begin{aligned} \mathrm{RS}_{\mathbb{F},S}[n,k] &= \{(c_0, c_1, ..., c_{n-1}) \in \mathbb{F}^n | c(X) = c_0 + c_1 X + \ldots + c_{n-1} X_{n-1} \\ &\quad satisfies\ c(\alpha) = c(\alpha^2) = \ldots = c(\alpha^{n-k}) = 0\} \end{aligned}$$

Suppose a codeword $c \in \mathrm{RS}_{\mathbb{F},S}[n,k]$ is transmitted and an error vector $e \in \mathbb{F}^n$ of Hamming weight at most

$$\tau = \left\lfloor \frac{n-k}{2} \right\rfloor$$

is added to c, so that it is received as $y = c + e = (y_0, y_1, \ldots, y_{n-1})$. The goal is to efficiently recover c (or the polynomial $f \in \mathbb{F}[X]$ of degree $< k$ that it encodes).

For $l \in \{1, \ldots, n-k\}$, we can compute the syndrome

$$S_l = \sum_{j=0}^{n-1} y_j \alpha^{lj} = c(\alpha^l) + \sum_{j=0}^{n-1} e_j \alpha^{lj} = \sum_{j=0}^{n-1} e_j \alpha^{lj} \tag{19}$$

since $c(\alpha^l) = 0$ for $1 \leq l \leq n-k$. Let us define the *syndrome polynomial* $S(X)$

$$S(X) = \sum_{l=1}^{n-k} S_l X^{l-1} \ . \tag{20}$$

This polynomial and its properties play a key role in the development and analysis of the decoding algorithm. Note that the syndrome polynomial can be efficiently computed from the received word.

We define $T \subseteq \{0, 1, \ldots, n-1\}$ be the set of error locations, i.e., $T = \{i \mid e_i \neq 0\}$. Define the Error Locator Polynomial as follows (note that T and the error locator polynomial are *not* known at the decoder, and computing them is at the heart of the decoding task).

$$E(X) = \prod_{j \in T} (1 - \alpha^j X) \tag{21}$$

This polynomial is defined so that it has roots exactly at α^{-j} for those indices j where the received vector y is in error. The degree of E is $|T| \leq \tau$.

We now simplify the expression for the syndrome polynomial:

$$\begin{aligned} S(X) &= \sum_{l=1}^{n-k} S_l X^{l-1} \\ &= \sum_{l=1}^{n-k} X^{l-1} \sum_{j \in T} e_j \alpha^{lj} \\ &= \sum_{j \in T} e_j \alpha^j \sum_{l=1}^{n-k} X^{l-1} \alpha^{j(l-1)} \\ &= \sum_{j \in T} e_j \alpha^j \left(\frac{1 - (\alpha^j X)^{n-k}}{1 - \alpha^j X} \right) \end{aligned}$$

Hence

$$E(X)S(X) = \Big(\prod_{j \in T} (1 - \alpha^j X) \Big) S(X) = \sum_{j \in T} e_j \alpha^j \big(1 - (\alpha^j X)^{n-k} \big) \prod_{i \in T, i \neq j} (1 - \alpha^i X) \ .$$

Defining

$$\Gamma(X) = \sum_{j \in T} e_j \alpha^j \prod_{i \in T, i \neq j} (1 - \alpha^i X)$$

leads us to the equation (often called the *Key Equation*)

$$E(X)S(X) \equiv \Gamma(X) \pmod{X^{n-k}} \tag{22}$$

We would like to use the above equation to solve for $E(X)$. Once we do that we can find the roots of E to determine the error locations, and then find the message polynomial by interpolation using the non-erroneous locations. Note that in the Key Equation, we know $S(X)$ but do not know either $E(X)$ or $\Gamma(X)$. However, we observe the following property of $\Gamma(X)$: it has degree at most $\tau - 1$. Therefore the Key Equation implies that the coefficients of the X^j for $\tau \leq j \leq n-k-1$ in $E(X)S(X)$ all equal 0. The decoder will use this to find the coefficients of $E(X)$ by solving a homegeneous linear system. Specifically, the algorithm will solve for unknowns a_i where $E(X) = 1 + \sum_{i=1}^{\tau} a_i X^i$.

The decoding algorithm proceeds as follows. (Since the system solved by the decoder could have multiple solutions, we will denote the candidate error locator polynomial found by the decoder as E_1 and later prove that E_1 must be divisible by E, which suffices to locate all the erroneous positions.)

Step 1: Compute the syndrome polynomial $S(X)$.

Step 2: Solve for $\{a_i\}_{i=1}^{\tau}$ to find $E_1(X) = 1 + \sum_{i=1}^{\tau} a_i X^i$ such that the coefficients of X^j in $E_1(X)S(X)$ all equal 0 for $j = \tau, \tau+1, \ldots, n-k-1$. (This is a system of $n-k-\tau$ homogeneous linear equations in the a_i's.)

Step 3: Find all roots of $E_1(X)$ using brute force search (we also later describe an optimization called Chien search which is faster for hardware implementations.) Suppose the roots are $\alpha^{-i_1}, \ldots, \alpha^{-i_l}$ for some $l \leq \tau$.

Step 4: Erase the locations $\{i_1, i_2, \ldots, i_l\}$ in the received word y and then interpolate a degree $< k$ polynomial $f(X)$ through the unerased positions. If this is not possible, declare decoding failure. Otherwise, return $f(X)$ as the message polynomial.

We now prove that assuming at most τ errors occurred, the polynomial $E_1(X)$ found in Step 2 will be divisible by $E(X)$ and hence all the error locations will be roots of $E_1(X)$ as well.

Proposition 9 *The error locator polynomial $E(X)$ divides the polynomial $E_1(X)$ found by the algorithm.*

PROOF: Recall that $E(X) = \prod_{j \in T}(1 - \alpha^j X)$. In general polynomials may not have a multiplicative inverse modulo X^{n-k} (for example the polynomial X has no inverse $\mod X^{n-k}$). However, $E(X)$ has an inverse modulo X^{n-k}, namely

$$E^{-1}(X) = \prod_{j \in T}(1 + \alpha^j X + \ldots + (\alpha^j X)^{n-k-1}).$$

Therefore we can solve for the syndrome (modulo X^{n-k}) as

$$S(X) \equiv \Gamma(X) E^{-1}(X) \pmod{X^{n-k}} . \tag{23}$$

Let $\Gamma_(X)$ be the polynomial of degree at most $\tau - 1$ such that

$$E_1(X) S(X) \equiv \Gamma_1(X) \pmod{X^{n-k}} . \tag{24}$$

Combining (23) and (24), we get

$$E_1(X)\Gamma(X) \equiv E(X)\Gamma_1(X) \pmod{X^{n-k}} .$$

Both the polynomials $E_1(X)\Gamma(X)$ and $E(X)\Gamma_1(X)$ have degree at most $\tau + (\tau - 1) = 2\tau - 1 \leq n - k - 1$, therefore we can conclude that they are in fact equal as polynomials:

$$E_1(X)\Gamma(X) = E(X)\Gamma_1(X) . \tag{25}$$

We now note that $\gcd(E(X), \Gamma(X)) = 1$. This follows from the following two observations: (i) the elements α^{-j} for $j \in T$ are *all* the roots of $E(X)$, and (ii) by definition of $\Gamma(X)$, it follows that $\Gamma(\alpha^{-j}) = e_j \prod_{i \in T; i \neq j}(\alpha^j - \alpha^i) \neq 0$. By (25) we know that $E(X)$ divides $E_1(X)\Gamma(X)$, and since $\gcd(E(X), \Gamma(X)) = 1$, we conclude that $E(X)$ must divide $E_1(X)$. □ Since both E and E_1 have constant term equal to 1, it follows from Proposition 9 that if instead of finding a degree τ polynomial $E_1(X)$, we try to find a polynomial of degree $1, 2, \ldots, \tau$ successively till the algorithms succeeds, then in fact we will recover the error locator polynomial $E(X)$ as $E_1(X)$. This is an alternate way to implement the algorithm, which will in fact be faster when the number of errors is much smaller than τ.

Corollary 10 *If we find $E_1(X)$ of the smallest degree that satisfies the Key Equation E_1 (22) then, $E_1(X) = E(X)$.*

Remark 11 *We can find the roots of $E_1(X)$ in Step 3 of the Peterson Algorithm by brute force search over all elements of the field checking if $E_1(\alpha^j) = 0$ for $j = 0, 1, 2, \ldots, n-1$. An optimization called "Chien search" leads to more practical hardware implementations. Chien search [4] is based on the following observation (our description is from [16])*

> *If $E_1(X)$ has degree τ, then $E_1(\alpha^{i+1})$ can be computed from $E_1(\alpha^i)$ by τ multiplications of variables with **constants** as opposed to τ multiplications of variables with **variables** needed in the brute force search.*

This is because of the following relationship between the evaluations $E_1(\alpha^i)$ and $E_1(\alpha^{i+1})$. If

$$\begin{aligned} E_1(\alpha^i) &= e_0 + e_1\alpha^i + \ldots + e_\tau\alpha^{i\tau} = \gamma_{0,i} + \gamma_{1,i} + \ldots + \gamma_{\tau,i} \ , \ \textit{then} \\ E_1(\alpha^{i+1}) &= e_0 + e_1\alpha^{i+1} + \ldots + e_\tau\alpha^{(i+1)\tau} = \gamma_{0,i} + \gamma_{1,i}\alpha + \ldots + \gamma_{\tau,i}\alpha^\tau \ . \end{aligned}$$

Exercise 4 (Forney's formula) *Once the error locations T are computed, show that the error values are given by the following formula. For every $j \in T$,*

$$e_j = \frac{-\alpha^j \Gamma(\alpha^{-j})}{E'(\alpha^{-j})} \tag{26}$$

where $E'(X)$ is the derivative of $E(X)$. This can be used in place of the interpolation step for recovering the codeword.

Remark 12 *For binary BCH codes, once we know the exact error locations (say by finding a solution $E_1(X)$ with smallest degree), we can just flip those locations to get the true codeword, and no separate step is needed to compute the error values.*

Remark 13 (Complexity the algorithm) *The naive implementation of the algorithm takes cubic time, with the dominant step being solving the linear system to find $E_1(X)$. Sugiyama, Kasahara, Hirasawa and Namekawa [14] gave a quadratic time implementation using Euclid's algorithm for solving the Key equation.*

Berlekamp [2] gave an iterative quadratic time implementation of the solution to the key equation, which was later generalized by Massey [10] to give an algorithm for finding the shortest linear feedback shift register (LFSR) that generates a given sequence. This method is now referred to as the Berlekamp-Massey algorithm and is widely used in practice for decoding Reed-Solomon codes. Blahut [3] gave a method to accelerate the Berlekamp-Massey algorithm through recursion, resulting in a near-linear ($n \log^{O(1)} n$) time algorithm for decoding RS codes.

5 Decoding concatenated codes

5.1 A Naive algorithm

As we noted at the beginning of this class, RS codes require large alphabets and are therefore not suited for applications where we need, for example, binary codes. We also saw in previous

lectures the method of code concatenation which can be used to reduce the alphabet size to while preserving a good rate vs distance trade-off. We now consider efficient decoding algorithms for decoding concatenated codes with outer Reed-Solomon code. We start with a naive decoder, and later improve upon it.

Suppose that x is the message to be transmitted, coded with a Reed-Solomon outer code C_{out} to obtain symbols $c_1, \ldots, c_n$. Each of these is coded with an inner code C_{in} resulting in the transmitted codeword, whose i'th block is $C_{in}(c_i)$. As before assume that the outer code has distance D and inner codes have distance d. In theory, we should be able to correct up to $\frac{dD}{2}$ errors since the distance of the concatenated code is at least dD.

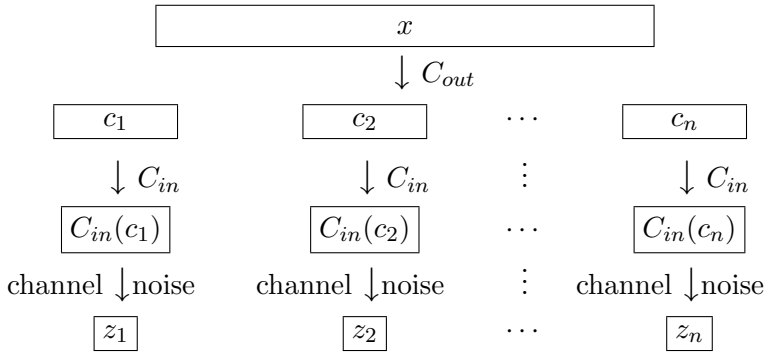


Figure 1: Decoding concatenated codes.

Suppose (due to errors) we receive z_i for $i = 1, \ldots, n$ instead of $C_{in}(c_i)$. Note that the total number of errors introduced equals $\sum_{i=1}^{n} \Delta(C_{in}(c_i), z_i)$. A simple decoder tries to reverse the encoding as follows.
Step 1 : Find a_i such that $\Delta(C_{in}(a_i), z_i)$ is minimized.
Step 2 : Decode $(a_1, \ldots, a_n)$ as per the decoder for the outer code C_{out} which can correct $< D/2$ errors.

Lemma 14 *The above algorithm recovers the correct codeword if the number of errors is less than* $\frac{dD}{4}$.

PROOF: Note that for each i for which $< \frac{d}{2}$ errors occur in i^{th} block, the inner decoding succeeds and we have $a_i = c_i$. If less than $\frac{dD}{4}$ errors occur in total, then the number of blocks with at least $d/2$ errors is less than $\frac{D}{2}$. Therefore the string $(a_1, \ldots, a_n)$ passed to the outer decoder differs from the true outer codeword $(c_1, c_2, \ldots, c_n)$ in $< D/2$ locations. The outer decoder then succeeds is recovering $(c_1, \ldots, c_n)$. □

5.2 Generalized Minimum Distance (GMD) Decoding of concatenated codes : Randomized version

Forney [6] gave a better algorithm for decoding that can correct a number of errors $< \frac{dD}{2}$ errors. This method is called Generalized Minimum Distance (GMD) decoding, and is based on using an errors-and-erasures decoder for the outer code. Recall that the Welch-Berlekamp algorithm that we discussed for Reed-Solomon can handle τ errors and s erasures as long as $2\tau + s < D$.

The naive algorithm discussed above is sub-optimal because all the guesses a_i from the inner decoder are treated on equal footing regardless of how far their inner encodings were from z_i. Intuitively, the outer decoder should place higher confidence in symbols whose inner encodings are close to z_i. This is achieved by assigning a measure of confidence to each a_i and erasing symbols whose confidence is below some threshold. Running such an algorithm for various choices of the threshold leads gives the GMD algorithm.

We first present a randomized version of GMD decoding that is easier and more intuitive to analyze, and then discuss how the algorithm can be derandomized to give a deterministic version. Below is the algorithm description.

Step 1: Decode z_i to a_i as before by finding the a_i that minimizes $\Delta(C_{in}(a_i), z_i)$

Step 2: Set $w_i = \min[\Delta(C^i_{in}(a_i), z_i), \frac{d}{2}]$.

Step 3: With probability $\frac{2w_i}{d}$ set a_i to be an erasure symbol '?'.

Step 4: Run the errors-and-erasures decoder (say Welch-Berlekamp in the case of Reed-Solomon codes) on the resulting sequence of a_is and ?s.

We assume that the total number of errors, $\sum_{i=1}^{n} \Delta(C_{in}(c_i), z_i) < dD/2$ and study the conditions under which this algorithm successfully decodes to the true codeword. Define τ as the number of errors and s as number of erasures in the outer codeword after the (randomized) inner decoding.

Lemma 15 *$\mathbb{E}[2\tau + s] < D$ where the expectation is taken over the random choices of erasures.*

PROOF: Let Z_i^{err} and $Z_i^{erasures}$ be indicator random variables for the occurrence of an error and declaration of an erasure respectively in the decoding of the i'th block z_i. We have

$$\tau = \sum_{i=1}^{n} Z_i^{err} \qquad s = \sum_{i=1}^{n} Z_i^{erasures} .$$

Claim 16 *$E[2Z_i^{err} + Z_i^{erasures}] < \frac{2e_i}{d}$, where $e_i = \Delta(z_i, C_{in}(a_i))$ is the number of errors introduced by the channel in the i^{th} block introduced by the channel.*

Note that once we prove the claim, by linearity of expectation

$$\mathbb{E}[2\tau + s] \leq \frac{2\sum_{i=1}^{n} e_i}{d} < D ,$$

so that Lemma 15 would be proved. □

PROOF: *(Of Claim 16)* We consider two cases.

Case 1 : $a_i = c_i$ (i'th block is correctly decoded)
In this case, trivially $E[Z_i^{err}] = 0$ and

$$\mathbb{E}[Z_i^{erasures}] = \Pr[Z_i^{erasures} = 1] = \frac{2w_i}{d} = \frac{2e_i}{d}$$

since $w_i = \min\{\Delta(C_{in}(a_i), z_i), \frac{d}{2}\} = \min\{e_i, \frac{d}{2}\} \leq e_i$. Thus $\mathbb{E}[2Z_i^{err} + Z_i^{erasures}] \leq \frac{2e_i}{d}$.

Case 2 : $a_i \neq c_i$ (i'th block is incorrectly decoded)
In this case

$$\mathbb{E}[Z_i^{erasures}] = \frac{2w_i}{d} \quad \text{and} \quad \mathbb{E}[Z_i^{err}] = 1 - \frac{2w_i}{d}$$

so that

$$\mathbb{E}[2Z_i^{err} + Z_i^{erasures}] = 2 - \frac{2w_i}{d} \ . \tag{27}$$

Since $a_i \neq c_i$, we have

$$d \leq \Delta(C_{in}(c_i), C_{in}(a_i)) \leq \Delta(C_{in}(c_i), z_i) + \Delta(z_i, C_{in}(a_i)) = e_i + \Delta(z_i, C_{in}(a_i)) \ .$$

Thus if $w_i = \Delta(z_i, C_{in}(a_i))$, then $w_i + e_i \geq d$. On the other hand if $w_i = d/2$, then $e_i \geq w_i \geq d/2$, so $w_i + e_i \geq d$ as well. Thus $w_i \geq d - e_i$. Plugging this into (27) we get $\mathbb{E}[2Z_i^{err} + Z_i^{erasures}] \geq 2e_i/d$ as desired.

The two cases together complete a proof of the claim. □

5.3 GMD Decoding: Deterministic version

We now see how to derandomize the previously presented GMD decoder. We first recall that the randomness was used when we declared a particular block to be an erasure with probability $\frac{2w_i}{d}$. The derandomization is based on the following observation.

Claim 17 *There exists some threshold Θ such that if we declare an erasure in the i'th location if $\Theta \leq \frac{2w_i}{d}$ for every i, then $2\bar{\tau} + \bar{s} < D$, where $\bar{\tau}$ and $\bar{s}$ as the number of errors and erasures when Θ is used as the threshold.*

PROOF: Suppose we pick $\Theta \in [0, 1]$ uniformly. For each i declare erasure for block if $\theta \leq \frac{2w_i}{d}$. Define $\bar{\tau}$ and $\bar{s}$ as the number of errors and erasures when Θ is used as the threshold. The previous argument shows that

$$E_\Theta[2\bar{\tau} + \bar{s}] < D \tag{28}$$

since all that was required for the argument was that at location i we declared an erasure with probability $2w_i/d$. Thus,there exists a Θ for which $2\bar{\tau} + \bar{s} < D$. □ Now the derandomization problem reduces to one of searching for an appropriate value for Θ. Since Θ takes value in the continuous range $[0, 1]$, we cannot try out all possibilities in the range. However, note that if we order the w_i in increasing order so that (with notational abuse) $0 \leq w_1 \leq \ldots \leq w_N$. All values of Θ in the range $[\frac{2w_i}{d}, \frac{2w_{i+1}}{d})$ lead to the same set of erasure decisions (the first i locations are erased, and the last $n - i$ are not). Thus, our search for the right threshold Θ only needs to be over the discrete set of values $\Theta \in \{0, 1\} \bigcup \{\frac{2w_1}{d}, \ldots, \frac{2w_n}{d}\}$. Noting that w_i is an integer in the range $[0, \frac{d}{2})$ (or $d/2$ itself), there are only $O(d)$ relevant values of Θ to search over. We have thus proved the following.

Theorem 18 *For a concatenated code with outer code of block length N, alphabet size Q, and distance D and an inner code of distance d and block length n, we can correct any pattern of*

$< dD/2$ errors using $O(d)$ calls to an errors-and-erasure decoder for the outer code that can correct any pattern of τ and s erasures provided $2\tau + s < D$. The runtime of the decoding algorithm is $O(NQn^{O(1)} + NT_{\text{out}})$ where T_{out} is the running time of the outer errors-and-erasures decoder.

Together with our construction of concatenated codes with outer Reed-Solomon code that meet the Zyablov bound, we can conclude the following.

Corollary 19 *For any $R \in (0,1)$ and $\gamma > 0$, there is a polynomial time constructible family of binary codes of rate R that can be decoded from up to a fraction*

$$\frac{1}{2} \max_{R \leq r \leq 1} (1 - R/r) h^{-1}(1 - r) - \gamma$$

of errors.

Remark 20 *The above result implies the following for the two extremes of low rate and high rate codes. For $\epsilon \to 0$, we can correct up to a fraction $(\frac{1}{4} - \epsilon)$ errors in polytime with explicit binary codes of rate $\Omega(\epsilon^3)$, and we can correct a fraction ϵ of errors with explicit binary codes of rate $1 - O(\sqrt{\epsilon \log(1/\epsilon)})$. (We leave it as an exercise to check these calculations.) Note that the non-constructive rate bounds guaranteed by the Gilbert-Varshamov bound for these regimes are $\Omega(\epsilon^2)$ and $1 - O(\epsilon \log(1/\epsilon))$ respectively.*

6 Capacity achieving codes for the BSC and BEC

We will now use concatenation with outer code that can correct a small fraction of worst-case errors to construct codes that achieve the Shannon capacity of the BEC and BSC, together with polynomial time encoding/decoding. First, let us recall the definition of the Binary erasure channel.

Definition 21 (The Binary Erasure Channel (BEC)) *is parameterized by a real α, $0 \leq \alpha \leq 1$, which is called the erasure probability, and is denoted BEC_α. Its input alphabet is $\mathcal{X} = \{0,1\}$ and output alphabet is $\mathcal{Y} = \{0,1,?\}$. Upon input $x \in X$, the channel outputs x with probability $1 - \alpha$, and outputs ? (corresponding to erasing the symbol) with probability α. (It never flips the value of a bit.)*

The capacity of BEC_α equals $1-\alpha$. Recall that, if we have a $[n,k,d]_q$ code with message $x \in \mathbb{F}_q^k$ and corresponding codeword $y = Gx \in \mathbb{F}_q^n$. Suppose a subset $S \subseteq \{1,\ldots,n\}$ is received (uncorrupted). Decoding the message involves solving the linear system

$$G x_S = y_S \tag{29}$$

We reorder the indices so that the first $|S|$ entries in y are the uncorrupted entries, resulting in the following matrix system,

$$\left[\begin{array}{c} G_S \\ \hline G_{\overline{S}} \end{array}\right] \left[\begin{array}{c} x \end{array}\right] = \left[\begin{array}{c} y_S \\ \hline y_{\overline{S}} \end{array}\right] \tag{30}$$

As long as $\text{rank}(S) \geq k$, then this linear system can be solved exactly in $O(n^3)$ time by Gaussian Elimination. As long as the matrix G_S is full column rank then the solution is unique. We have the following.

Proposition 22 *Using a binary matrix of size $n \times n(1-\alpha)$ with entries chosen i.i.d uniform from $\{0,1\}$ as the generator matrix G achieves the capacity of the BEC with high probability.*

The drawbacks with this solution are the cubic time and randomized nature of the construction. in addition, for a given choice of G it is hard to certify that (most) $(1+\alpha)k \times k$ sub-matrices of G have full (column) rank. We will use the idea of concatenation to make this constructive.

Let α be the erasure probability of the BEC and say our goal is to construct a code of rate $(1-\alpha-\epsilon)$ that enables reliable communication on BEC_α. Let C_1 be a linear time encodable/decodable binary code of rate $(1-\epsilon/2)$ that can correct a small constant fraction $\gamma = \gamma(\epsilon) > 0$ of *worst-case* erasures. Such codes were constructed in [13, 1]. For the concatenated coding, we do the following. For some parameter b, we block the codeword of C_1 into blocks of size b, and then encode each of these blocks by a suitable *inner* binary linear code C_2 of dimension b and rate $(1-\alpha-\epsilon/2)$. The inner code will be picked so that it achieves the capacity of the BEC_α, and specifically recovers the correct message with success probability at least $1-\gamma/2$. For $b = b(\epsilon,\gamma) = \Omega\left(\frac{\log(1/\gamma)}{\epsilon^2}\right)$, a random code meets this goal with high probability, so we can find one by brute-force search (that takes constant time depending only on ϵ).

The decoding proceeds as one would expect: first each of the inner blocks is decoded, by solving a linear system, returning either decoding failure or the correct value of the block. (There are no errors, so when successful, the decoder knows it is correct.) Since the inner blocks are chosen to be large enough, each inner decoding fails with probability at most $\gamma/2$. Since the noise on different blocks are independent, by a Chernoff bound, except with exponentially small probability, we have at most a fraction γ of erasures in the outer codeword. (For $R = 1-\alpha-\epsilon$, we have $Pr(decoder\ failure,\ error) \leq 2^{-\epsilon^2 n} < \gamma$). These are then handled by the linear-time erasure decoder for C_1. We thus have,

Theorem 23 *For the BEC_α, we can construct codes of rate $1-\alpha-\epsilon$, i.e., within ϵ of capacity, that can be encoded and decoded in $n/\epsilon^{O(1)}$ time.*

While this is pretty good, the brute-force search for the inner code is unsatisfying, and the BEC is simple enough that better runtimes (such as $O(n\log(1/\epsilon))$) are achieved by certain irregular LDPC codes.

A similar approach can be used for the BSC_p. We recall the definition of the BSC.

Definition 24 (The Binary Symmetric Channel (BSC)) *has input alphabet $\mathcal{X} = \{0,1\}$ and output alphabet $\mathcal{Y} = \{0,1\}$. The BSC is parameterized by a real number p, $0 \leq p \leq \frac{1}{2}$ called the crossover probability, and often denoted BSCp. The channel flips its input with probability p.*

For the BSC, the outer code C_1 must be picked so that it can correct a small fraction of worst-case *errors* — again, such codes of rate close to 1 with linear time encoding and decoding are known [13, 8]. Everything works as above, except that the decoding of the inner codes, where we

find the codeword of C_2 closest to the received block, requires a brute-force search and this takes $2^b = 2^{\Omega(1/\epsilon^2)}$ time. This can be improved to polynomial in $1/\epsilon$ by building a look-up table, but then the size of the look-up table, and hence the space complexity and time for precomputing the table, is exponential in $1/\epsilon$. Thus,

Theorem 25 *For the BSC_p, we can construct codes of rate $1 - H(p) - \epsilon$, i.e., within ϵ of capacity, that can be encoded in $n/\epsilon^{O(1)}$ time and which can be reliably decoded in $n2^{1/\epsilon^{O(1)}}$ time.*

It remains an important open question to obtain such a result with decoding complexity $n/\epsilon^{O(1)}$, or even poly(n/ϵ).[1]

We also want to point out that recently an alternate method using LP decoding has been used to obtain polynomial time decoding at rates arbitrarily close to capacity [5]. But this also suffers from a similar poor dependence on the gap ϵ to capacity.

References

[1] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996. 16

[2] E. R. Berlekamp. Nonbinary bch decoding. *International Symposium on Information Theory*, 1968. 11

[3] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, 1983. 11

[4] R. T. Chien. Cyclic decoding procedure for the bose-chaudhuri-hocquenghem codes. *IEEE Transactions on Information Theory*, IT-10:357–363, 1964. 11

[5] Jon Feldman and Clifford Stein. LP decoding achieves capacity. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 460–469, 2005. 17

[6] G. D. Forney. Generalized minimum distance decoding. *IEEE Trans. Information Theory*, IT-12:125–131, 1966. 12

[7] Peter Gemmell and Madhu Sudan. Highly resilient correctors for multivariate polynomials. *Information Processing Letters*, 43(4):169–174, 1992. 6

[8] V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, October 2005. 16

[9] J. Justesen. Class of constructive asymptotically good algebraic codes. *Information Theory, IEEE Transactions on*, 18(5):652 – 656, sep 1972. 1

[1] We remark that asymptotically, with ϵ fixed and $n \to \infty$, the exponential dependence on $1/\epsilon$ can be absorbed into an additional factor with a slowly growing dependence on n. However, since in practice one is interested in moderate block length codes, say $n \leq 10^6$, a target runtime such as $O(n/\epsilon)$ seems like a clean way to pose the underlying theoretical question.

[10] J. L. Massey. Shift-register synthesis and bch decoding. *IEEE Trans. Information Theory*, IT-15:122–127, 1969. 11

[11] W. W. Peterson. Encoding and error-correction procedures for the bose-chaudhuri codes. *IRE Transactions on Information Theory*, IT-6:459–470, 1960. 8

[12] B.-Z. Shen. A justesen construction of binary concatenated codes that asymptotically meet the zyablov bound for low rate. *Information Theory, IEEE Transactions on*, 39(1):239 –242, jan 1993. 4

[13] D. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1732, 1996. 16

[14] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding goppa codes. *Information and Control*, 27:8799, 1975. 11

[15] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. *US Patent Number 4,633,470*, December 1986. 6

[16] Wikipedia. Chien search — Wikipedia, the free encyclopedia, 2004. [Online accessed 24-Mar-2010]. 11

Introduction to Coding Theory **CMU: Spring 2010**

Notes 8: Expander Codes and their decoding

March 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkat Guruswami & Ankit Sharma*

In this lecture, we shall look at Expander Codes, Tanner Codes which are a generalization of Expander Codes and Distance amplification codes. These codes visualize the code as a graph where the value of the i^{th} bit of a codeword corresponds to, for example, the value associated with the i^{th} vertex or the i^{th} edge of the graph. The properties of the graph determine the properties such as distance and efficient decodability of the code. Further, the constraints such as parity checks put on the codewords manifest as local constraints on vertices or edges such as constraints on the set of the values the edges incident to a vertex can take.

The motivation to view linear codes as graphs and use a graph-theoretic approachto construct them comes from the parity-check and generator matrix view of the code. For a $[n, n-m]_2$, we can interpret the parity check matrix as representing a $n \times m$ bipartite graph and then describe the properties of the code from the properties of the graph. An interesting class of codes are Low Density Parity Check (LDPC) codes which have a small number of 1's in each row and column of the parity-check matrix and hence manifest, under the graph view, as sparse graphs.

Let us start with a few notations and definitions.

1. $G(V, E)$ denotes a graph G with vertex set V and edge set E. For a bipartite graph G, we shall denote by L the set of vertices in the left partition of G and by R the set of vertices in the right partition of G. A bipartite graph G will usually be denoted as $G(L \cup R, E)$.

2. A graph G is said to be *d-regular* if each vertex of G has degree d. A bipartite graph G is said to be *d-left regular* if all vertices in the left partition of G have degree d. Similarly, we can define a bipartite graph to be *d-right regular*.

3. For any vertex set $S \subseteq V$ in a graph $G(V, E)$, a vertex $q \in V \backslash S$ is said to be a *neighbour* of S if it is adjacent to some vertex in S. We denote by $N(S)$ the set of neighbours of S. For a bipartite graph $G(L \cup R, E)$, if $S \subseteq L$, then $N(S) \subseteq R$ and vice-versa.

4. For any vertex set $S \subseteq V$ in a graph $G(V, E)$, a vertex $q \in V \backslash S$ is said to be a *unique neighbour* of S if it is adjacent to exactly one vertex in S. We denote by $U(S)$ the set of unique neighbours of S. For a bipartite graph, if $S \subseteq L$, then $U(S) \subseteq R$ and vice-versa.

1 Expander Codes

We first give the definition of a bipartite expander graphs. A bipartite expander graph essentially has the property that every 'small' set of vertices in the left partition has a significantly 'large' neighbourhood in the right partition. Formally,

Definition 1 *A $(n, m, D, \gamma, \alpha)$ bipartite expander is a D-left-regular bipartite graph $G(L \cup R, E)$ where $|L| = n$ and $|R| = m$ such that $\forall S \subseteq L$ with $|S| \leq \gamma n$, $N(S) \geq \alpha |S|$.*

In the above definition, γ gives the measure of a 'small' set and α gives the measure of a 'large' neighbourhood. α is called the *expansion factor*. Note that $\alpha \leq D$ trivially. The following theorem shows the existence of expander graphs that approach this optimal bound.

Theorem 2 *$\forall \epsilon > 0, m \leq n$, $\exists \gamma > 0$ and $D \geq 1$ such that a $(n, m, D, \gamma, D(1-\epsilon))$ expander exists. Additionally, $D = \Theta(\frac{\log(n/m)}{\epsilon})$ and $\gamma n = \Theta(\frac{\epsilon m}{D})$.*

Some remarks about the above expanders:

1. Note that $m \leq n$, so we want expansion from the larger side to the smaller side. This is the harder direction as there is less room to expand into.
2. For a given value of m and n, the parameters of the the above expander are optimal up to $O(1)$ factors and the expansion factor can be brought as close to D at the cost of increasing D.
3. $\gamma n D$ is a trivial lower bound on m since sets of size up to γn expand by a factor of almost D. The above result achieves a value of m that is $1/\epsilon$ times larger than this trivial bound.

The proof of the above theorem is through the probabilistic method. Explicit constructions were also known for $\alpha \approx D/2$, but for a long time there was no explicit construction known with expansion better than $D/2$. Capalbo, Reingold, Vadhan, and Wigderson [2] in 2002 gave an explicit construction of a constant degree expander with expansion $D(1-\epsilon)$ for any desired $\epsilon > 0$, and any desired imbalance ratio m/n.

We now come to the connection between expanders and codes. For this let us define Factor Graphs. For an $[n, n-m]_2$ linear code, construct the bipartite graph F corresponding to the $(n-m) \times n$ parity-check matrix of C as follows

- there is a node in the left partition L_F corresponding to each bit of a codeword in C ($|L_F)| = n$),
- there is a node in the right partition R_F corresponding to each parity check ($|R_F| = n - m$),
- and there is an edge between a node u in L_F and node v in R_F if and only if the bit corresponding to u participates in the parity check corresponding to v in C.

Such a bipartite graph F is called the *factor graph* of C. We can see that the above construction establishes a correspondence between linear codes and bipartite graphs. The nodes on the left are the bits of the code and the nodes on the right are the parity checks. Each codeword assigns each node on the left the value of the corresponding bit. An assignment of values to the nodes on the left is a valid codeword if and only if it satisfies all parity checks i.e. if we denote by V_q the value of the node q on the left, then V corresponds to a valid codeword if and only if $\forall u \in R_F, \sum_{j \in N(u)} V_j = 0$. The above construction also indicates how we can construct a parity-check matrix corresponding to a bipartite graph that has fewer nodes in the right partition compared to the left partition.

Expander codes are linear codes whose factor graphs are bipartite expander graphs. Let us denote the code corresponding to an expander graph G by $C(G)$.

We now establish a useful property of bipartite expander graphs with expansion close to degree D.

Lemma 3 *Let G be a $(n, m, D, \gamma, D(1-\epsilon))$ expander graph with $\epsilon < 1/2$. For any $S \subseteq L_G$ such that $|S| \leq \gamma n$, $U(S) \geq D(1-2\epsilon)|S|$.*

PROOF: The total number of edges going out of S is $D|S|$ by virtue of G being D-left-regular. By the expansion property, $N(S) \geq D(1-\epsilon)|S|$. Hence, out of $D|S|$ edges emanating out of S, at least $D(1-\epsilon)|S|$ go to unique vertices which leaves at most $\epsilon D|S|$ edges. Therefore, at most $\epsilon D|S|$ vertices out of the at least $D(1-\epsilon)|S|$ vertices in $N(S)$ can have more than one incident edge. Hence, $U(S) \geq D(1-2\epsilon)|S|$. □

The above already implies that the distance $\Delta(C(G))$ of the code satisfies $\Delta(C(G)) \geq \gamma n$. (Why?) The next theorem indicates this bound by roughly a factor of two.

Theorem 4 *Let G be a $(n, m, D, \gamma, D(1-\epsilon))$ expander. Then $\Delta(C(G)) \geq 2\gamma(1-\epsilon)n$.*

PROOF: Since $C(G)$ is a linear code, it is sufficient to establish that the weight of any non-zero codeword is at least $2\gamma(1-\epsilon)n$. For contradiction, let p be a codeword of Hamming weight less than $2\gamma(1-\epsilon)n$. Let S be the set of nodes in G which are set to 1 in p.

Clearly, the parity check of any node in $U(S)$ cannot be satisfied since the parity-check-node shares an edge with exactly one node in set S and this node has value 1 by virtue of being in S and edges to nodes in $L \setminus S$ cannot satisfy the parity since the value of those nodes is 0. Hence, if we show that $U(S)$ is non-empty, we have shown that the purported codeword does not satisfy all parity checks which is a contradiction.

If $|S| \leq \gamma n$, then we are through since by Lemma 3, $|U(S)| \geq D(1-2\epsilon)|S|$. If $|S| \geq \gamma n$, then choose a subset Q of S having exactly γn nodes. By Lemma 3, $|U(Q)| \geq D(1-2\epsilon)|Q| = D(1-2\epsilon)\gamma n$.

Now $|S \setminus Q| < \gamma(1-2\epsilon)n$ since $|S| < 2\gamma(1-\epsilon)n$. Hence, there are less than $D\gamma(1-2\epsilon)n$ edges emanating out of $S \setminus Q$ and since $|U(Q)| \geq D(1-2\epsilon)\gamma n$, there cannot be an edge incident to each node in $U(Q)$ from $S \setminus Q$. Hence, there exists a node in $U(Q)$ which has exactly one edge incident to it from S and therefore $U(S) \neq \emptyset$ which completes the proof. □

The rate of expander code is at least $1 - m/n$ and hence if m is smaller by a constant factor compared to n, the expander code has rate bounded away from 0. Thus codes whose factor grpahs are unbalanced expanders with expansion factor $(1-\epsilon)D$ for $\epsilon < 1/2$ are asymptotically good, and therefore explicit constructions of such expanders immediately gives an explicit construction of asymptotically good odes. This was the first constructive method for asymptotically good codes that did not rely on code concatenation. This is in itself a nice feature, but we will now see that these expander codes also admit very efficient (linear time) decoding algorithms.

1.1 Decoding algorithm for Expander Codes

We now present a decoding algorithm for correcting all error patterns of Hamming weight less than $\gamma(1-2\epsilon)n$ assuming $\epsilon < 1/4$ (i.e., when the expansion factor exceeds $3D/4$). Let r be the received word and let V_q be the value assigned to node q in L_G by r. Each parity check in R_G is either satisfied or unsatisfied. The algorithm proceeds by flipping the value of those nodes in L_G which have more unsatisfied checks in their neighbourhood than satisfied checks. This process continues till there no unsatisfied checks left.

Algorithm

While there exists a node y in L_G with more unsatisfied than satisfied checks in $N(y)$

- Flip V_y and update the list of satisfied and unsatisfied checks in R_G.

This completes the description of the algorithm. We now prove the correctness and analyze the time complexity of the algorithm. First we prove that if the number of errors is at most γn, there must exist a node on the left whose neighbourhood of D check nodes has more unsatisfied checks than satisfied checks. This ensures that the algorithm will get started.

Lemma 5 *If the number of errors is at most than γn (and at least 1), then there exists a node in L_G which is adjacent to more then $D/2$ unsatisfied checks. (This assumes that $\epsilon < 1/4$.)*

PROOF: Let $T \neq \emptyset$ be the set of error locations. Since $|T| \leq \gamma n$, by Lemma 3, $|U(T)| \geq D(1-2\epsilon)|T| > \frac{D}{2}|T|$ if $\epsilon < 1/4$. All checks in $U(T)$ are clearly unsatisfied. Hence, there exists a node in T that is adjacent to more than $D/2$ unsatisfied checks. Since each node in L_G has D neighbours, therefore the above lemma implies that the node would have more unsatisfied than satisfied checks. □

Lemma 6 *If we start with a received word having less than $\gamma(1-2\epsilon)n$ errors then we can never reach a word with γn errors in any interim step of the algorithm.*

PROOF: We flip a node on the left only when the number of unsatisfied checks is greater than the number of satisfied checks in its neighbourhood. Thus with each flip the number of unsatisfied checks decreases by at least 1.

The received word has less than $\gamma(1-2\epsilon)n$ errors and therefore it has less than $D\gamma(1-2\epsilon)n$ unsatisfied checks (by D-left-regularity of the graph) to begin with. If we ever reach an intermediate string with γn errors, then the set of error locations would have at least $D(1-2\epsilon)\gamma n$ unique neighbours (by Lemma 3) and hence there would be at least as many unsatisfied checks. This contradicts the facts that we start with less than $D\gamma(1-2\epsilon)n$ unsatisfied checks and this number cannot increase. □

1.1.1 Correctness of the algorithm

By Lemmas 5 and 6, we see that if we start with an erroneous codeword with less than $\gamma(1-2\epsilon)n$ errors, the algorithm will always find a node (participating in more unsatisfied checks than satusfied checks) to flip. With each node flip, the number of unsatisfied checks goes down by at least 1 and hence the algorithm terminates in at most m iterations. Lemma 6 proves that during the course of the algorithm, there won't be any stage at which the number of errors in the current word (compared to the originally closest codeword) is at least γn. Since the distance of the code is at least $2\gamma(1-\epsilon)n > \gamma n$ ($\because \epsilon < 1/2$), the final codeword to which the algorithm converges must be the original closest codeword.

1.1.2 Running Time of the algorithm

Let d be the maximum degree of a node in R_G.

1. Preprocessing Stage: The computation of all unsatisfied nodes in R_G as a preprocessing step takes $O(md)$ time. As part of preprocessing step, we also associate with each node in L_G the number of unsatisfied checks it is part of and make a list, call it Q, of nodes in L_G which have more unsatisfied than satisfied checks. This takes an additional $O(Dn) = O(Dmd)$ time.
2. Time complexity of each iteration: In each iteration instead of searching for a node with more unsatisfied than satisfied checks, we remove an element from list Q. Further, after flipping the node, we update the list of unsatisfied checks in R_G in $O(D)$ time. We take further $O(Dd)$ time to update the number of unsatisfied checks associated with each element in L_G and to insert any element which now have more unsatisfied than satisfied check into Q and to remove elements from Q which due to the flip have lesser unsatisfied than satisfied checks. Hence, each iteration can be implemented in $O(Dd)$ time.
3. Number of Iterations: The original number of unsatisfied checks can be at most m. As argued above, in each iteration the number of unsatisfied checks reduces by at least 1. Thus the total number of iterations is at most m.

Hence the algorithm can be implemented to run in $O(Ddm) = O(n)$ time when D, d are constants.

Note: These codes being linear can be encoded in $O(n^2)$ time by multiplying the message vector by the generator matrix.

2 Tanner Codes

Let G be a $n \times m$ bipartite graph which is d-right regular and let $C_0 \subseteq \mathbb{F}_2^q$ be a binary linear code.

Definition 7 (Tanner code) *The Tanner code $X(G, C_0)$ is defined as the set*

$$\{c \in \mathbb{F}_2^n | \forall u \in R_G, c_{|N(u)} \in C_0\}$$

where $c_{|N(u)} \in \mathbb{F}_2^d$ denotes the subsequence of c formed by the bits corresponding to the neighbours of u in L_G.

Remark 8 *We note the following:*

1. *Tanner codes are linear codes since C_0 is a linear code.*
2. *Tanner Codes are a generalization of expander codes since in expander code C_0 was chosen to be the $[d, d-1, 2]_2$ parity check code.*

We claim that the dimension of $X(G, C_0)$ is at least $n - m(d - dim(C_0))$. This is because for each $i \in R_G$, the condition that $c_{|N(i)} \in C_0$ imposes $d - dim(C_0)$ independent linear constraints on the bits of c. Therefore, the condition $\forall i \in R_G, c_{|N(i)} \in C_0$ imposes at most $m(d - dim(C_0))$ independent linear constraints and hence the claim follows.

The usefulness of Tanner codes comes from the fact that they require a much lower expansion factor through the choice of an appropriate C_0. In expander codes which are a special case of Tanner codes, we used parity check codes for C_0. Since parity check codes have distance 2, we required that all sets of size at most γn expand by a factor of more than $D/2$ in order to argue that the code had distance at least γn. However, if we use a local code C_0 of distance d_0, then we only require an expansion factor exceeding D/d_0 to ensure the same code distance. Hence a good choice of C_0 allows us to construct explicit Tanner codes using graphs with weaker expansion properties which are therefore easier to construct.

In order to construct codes with a high rate, we require a highly unbalanced bipartite graph so that there are much fewer parity checks than the number of bits in a code word. One way of achieving this is through the construction of Edge Vertex Incidence Graph which we define next.

Definition 9 *The Edge Vertex Incidence Graph of a graph $G = (V, E)$ is defined as the bipartite graph $H_0 = (L \cup R, E')$ where L has a node corresponding to each edge in E, R has a node corresponding to each node in V and an edge exists in E' between each node e in L and corresponding u_e and v_e in R where u_e and v_e are the nodes in R corresponding to the end-points of edge e in E.*

In the graph-code correspondence which we had so far, the bits of the codeword used to sit on nodes in the left partition and the constraints used to be imposed by nodes in the right partition. The edges of graph G form the left partition of H_0 and the nodes of graph G form the right partition of H_0. Hence, we can equivalently view the codeword bits as "sitting" on the edges of the graph G, with each node of the graph G imposing a local constraint on the values on the d edges incident at that node.

Let G be a d-regular graph with N vertices and $Nd/2$ edges. Under the modified view of code-bits sitting on the edges of the graph, the Tanner code $X(H_0, C_0)$ can alternately been defined directly in terms of G as

$$T(G, C_0) = \{c \in \mathbb{F}_2^{\frac{Nd}{2}} | \forall v \in V(H), c_{|\Gamma(v)} \in C_0\} .$$

(We use the notation $T(\cdot, \cdot)$ instead of $X(\cdot, \cdot)$ to highlight this distinction.) Again, $T(G, C_0)$ is a linear code of dimension at least $Nd/2 - N(d - dim(C_0))$ and a sufficient condition for positive dimension is that $dim(C_0) > \frac{d}{2}$.

We will now use for G a good "spectral" expander.

Definition 10 *A graph $G = (V, E)$ is said to be a (n, d, λ)-expander if G is a d-regular graph on n vertices and $\lambda = min\{\lambda_2, |\lambda_n|\}$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of the adjacency matrix of H.*

What makes (n, d, λ) expander useful? This is because of Expander Mixing Lemma, which we state next and crucially use in our analysis. We do not prove this lemma here, but it is not hard to prove and a proof can be found in several places (eg. the book by Alon and Spencer).

Lemma 11 (Expander mixing lemma) *Let $G = (V, E)$ be a (n, d, λ) expander graph. Then $\forall S, T \subseteq V$ we have*

$$\left| \; |E(S,T)| - \frac{d|S||T|}{n} \; \right| \leq \lambda \sqrt{|S||T|} \tag{1}$$

where $|E(S, T)|$ is the number of edges between sets S and T with the edges in $(S \cap T) \times (S \cap T)$ counted twice.

Remark 12 *Some comments:*

1. *Since G is a d-regular bipartite graph, therefore the highest eigen value λ_1 would be d.*
2. *Since G is a d-regular graph, we know that there are $d|S|$ edges coming out of S. Now, if it were a purely random graph, we would expect that $|T|/n$ fraction of these edges to end up in T. Thus the expected value of $|E(S, T)|$ would be $\frac{d|S||T|}{n}$. The expander mixing lemma upper bounds the deviation of $|E(S, T)|$ from the random graph expected value in terms of how small the second largest eigenvalue (in absolute value) is in comparison to d. Note that the lemma bounds this deviation for all pairs of sets S, T.*

For technical reasons, we will find it convenient to work with a "bipartite version" of G called its double cover instead of G itself.

Definition 13 (Double cover) *The Double Cover of a graph $G = (V_G, E_G)$ is defined as the bipartite graph $H = (L_H \cup R_H, E_H)$ with both left and right partitions of graph H being equal to V_G i.e. $L_H = R_H = V_G$ and $\forall (u, v) \in E_G$, there is (u, v) and (v, u) in E_H. Hence, the double cover H of graph G has two copies of each node of G, one in the left partition and the other in the right partition, and there are two copies of each edge (u, v) of G.*

Our final code will be $T(H, C_0)$ for suitable C_0 where H is the double cover of an (n, d, λ)-expander G. Working with the bipartite graph H (instead of G) makes the description and analysis of the decoding algorithm simpler and cleaner.

Hence, if G is (n, d, λ) expander, we now take H, the double cover of G. The code-bits still sit on the edges of H as they were in G and the constraints on the values which the edges can take are also essentially the same as they were in G. However, as we will see the analysis becomes a lot more simplified.

Clearly H is a $n \times n$ d-regular bipartite graph i.e. there are n nodes in both left and right partition and each node has d neighbours. What is the analogue of expander mixing lemma for H? It is

essentially the same. Consider the $n \times n$ matrix corresponding to H that has 1 in $(i,j)^{th}$ position if there is an edge in H between the i^{th} node in left partition and j^{th} node in right partition else it is 0. This is exactly the adjacency matrix of G. Hence $d = \lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \geq \lambda_n$ would be the eigenvalues of the matrix (same as that of adjacency matrix of G) and define $\lambda = \min\{\lambda_2, |\lambda_n|\}$.

Lemma 14 (Expander mixing lemma: bipartite version) *Let H be a $n \times n$ d-regular bipartite graph with λ as defined above. Then $\forall S \subseteq L, T \subseteq R$,*

$$\left| \; |E(S,T)| - \frac{d|S||T|}{n} \; \right| \leq \lambda \sqrt{|S||T|}$$

where $E(S,T)$ is the set of edges between sets S and T. The double counting which we had stated explicitly in Lemma 11 is implicit here.

Theorem 15 *Let $C_0 \subset \mathbb{F}_2^d$ have distance $\geq \delta_0 d$. Then the relative distance of $T(H, C_0)$ is $\geq \delta_0(\delta_0 - \frac{\lambda}{d})$*

Proof: Since $T(H, C_0)$ is a linear code, it will be sufficient to prove that no non-zero codeword has weight less than $\delta_0(\delta_0 - \frac{\lambda}{d})nd$.

Let c be a codeword in $T(H, C_0)$ and let F be the set of edges in H which have their corresponding bits non-zero in c. Let S be the set of nodes in L which have at least one edge belonging to F incident on them. Similarly, let T be the set of nodes in R which have at least one edge from F incident on them.

Since the distance of C_0 is $\delta_0 d$ any node in H which is incident to a non-zero edge (i.e., an edge of F) should have at least $\delta_0 d$ edges from F incident to it. (This is because we know that any codeword $c \in T(H, C_0)$ satisfies the condition that for each node in H, the values assigned to the edges incident to the node forms a codeword in C_0.)

This implies that each vertex in S and T must have at least $\delta_0 d$ non-zero edges incident to it.

Hence, $|F| \geq \delta_0 d|S|$ and $|F| \geq \delta_0 d|T|$ and therefore, $|F| \geq \delta_0 d\sqrt{|S||T|}$.

Now, $|F| \leq |E(S,T)|$ and by Expander Mixing Lemma, $|E(S,T)| \leq \frac{d|S||T|}{n} + \lambda\sqrt{|S||T|}$. From above we get,

$$\delta_0 d\sqrt{|S||T|} \leq \frac{d|S||T|}{n} + \lambda\sqrt{|S||T|}$$

which implies that $\sqrt{|S||T|} \geq (\delta_0 - \frac{\lambda}{d})n$. Recalling that $|F| \geq \delta_0 d\sqrt{|S||T|}$, we conclude $|F| \geq \delta_0(\delta_0 - \frac{\lambda}{d})nd$ which completes the proof. □

Remark 16 *Note that when H is the $n \times n$ complete bipartite graph, the code $T(H, C_0)$ is simply the tensor product of C_0 with itself and thus has relative distance exactly δ_0^2. The above theorem states that for a good expander with $\lambda = o(d)$, in the limit of large degree d, the relative distance becomes $\approx \delta_0^2$. Thus we can obtain distance as good as the product construction, but we can get much longer codes (compared to the product construction, which only gives a code of block length d^2 starting with a code of block length d).*

2.1 Rate-Distance Tradeoff

Denote the rate of code C_0 by R. Then rate of $T(H, C_0)$ is at least

$$\frac{nd - 2nd(1-R)}{nd} = 2R - 1 \ .$$

Let $\delta = \delta_0^2$ be approximately the relative distance of $T(H, C_0)$ in the limit of large d. By picking C_0 to satisfy $R \geq 1 - h(\delta_0)$ (meeting the Gilbert-Varshamov bound), we obtain the following rate vs relative distance for $T(H, C_0)$:

$$R(T(H, C_0) \geq 2(1 - h(\delta_0)) - 1 \approx 1 - 2h(\sqrt{\delta}) \ .$$

The rate is positive $\delta < 0.01$. This implies that we get a positive rate for $T(H, C_0)$ only when the relative distance of $T(H, C_0)$ is rather small.

2.2 Algorithm for Decoding

Sipser and Spielman in 1995 [3] introduced the above construction Tanner codes and gave an algorithm to correct a fraction $\delta_0^2/48$ of errors. Zémor in 2001 [4] gave an improved algorithm to correct a fraction $\delta_0^2/4$ of errors for $T(H, C_0)$ by exploiting the bipartiteness of H. We describe this algorithm and its analysis now.

Specifically, we will give an algorithm to decode a received word to the correct codeword assuming that the number of errors is at most $(1-\epsilon)\frac{\delta_0}{2}(\frac{\delta_0}{2} - \frac{\lambda}{d})nd$ which is roughly a quarter of the distance of the code.

Let $y \in \mathbb{F}_2^{|E|}$ be the received word which we wish to decode. Denote by $y_{|\Gamma(u)} \in \mathbb{F}_2^d$ the subsequence of y formed by the bits corresponding to the edges incident to u.

Define a *Left-decoding iteration* to be the following step performed in parallel for all left nodes:

> $\forall v \in L$ replace $y_{|\Gamma v}$ by $c \in C_0$ such that $\Delta(c, y_{|\Gamma(v)})$ is minimized.

We can efficiently find the closest c to $y_{|\Gamma v}$ by brute force since code C_0 is a small sized code. Note that since this is a bipartite graph, therefore, no two v's in L share an edge and hence there are no conflicts when changing the value of an edge. Similarly, we can define *Right-decoding iteration.*

The algorithm for decoding is as follows:

> alternately perform left-decoding and right-decoding iterations for $A \log n$ iterations

for a large enough constant $A < \infty$.

2.2.1 Correctness of the decoding algorithm

Lemma 17 *Assume that $\lambda/d < \delta_0/3$. When the number of errors is at most $(1-\epsilon)\frac{\delta_0}{2}(\frac{\delta_0}{2} - \frac{\lambda}{d})nd$, the above algorithm converges to the correct codeword when run for $A(\epsilon) \log n$ iterations.*

PROOF: We use the terminology that an error is incident to a vertex if the value associated with at least one edge incident to the node is incorrect in comparison to the correct codeword. Let

$$S_1 = \{v \in L_H | \exists \text{ an error incident to } v \text{ after the first left side decoding step}\} .$$

After the first left-decoding step a node v in L_H has an error incident onto itself only if it had more $\delta_0 d/2$ error edges incident onto itself before the decoding step. This is because any v which had less than $\delta_0 d/2$ incident error edges would, after the decoding step, have corrected all the incident errors. Let E_0 denote the number of errors in y before the decoding step.

Since each node in S_1 had more than $\delta_0 d/2$ errors incident on itself, therefore $E_0 \geq |S_1|\frac{\delta_0 d}{2}$. However, from assumption $E_0 \leq (1-\epsilon)\frac{\delta_0}{2}(\frac{\delta_0}{2} - \frac{\lambda}{d})nd$ and therefore, $|S_1| \leq n(\frac{\delta_0}{2} - \frac{\lambda}{d})(1-\epsilon)$.

Now we run the right-side decoding over R_H.

Let $T_1 = \{v \in R_H | \exists$ an error incident to v after the first right side decoding step$\}$.

We want to show that $|T_1| \leq \alpha |S_1|$ for some $\alpha < 1$.

Using the same reasoning as above, after the right-decoding step, a node in R_H would have error incident onto itself only if it had more than $\delta_0 d/2$ errors incident onto itself before the right-decoding step. Now, $|E(S_1, T_1)|$ is clearly an upper bound on the number of error edges incident onto the nodes in T_1 before the right-decoding step. Since each node in T_1 had at least $\delta_0 d/2$ errors incident onto itself before the right decoding step, therefore, $\frac{\delta_0 d}{2}|T_1| \leq |E(S_1, T_1)|$.

However, by Expander Mixing Lemma, we can upper bound $|E(S_1, T_1|$ by $\frac{d|S_1||T_1|}{n} + \lambda\sqrt{|S_1||T_1|}$. On using the fact that $|S_1| \leq n(\frac{\delta_0}{2} - \frac{\lambda}{d})(1-\epsilon)$ and AM-GM inequality, $\frac{|S_1|+|T_1|}{2} \geq \sqrt{|S_1||T_1|}$, we can change the upper bound to

$$|E(S_1, T_1)| \leq |T_1|(\frac{d\delta_0}{2} - \lambda)(1-\epsilon) + \lambda\frac{|S_1| + |T_1|}{2} .$$

Together with $\frac{\delta_0 d}{2}|T_1| \leq |E(S_1, T_1)|$, we get

$$\frac{\delta_0 d}{2}|T_1| \leq |T_1|(\frac{d\delta_0}{2} - \lambda)(1-\epsilon) + \lambda\frac{|S_1| + |T_1|}{2}$$

which upon rearranging yields

$$|T_1| \leq \frac{\lambda}{\epsilon\delta_0 d + (1-2\epsilon)\lambda}|S_1| \leq \frac{|S_1|}{1+\epsilon}$$

provided $\delta_0 d > 3\lambda$.

Hence, in each iteration set of nodes on which the error-edges are incident decreases geometrically by a factor of $(1+\epsilon)$. Therefore in $O_\epsilon(\log n)$ rounds, no error edges remain. □

2.2.2 Running Time

The above decoding algorithm can clearly be implemented in $O(n \log n)$ time since each iteration takes $O(n)$ time and the number of iteration by the Lemma 17 above is $O(log(n))$. We now show a

linear time implementation of the decoding algorithm.The key observation is that in each iteration, the only vertices (in the relevant side for that iteration) that need to be locally decoded are those that are adjacent to some vertex of the opposite side which had some incident edges flipped in the local decoding of the previous iteration. The latter set shrinks geometrically in size by Lemma 17. Let us be somewhat more specific. After the first (left) iteration, for each $v \in L$, the local subvector $y_{|\Gamma(v)}$ of the current vector $y \in \{0,1\}^E$ belongs to the code C_0. Let $T(y) \subset R$ be the set of right hand side vertices u for which $y_{|\Gamma(u)}$ does not belong to C_0. Let $z \in \{0,1\}^E$ be the vector after the running the right side decoding on y. Note that for each $w \in L$ that is not a neighbor of any vertex in $T(y)$, its neighborhood is untouched by the decoding and hence the incident edges on such nodes still form a valid codeword in C_0. This means that in the next iteration (left side decoding), all these vertices w need not be examined at all.

The algorithmic trick therefore is to keep track of the vertices whose local neighborhoods do not belong to C_0 in each round of decoding. In each iteration, we only perform local decoding at a subset of nodes D_i that was computed in the previous iteration. (This subset of left nodes gets initialized after the first two rounds of decoding as discussed above.) After performing local decoding at the nodes in D_i, we prepare the stage for the next round by computing D_{i+1} for the opposite side as the set of neighbors of all nodes in D_i some of whose incident edges were flipped during the current iteration. Using an argument similar to Lemma 17 one can show that the D_i's are geometrically decreasing in size. This implies that the total number of nodes whose local neighborhoods have to be examined over all iterations is $O_\epsilon(n)$. As each local decoding step can be performed in $O_d(1)$ time, overall we get a linear time implementation of the decoding algorithm.

3 Distance Amplification of Codes

The expander codes constructed so far have rather small relative distance. We now see a way to boost the distance of a code by combining it with certain expander-like graphs. The construction is originally due to Alon, Bruck, Naor, Naor, and Roth [1].

Let $G = (L, R, E)$ be a bipartite graph with $L = \{1, 2, \ldots, n\}$ and $R = \{1, 2, \ldots, m\}$ which is D-left-regular and d-right-regular. Let C be a binary linear code of block length $n = |L|$, so that bits of codewords of C can be "placed" on the left vertices of G.

We define the distance amplified code $G(C) \subset \Sigma^m$ where $\Sigma = \{0,1\}^d$ as follows. Note that the alphabet size of the code $G(C)$ is 2^d.

Definition 18 *For $c \in \{0,1\}^n$, define $G(c) \in (\{0,1\}^d)^m$ by*

$$G(c)_j = (c_{\Gamma_1(j)}, c_{\Gamma_2(j)}, \cdots, c_{\Gamma_d(j)})$$

for $j = 1, 2, \ldots, m$ where $\Gamma_i(j) \in L$ denotes the i'th neighbor of $j \in R$. Now define the code $G(C)$ as

$$G(C) = \{G(c) \mid c \in C\} .$$

Since each bit of a codeword $c \in C$ is repeated D times in the associated codeword $G(c) \in G(C)$, we get the following.

Lemma 19 *Rate of Code $G(C)$ is $1/D$ times the rate of C.*

Lemma 19 follows from the definition of distance amplification code $G(C)$.

Definition 20 *A bipartite graph $G = (L, R, E)$ is said to be (γ, β) disperser if $\forall S \subseteq L_G$ with $|S| \geq \gamma n$, $|N(S)| \geq \beta m$.*

The lemma below follows from the definition of a (γ, β) disperser.

Lemma 21 *If G is a (γ, β) disperser and $\Delta(C) \geq \gamma n$, then $\Delta(G(C)) \geq \beta m$.*

References

[1] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992. 11

[2] M. R. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 659–668, 2002. 2

[3] M. Sipser and D. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996. 9

[4] G. Zémor. On expander codes. *IEEE Transactions on Information Theory*, 47(2):835–837, 2001. 9

Introduction to Coding Theory **CMU: Spring 2010**

Notes 9: Expander codes, beginning of list decoding

Mar 31, 2010

Lecturer: Venkatesan Guruswami *Scribe: Yuan Zhou*

1 Distance amplification of codes (continued from last lecture)

1.1 Code construction

Recall some definitions and lemmas introduced from last lecture.

Definition 1 (distance amplified code $G(C)$) *Let $G = (L, R, E)$ be a bipartite graph with $L = [n], R = [m]$, which is D-left-regular and d-right-regular. Let C be a binary linear code of block length $n = |L|$. For $c \in \{0,1\}^n$, define $G(c) \in (\{0,1\}^d)^m$ by*

$$G(c)_j = (c_{\Gamma_1(j)}, c_{\Gamma_2(j)}, \cdots, c_{\Gamma_d(j)}),$$

for $j \in [m]$, where $\Gamma_i(j) \in L$ denotes the i-th neighbor of $j \in R$. Now define the code $G(C)$ as

$$G(C) = \{G(c) | c \in C\}.$$

Since each bit of a codeword $c \in C$ is repeated D times in the associated codeword $G(c) \in G(C)$, we have

Lemma 2 $R(G(C)) = \frac{1}{D} \cdot R(C)$.

To make the (relative) distance of $G(C)$ larger than that of C, we would like to use a special class of graphs to be G, which is defined as follows.

Definition 3 (dispersers) *A bipartite graph $G = (L, R, E)$ is said to be a (γ, β)-disperser if for all subset $S \subseteq L$ with $|S| \geq \gamma n$, we have $|\Gamma(S)| \geq \beta m$.*

The following lemma follows from the definition of dispersers.

Lemma 4 *If G is a (γ, β)-disperser and $\Delta(C) \geq \gamma n$, then $\Delta(G(C)) \geq \beta m$.*

Thus, if we can get a "good" disperser (say, with large β and small γ), then we can use it to amplify the distance of a code. The follow lemma says that such a "good" disperser exists.

Lemma 5 *There exists an explicit (poly-time constructible) $(\gamma, 1-\epsilon)$-disperser with $D = d = \Theta(1/(\gamma\epsilon))$.*

PROOF: Let $G = (L, R, E)$ be the double cover of Ramanujan expander (an $(n, d, \lambda \le 2\sqrt{d})$-expander). I.e. take a Ramanujan graph $H = (V, E_H)$, let L and R be two independent copies of V, and for each edge $(u, v) \in E_H$, make two edges $(u_l, v_r), (v_l, u_r)$ for E.

Now we only need to prove that for each $S \subseteq L$ with $|S| = \gamma n$, we have $\Gamma(S) \ge (1-\epsilon)n$. Fix S, let $T = R \setminus \Gamma(S)$, it suffices to prove that $|T| \le \epsilon n$. By Expander Mixing Lemma, we have

$$\begin{aligned}
& 0 = |\mathrm{E}(S,T)| \ge \frac{d|S||T|}{n} - \lambda\sqrt{|S||T|} \\
\Rightarrow\ & d^2|S||T| \le \lambda^2 n^2 \\
\Rightarrow\ & d^2|T| \le \frac{\lambda^2 n}{\gamma} && (|S| = \gamma n) \\
\Rightarrow\ & |T| \le \frac{(\lambda/d)^2}{\gamma} n \\
\Rightarrow\ & |T| \le \frac{4}{\gamma d} n && (\lambda \le 2\sqrt{d})
\end{aligned}$$

To make $|T| \le \epsilon n$, it suffices that $d \ge 4/(\gamma\epsilon)$. □

Use Tanner code (or any explicit code with positive relative rate and relative distance) as C, plug Lemma 5 into Lemma 2 and Lemma 4, we get the following codes, whose rate-distance tradeoff matches the Singleton bound, up to a constant factor.

Corollary 6 *There are explicit codes of relative distance $(1-\epsilon)$ and relative rate $\Omega(\epsilon)$, over alphabet size $2^{O(1/\epsilon)}$.*

Remark 7 *We compare the codes in Corollary 6 with algebraic geometry codes (with the same relative distance, and the same relative rate up to a constant factor), the latter one has alphabet size $O(1/\epsilon^2)$ (where the current lower bound is $\Omega(1/\epsilon)$). But the combinatorial construction of distance amplification codes is much simpler.*

If we use the code in Corollary 6 as outer code, and concatenate it with constant-size binary codes with relative rate $\Omega(\epsilon^2)$ and relative distance $(1/2 - \epsilon)$ (i.e. random code that matches Gilbert-Varshamov bound), we get a binary code with large distance.

Corollary 8 *There are explicit binary codes of relative distance $(1/2\epsilon)$ and relative rate $\Omega(\epsilon^3)$, which matches the Zyablov bound, up to a constant factor.*

1.2 Decoding algorithm for $G(C)$

Let $G = (L, R, E)$ be a double cover of Ramanujan $(n, d, \lambda \le 2\sqrt{d})$-expander constructed by Lemma 5. (Indeed we need the fact that G is a double cover of Ramanujan expanders, rather than its disperser properties). Let $C \in \{0,1\}^n$ be a binary code that can be decoded against $\gamma_0 n$ errors in linear time (say, Tanner codes).

We address our decoding problem for $G(C) \subseteq \Sigma^n$ (where $\Sigma = \{0,1\}^d$) as follows, for each $y \in \Sigma^n$ such that there exists some $c \in C$ with $\Delta(G(c), y) < (1-\epsilon)n/2$, the decoding algorithm should output $G(c)$. Note that since the distance of $G(C)$ is $(1-\epsilon)n$, if such $c \in C$ exists, it is unique.

We describe the algorithm as follows, and then explain the idea and give the proof of it.

Decoding algorithm for $G(C)$

(1) Compute $c' \in \{0,1\}^n$: for each $i \in L$, set c'_i as the majority vote for the value of i-th bit from its d-neighbors.

(2) Decode c from c' (by the decoder of C).

(3) Output $G(c)$.

The idea of the decoding algorithm is as follows. Note that $\Delta(G(c), y) < (1-\epsilon)n/2$ means that more than a half of coordinates of y are correct (i.e. agree with $G(C)$). Fix a vertex $i \in L$, think of its d neighbors $\Gamma(i)$ are randomly sampled from R (expanders simulate this in a deterministic way). Then, with high probability, more than a half of $y_{\Gamma(i)}$ will be correct. Taking the majority votes (namely c'_i) of the values of c_i recorded by $y_{\Gamma(i)}$, we will be able to recover the real c_i with high probability. Thus $\Delta(c', c)$ is small (hopefully smaller than $\gamma_0 n$), and we are able to decode c from c' by the decoder of C.

Lemma 9 *There are codes of relative rate $\Omega(\epsilon^2)$ such that we can correct a fraction $(1-\epsilon)/2$ of errors in linear time.*

PROOF: Use $G(C)$ and the algorithm described above.

We claim that after the first step in the algorithm, $\Delta(c', c) \le \gamma_0 n$. To see this, let $S = \{i \in L : c'_i \neq c_i\}$, i.e., S be the set of coordinates in which the majority voting get wrong. We only need to prove that $|S| \le \gamma_0 n$. The defintion of S implies that for each $i \in S$, a majority of its neighbors are corrupted, i.e., if we let $T \subseteq R$ be the locations in which y is not corrupted ($|T| \ge (1+\epsilon)n/2$) , then

$$\forall i \in S, |\Gamma(i) \cap T| \le \frac{d}{2}.$$

Thus, by Expander Mixing Lemma, we have

$$\begin{aligned}
& \frac{d|S|}{2} \ge |\mathrm{E}(S,T)| \ge \frac{d|S||T|}{n} - \lambda\sqrt{|S||T|} \\
\Rightarrow\ & \frac{d|S|}{2} \ge \frac{d|S|(1+\epsilon)}{2} - \lambda\sqrt{|S||T|} && (|T| \ge (1+\epsilon)n/2) \\
\Rightarrow\ & \frac{\epsilon d}{2}|S| \le \lambda\sqrt{|S||T|} \\
\Rightarrow\ & \frac{\epsilon^2 d^2}{4}|S| \le \lambda^2|T| \le \lambda^2 n \\
\Rightarrow\ & |S| \le \frac{4(\lambda/d)^2}{\epsilon^2} n \\
\Rightarrow\ & |S| \le \frac{16}{\epsilon^2 d} n && (\lambda \le 2\sqrt{d})
\end{aligned}$$

Thus, to make $|S| \le \gamma_0 n$, it suffices that $d \ge 16/(\gamma_0 \epsilon^2)$.

If we start with a graph G with $d \geq 16/(\gamma_0\epsilon^2)$, in the second step of the algorithm, the decoder finds the correct c, and then outputs the correct $G(c)$. $\square$

Exercise 1

(1) Extend the algorithm described above to correct τ errors and s erasures when $2\tau+s \leq (1-\epsilon)n$.

(2) Use GMD decoding to give binary codes of rate $\Omega(\epsilon^3)$, which are linear-time decodable up to a fraction $(1-\epsilon)/4$ of errors.

1.3 Near-optimal linear time codes

We now discuss how the above scheme can be used to achieve a near-optimal trade-off between rate and relative distance, together with linear time encoding/decoding algorithms.

Theorem 10 *[GI02] For every r, $0 < r < 1$, and all $\epsilon > 0$, there is an explicit family of codes of rate r and relative distance at least $(1-r-\epsilon)$ such that codes in the family can be encoded as well as unique decoded from a fraction $(1-r-\epsilon)/2$ of errors in time linear in the block length.*

Proof Sketch: The construction of the code looks like a combination of concatenation of Tanner codes and Reed-Solomon-based expander codes, but here the expander is used solely for simulation of a random process, rather than for distance amplification. We start with Tanner codes, which has relative rate $1/(1+\gamma)$ and can be decoded against $O(\gamma^3)$ errors in linear time for some small γ (indeed $O(\gamma^2/\log(1/\gamma))$, but $O(\gamma^3)$ is enough for the proof).

Construction of the code. Let $\gamma = \epsilon/4$. Let $G = (L, R, E)$ be a double cover of Ramanujan expanders (as constructed by Lemma 5), with degree $d = \Theta(1/(\beta\epsilon^2))$. Given a message, we encode it by Tanner codes to be C, which is guaranteed to have relative rate $1/(1+\gamma)$ and be able to decoded against a fraction $\beta > O(\gamma^3)$ of errors in linear time. Then we break C into $dr(1+\gamma)$-sized (constant-sized) blocks (say, $n = |L| = |R|$ blocks in total). Let the i-th block be $C^{(i)}$. Then encode each block $C^{(i)}$ with constant-sized Reed-Solomon codes (of relative rate $r(1+\gamma)$ and relative distance $1-r(1+\gamma)$) into $\widetilde{C^{(i)}}$. We see that the block length of each $\widetilde{C^{(i)}}$ is exactly d. We associate each $\widetilde{C^{(i)}}$ with i-th node in L. Then let each node in $j \in R$ collect the tuple

$$\mathcal{C}_j = (\widetilde{C^{(\Gamma_1(j))}}_{\Gamma^{-1}(\Gamma_1(j),j)}, \widetilde{C^{(\Gamma_2(j))}}_{\Gamma^{-1}(\Gamma_2(j),j)}, \cdots, \widetilde{C^{(\Gamma_d(j))}}_{\Gamma^{-1}(\Gamma_d(j),j)}),$$

where $\Gamma_k(j)$ is the k-th neighbor of j, and $\Gamma^{-1}(i,j)$ is the index of j among all the neighbors of i. In English, each node i on the left distributes $\widetilde{C^{(i)}}_k$ to its k-th neighbor. All the nodes on the right collect the distributed letters, and write down from whom each letter comes from.

Finally, let $\mathcal{C} = \mathcal{C}_1\mathcal{C}_2\cdots\mathcal{C}_n$ be the encoded codeword.

Rate. It is easy to see that the rate of the code is the product of the Tanner code and Reed-Solomon code, which is $1/(1+\gamma)\cdot r(1+\gamma) = r$.

Decoding algorithm. Given a (corrupted) codeword $\mathcal{C}'$, the decoding algorithm put each $\mathcal{C}'_j$ on the j-th vertex on the right side of G. Then, similar to the encoding algorithm, the right nodes

distribute all the letters to the left side, and the left vertices collect the letters, getting $\widetilde{C^{(i)}}'$ for each vertex $i \in L$. Then, we run the Reed-Solomon unique decoder decode each $\widetilde{C^{(i)}}'$, getting $C^{(i)'}$. Put all the $C^{(i)'}$ together, we get C'. Run the linear-time decoder for Tanner codes to get the original message.

It is easy to see that the algorithm runs in linear time (in terms of the block length of the codeword). Now we prove that the decoding algorithm output the correct original message when at most $(1-r-\epsilon)/2$ errors occur. Indeed, we only need to prove that at most a fraction β of $\widetilde{C^{(i)}}'$ contains more than fraction $(1-r-\epsilon/4)/2 < (1-r(1+\gamma))/2$ of errors (which is the decoding capacity of Reed-Solomon unique decoder).

Let $T \in R$ be the set of corrupted entries in codeword $\mathcal{C}'$, where $|T| \leq (1-r-\epsilon)n/2$. Let $S \in L$ be the set of vertices on the left who receive more than fraction $(1-r-\epsilon/4)/2$ of corrupted letters (surely from T). Our objective is to prove that $|S| \leq \beta n$. By definition of S, we have

$$|E(S,T)| > \frac{d|S|(1-r-\epsilon/4)}{2}.$$

On the other hand, by Expander Mixing Lemma, we have

$$|E(S,T)| \leq \frac{d|S||T|}{n} + \lambda\sqrt{|S||T|}.$$

Putting two inequalities together, we have

$$\begin{aligned}
& \frac{d|S|(1-r-\epsilon/4)}{2} < \frac{d|S||T|}{n} + \lambda\sqrt{|S||T|} \\
\Rightarrow \quad & \frac{d(1-r-\epsilon/4)}{2} < \frac{d|T|}{n} + \lambda\sqrt{\frac{|T|}{|S|}} \\
\Rightarrow \quad & \frac{d(1-r-\epsilon/4)}{2} < \frac{d(1-r-\epsilon)}{2} + 2\sqrt{\frac{d|T|}{|S|}} \qquad (|T| \leq (1-r-\epsilon)n/2 \text{ and } \lambda \leq 2\sqrt{d}) \\
\Rightarrow \quad & \sqrt{\frac{|T|}{|S|}} > \frac{3\epsilon\sqrt{d}}{16} < \frac{\epsilon\sqrt{d}}{8} \\
\Rightarrow \quad & |S| < \frac{64|T|}{\epsilon^2 d} \leq \frac{64}{\epsilon^2 d} n \leq \beta n. \qquad (|T| \leq n \text{ and } d \geq \frac{64}{\epsilon^2\beta})
\end{aligned}$$

This finishes the proof of the correctness of the algorithm. □

Comments: The near-optimal trade-off (rate r for distance close to $(1-r)$) that almost matches the optimal Singleton bound comes from using the Reed-Solomon codes. The overall scheme can be viewed as using several independent constant-sized RS codes; the role of the expander is then to "spread out" the errors among the different copies of the RS code, so that most of these copies can be decoded, and the remaining small number of errors can be corrected by the left code C. Since only a small fraction of errors needs to be corrected by C it can have rate close to 1 and there is not much overhead in rate on top of the Reed-Solomon code.

The above codes are defined over a large alphabet (whose size depends exponentially on $1/\epsilon$). But they can be concatenated with constant-sized binary codes that lie on the Gilbert-Varshamov bound

to give constructions of binary codes which meet the so-called Zyablov bound. In particular, we can have linear-time binary codes of rate $\Omega(\epsilon^3)$ to correct a fraction $(1/4-\epsilon)$ of errors for arbitrary $\epsilon > 0$ (Exercise 1 (2)). We point the reader to [GI02, GI05] for further details.

2 List decoding – introduction

2.1 Motivation and definition

Recall that a random binary code $C \in \{0,1\}^n$ with size $2^{1-h(d/n)-o(1)}$ has distance d with high probability. For such codes, we can uniquely decode against $\lfloor (d-1)/2 \rfloor$ errors. And we also know that when $\lfloor (d+1)/2 \rfloor$ errors occur, it must be possible that two codewords get to the same string – therefore we cannot decode against $\lfloor (d+1)/2 \rfloor$ errors in the worst case assumption.

But on the other hand, Shannon's theorem tells us that when d errors happens in a random way (in $\mathrm{BSC}_{d/n}$ channel), such a code is decodable against (roughly) d errors with very high probability. Thus, most d-error patterns are decodable. And this reminds us that the worst-case uniquely decodable requirement, which leads to the limitation of $\lfloor (d-1)/2 \rfloor$-error decodability, might be too pessimistic.

Since we still want to talk about worst-case model (in "Hamming world", rather than "Shannon world"), we can only choose to give up the requirement of uniqueness, and introduce the concept of list decoding.

Definition 11 (list decoding problem of a code C) *Fix a code $C \subseteq \Sigma^n$, given $y \in \Sigma^n$, the list decoding problem (against a fraction p of errors) is to output all codewords of C within Hamming distance pn from y.*

Although in list decoding problem, we allow that a message y is "close" to more than one codewords, but we still want the number of "close" codewords small – or just outputting all of them takes a long time. Thus we come up with the following definition, which measures the list-decodability of a code.

Definition 12 ((p, L)-list decodable) *A code $C \subseteq \Sigma^n$ is (p, L)-list-decodable if $\forall y \in \Sigma^n, |B(y, pn) \cap C| \leq L$.*

Remark 13 *$(p, 1)$-list-decodability $\Leftrightarrow$ unique-decodability $\Leftrightarrow \delta(C) > 2p$.*

2.2 Existence of codes that are decodable to a small list

In this section, we justify Definition 11 and Definition 12, by proving the following lemma, which ensures that there exist codes that are decodable to a "small" list.

Lemma 14 *Let $|\Sigma| = q$, $0 < p < 1 - 1/q$. There exists a code $C \subseteq \Sigma^n$ of relative rate $(1 - h_q(p) - 1/L)$ that is (p, L)-list-decodable. In fact, a random $C \subseteq \Sigma^n$ works with high probability.*

PROOF: Let $R = 1 - h_q(p) - 1/L$, pick q^{Rn} codewords from Σ^n independently at random, and let C be the (multi-)set of these codewords. When C is not (p, L)-list-decoable, we know that there exists $y \in \Sigma^n$, and such that $|B(y, pn) \cap C| \geq L + 1$.

Therefore, fix y,

$$\begin{aligned}
\mathbf{Pr}_C[|B(y,pn) \cap C| \geq L+1] &\leq \binom{|C|}{L+1} \Big(\frac{\mathrm{Vol}(B_q(y,pn))}{q^n}\Big)^{L+1} \\
&\leq \Big(\frac{|C|\mathrm{Vol}(B_q(y,pn))}{q^n}\Big)^{L+1} \\
&\leq q^{(R+h_q(p)-1)n(L+1)} \\
&\leq q^{-n(L+1)/L}.
\end{aligned}$$

Thus, by a union bound,

$$\begin{aligned}
\mathbf{Pr}_C[C \text{ is not } (p,L)\text{-list-decodable}] &= \mathbf{Pr}_C[\exists y \in \Sigma^n, |B(y,pn) \cap C| \geq L+1] \\
&\leq q^n q^{-n(L+1)/L} = q^{-n/L}.
\end{aligned}$$

Thus, only with exponentially small probability C fails to be (p, L)-list-decodable. □

The code given by Lemma 14 is not ensured to be linear code. If we want to get linear list-decodable codes, we use the following fact,

Lemma 15 *For each $S \subseteq \mathbb{F}_q^n \setminus \{0\}$ with size $|S| = L$, there always exists a subset $T \subseteq S$ such that $|T| \geq \log_q(L+1)$ and the elements in T are linearly independent.*

Then, using similar proof as in Lemma 14, we get

Lemma 16 *A random linear code $C \subseteq \mathbb{F}_q^n$ of rate $(1 - h_q(p) - \frac{1}{\log_q(L+1)}$ is (p, L)-list-decodable with probability $(1 - q^{\Omega(n)})$.*

Remark 17 *The $\frac{1}{\log_q L}$ slack has been improved to $C_{p,q}/L$ recently.*

On the other hand, assuming L is a large constant, we prove that the rate/distance tradeoff shown in Lemma 14 and Lemma 16 is tight.

Lemma 18 *A code $C \subseteq [q]^n$ of rate $(1 - h_q(p) + \epsilon)$ is not $(p, q^{\Omega_\epsilon(n)})$-list-decodable.*

PROOF: Pick $y \in [q]^n$ at random. Then,

$$\mathbf{E}_y[|B(y,pn) \cap C|] = \frac{|C|\mathrm{Vol}_q(n,pn)}{q^n} \geq |C| q^{(h_q(p)-1-o(1))n} \geq q^{(\epsilon - o(1))n}.$$

Therefore, there exists $y \in [q]^n$, such that $|B(y, pn) \cap C| \geq q^{(\epsilon - o(1))n}$. □

We also remark that for large sized alphabet, say for alphabet with size $q = 2^{1/\gamma}$, the capacity (rate) of list decoding is almost

$$1 - h_q(p) = 1 - (p \log_q(q-1) + \frac{h(p)}{\log q}) \geq 1 - (p + \frac{1}{\log q}) = 1 - p - \gamma.$$

Comparing it to Reed-Solomon codes which matches the Singleton bound, we note that unique decoding of Reed-Solomon code works for at most fraction $(1-R)/2$ of errors (where R is the relative distance). But list decoding (with constant-size list) can deal with $p \to 1-R$ errors.

3 List decoding for Reed-Solomon codes (to be continued)

We have proved the optimal rate/distance tradeoff for list decoding. Thus, the main challenge remains now is to approach the optimal rate of $(1-h_q(p))$ with

- explicit codes,
- efficient list decoding algorithms.

In this section, we will show a list decoding algorithm for Reed-Solomon codes, under a general framework using Johnson bound.

3.1 A general framework of constructing list-decodable codes

We have seen many clever construction of codes with good rate and good distance. One might ask whether we can take the advantage of these codes to get codes with nice list-decodable properties. The Johnson bound should come to mind at this stage, which allow us to say something about the list-decodability just based on the distance of a code. Recall the Johnson bound as follows,

Theorem 19 (Johnson bound) *A q-ary code with block length n, relative distance δ is always $(J_q(\delta), O(n))$-list-decodable, where*

$$J_q(\delta) = \Big(1 - \sqrt{1 - \frac{q}{q-1}\delta}\Big)\Big(1 - \frac{1}{q}\Big).$$

In particular,

$$J_2(\delta) = \frac{1 - \sqrt{1-2\delta}}{2}.$$

Indeed, we have also shown a slightly different version of Theorem 19, as follows.

Theorem 20 *A q-ary code with relative distance δ is always*

$$\Big(\Big(1 - \sqrt{1 - \frac{q\delta(1-1/L)}{q-1}}\Big)\Big(1-1/q\Big), L\Big)\text{-}$$

list-decodable. In particular, a binary code with relative distance δ is

$$\Big(\frac{1 - \sqrt{1-2\delta(1-1/L)}}{2}, L\Big)\text{-}$$

list-decodable.

For large q, we see that a code with relative distance δ is $(1-\sqrt{1-\delta(1-1/L)}, L)$-list-decodable. Thus, for Reed-Solomon codes with rate R and distance almost $(1-R)$, we want to decode it against up to fraction $(1-\sqrt{R})$ of errors.

Remark 21 *Note that since Reed-Solomon codes meet the Singleton bound, $(1-\sqrt{R})$ is the best tradeoff between the fraction of errors can be list decoded and rate, if we work under the framework of Johnson bound (only appeal to distance and Johnson bound).*

3.2 A toy problem

Before we go into the Reed-Solomon list decoding algorithm, we work on the following toy problem as warmup.

The problem. [ALRS99] Fix $p_1(x), p_2(x) \in \mathbb{F}[X]$ as two different degree-k polynomials. With a parameter n ($n > 4k$, n even), the encoding procedure selects n different elements $\alpha_1, \alpha_2, \cdots, \alpha_n \in \mathbb{F}$, chooses a set $S \in [n]$ of size $|S| = n/2$, and computes $y_1, y_2, \cdots, y_n$ as follows,

$$\forall i \in [n], y_i = \begin{cases} p_1(\alpha_i) & i \in S \\ p_2(\alpha_i) & i \in [n] \setminus S \end{cases}.$$

The problem is to design a decoding algorithm, which, given $(\alpha_1, y_1), (\alpha_2, y_2), \cdots, (\alpha_n, y_n)$ output by the encoding procedure, no matter what S the encoding procedure chooses, recovers the two polynomials $p_1(x)$ and $p_2(x)$.

First try – unique decoding. One idea could be to treat $p_1(x)$ as the real message, and all the y_i's computed according to $p_2(x)$ as noise (errors). Then use unique decoder to decode $p_1(x)$, and interpolate $p_2(x)$ from all the pairs (α_i, y_i) that don't agree with $p_1(x)$. But this idea doesn't work, because the codeword $(\alpha_1, y_1), (\alpha_2, y_2), \cdots, (\alpha_n, y_n)$ is equally far away from the message generated solely by $p_1(x)$ and the one generated solely by $p_2(x)$ (i.e., the Hamming distance are all $n/2$). Therefore no unique decoder could figure out whether $p_1(x)$ or $p_2(x)$ is the "unique" original message.

In particular, if we treat $(\alpha_1, y_1), (\alpha_2, y_2), \cdots, (\alpha_n, y_n)$ as a variant of Reed-Solomon code, the Reed-Solomon unique decoder requires at least $(n+k)/2$ agreements between $p_1(x)$ and the message to be decoded. But this is not ensured in the problem.

Solution. We encode the fact that each y_i is either $p_1(\alpha_i)$ or $p_2(\alpha_i)$ as following algebraic statement.

$$\forall i \in [n], (y_i - p_1(\alpha_i))(y_i - p_2(\alpha_i)) = 0.$$

Therefore, if we let

$$Q(x,y) = (y - p_1(x))(y - p_2(x)) = y^2 - (p_1(x) + p_2(x))y + p_1(x)p_2(x) = y^2 - B(x)y + C(x),$$

where $B(x)$ is a degree k polynomial and $C(x)$ is a degree-$2k$ polynomial, we know that $Q(x,y)$ vanishes for each $(x = \alpha_i, y = y_i)$. Thus, our decoding algorithm works as follows.

The algorithm.

(1) By solving a linear system, find a polynomial of form $Q(x,y) = y^2 - B(x)y + C(x)$ such that $B(x)$ is a degree k polynomial, $C(x)$ is a degree-$2k$ polynomial, and for each $i \in [n]$, $Q(\alpha_i, y_i) = 0$.

(2) Factorize $Q(x,y) = (y - f_1(x))(y - f_2(x))$, and output $\{f_1(x), f_2(x)\}$ as $\{p_1(x), p_2(x)\}$.

In the first step of the algorithm, we are always able to find a solution because $(y-p_1(x))(y-p_2(x))$ is a candidate solution. Thus, we need to prove that, on the other hand, we are always able to find out $p_1(x)$ and $p_2(x)$.

Lemma 22 *For any $Q(x,y)$ found in Step (1) of the algorithm, $(y - p_1(x))|Q(x,y)$, and $(y - p_2(x))|Q(x,y)$.*

PROOF: By symmetry, we only need to prove that $(y - p_1(x))|Q(x,y)$. Think of Q as $Q(y)$, a univariate polynomial on y, to prove that $(y - \beta)$ is a factor of $Q(y)$, we only need to prove that $Q(\beta) = 0$. Thus, it suffices to prove $Q(x, p_1(x)) \equiv 0$, which implies $(y - p_1(x))|Q(x,y)$.

Let $R(x) = Q(x, p_1(x))$. We see that $R(x)$ has degree at most $2k$. Thus to prove $R(x) \equiv 0$, we only need to find out $(2k+1)$ roots for $R(x)$. Note that there are $n/2 > 2k$ α_i's with $y_i = p_1(\alpha_i)$. For these α_i's, we have $R(\alpha_i) = Q(\alpha_i, p_1(\alpha_i)) = Q(\alpha_i, y_i) = 0$. And this finishes the proof. □

Lemma 22 implies that $p_1(x), p_2(x) \in \{f_1(x), f_2(x)\}$. Thus, when $p_1(x)$ and $p_2(x)$ are different, the algorithm outputs the correct set.

3.3 List decode Reed-Solomon codes in general

The toy problem actually gives us an approach to decode Reed-Solomon codes into a list of size 2, against $n/2$ errors – requiring $t \geq n/2$ agreements (correct places), while unique decoder can only deal with $(n-k)/2$ errors – requiring $t > (n+k)/2$ agreements.

In general, to decode Reed-Solomon codes into a larger list, we consider the following polynomial reconstruction problem. Given n distinct pairs $(\alpha_i, y_i) \in \mathbb{F}^2$, find all polynomials $f \in \mathbb{F}[X]$ of degree at most k, such that the agreement $f(\alpha_i) = y_i$ holds for $\geq t$ values of $i \in [n]$.

In general, we would like t to be small as possible – so that we can deal with more errors. Unique decoding requires $t > (n+k)/2$. To match the Johnson bound (to decode into constant/$O(n)$-sized list), we want solve the polynomial reconstruction problem for $t > \sqrt{kn}$.

In next lecture, we will see a list decoding algorithm for $t > \sqrt{2kn}$ [Sud97].

References

[ALRS99] Sigal Ar, Richard Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488–511, 1999. 9

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 812–821, New York, NY, USA, 2002. ACM. 4, 6

[GI05] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005. 6

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. 10

Introduction to Coding Theory **CMU: Spring 2010**

Notes 10: List Decoding Reed-Solomon Codes and Concatenated codes

April 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkat Guruswami & Ali Kemal Sinop*

DRAFT

Last class, we saw a toy version of recovering from a mixture of two Reed-Solomon codewords the two polynomials in question. Now we turn to list decoding arbitrary received words with a bounded distance from the Reed-Solomon code using Sudan's [6] algorithm. This algorithm decodes close to a fraction 1 of errors for low rates. Then we will see an improvement by Guruswami and Sudan [4] which list decoded up to the Johnson radius.

1 List Decoding Reed-Solomon Codes

Let C_{RS} be a $[n, k+1, n-k]_q$ Reed-Solomon code over a field $\mathbb{F}$ of size $q \geq n$ with a degree k polynomial $p(X) \in \mathbb{F}[X]$ being encoded as $(p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_n))$. We can reduce the list-decoding problem to a *polynomial reconstruction* problem with agreement parameter t.

Remark 1 *If $t > \frac{n+k}{2}$, then the answer (if it exists) is unique.*

Remark 2 *Recall that Johnson Bound states that a code of distance $k+1$ can be list decoded up to $n - \sqrt{nk}$ errors. Our goal is to try solve this for $t > \sqrt{kn}$.*

However it is an open question whether if one can do it for (say) $t = 0.9\sqrt{kn}$. Another open question is what can be done if we allow super polynomial list size for $t = \frac{n}{k}$.

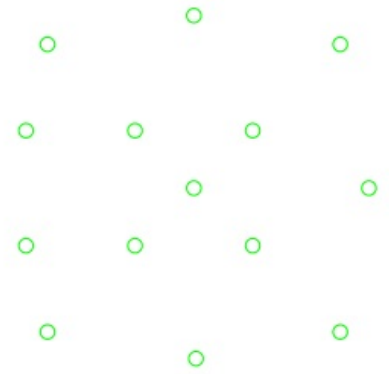
1.1 Polynomial Reconstruction

Given parameters n,t,k and n pairs $(\alpha_i, y_i) \in \mathbb{F}^2$, we want to find all degree $\leq k$ polynomials $f \in F[X]$ such that $\left|\left\{i \middle| f(\alpha_i) = y_i\right\}\right| \geq t$. Here we will present an algorithm due to Sudan [6] for $t \geq \sqrt{2kn}$. Later we will see how to remove $\sqrt{2}$.

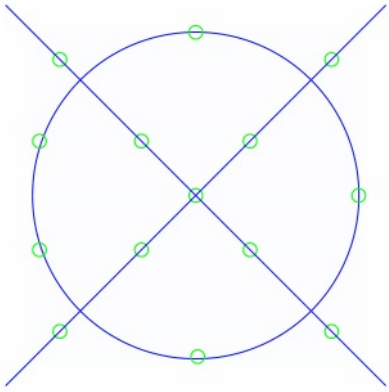
Figure for comparison of relative distances.

1.1.1 Geometric Intuition

Example 1 *Consider the following point configuration on which we want to find all the lines passing through at least 5 points.*



By inspecting the figure, we can see that there are only two such lines, $f_1(x) = x$ and $f_2(x) = -x$:



Claim 3 *Any total degree 4 polynomial $Q(x, y)$ such that $Q(\alpha_i, y_i) = 0$ for all i, must have these two lines as factors.*

PROOF: *Both lines must intersect $Q(x, y) = 0$ in at least 5 distinct points. Let $Q(\alpha_j, f_i(\alpha_j)) = 0$ for 5 different points. Since $Q(x, f_i(x))$ is a degree-4 polynomial in x, we have $Q(x, f_i(x)) \equiv 0$. Therefore $y - f_i(x) \mid Q(x, y)$.* $\square$

Hence a possible factorization for these points is given by $Q(x, y) = (x - y)(x + y)(x^2 + y^2 - 1)$.

1.1.2 Sudan's Algorithm

Armed with the above intuition, consider the following algorithm:

1. Find a low degree $Q(x, y) \not\equiv 0 \in \mathbb{F}[X, Y]$ such that $Q(\alpha_i, y_i) = 0$ for all i.
2. Find all factors of the form $y - f(x)$ of $Q(x, y)$. Output those with agreement at least t.

There are three questions we need to answer in order to show that this algorithm is correct. First we need to make sure that such Q exists. Second we need to find out when $y - f(x)$ is a factor of $Q(x, y)$. Third is how we can find such factors efficiently. We will touch upon a randomized algorithm for this at the end of this section.

Lemma 4 *Given positive integers d_x, d_y, $n < (d_x + 1)(d_y + 1)$ and n points $\{(\alpha_i, y_i)\}_{i=1}^n \subset \mathbb{F}^2$, the following holds: There exists $Q(X, Y) \not\equiv 0 \in \mathbb{F}[X, Y]$ such that $Q(\alpha_i, y_i) = 0$ for all i and $\deg_x(Q) \leq d_x, \deg_y(Q) \leq d_y$. Moreover such Q can be found by solving a linear system.*

PROOF: Notice that Q with $\deg_x(Q) \leq d_x, \deg_y(Q) \leq d_y$ has $(d_x + 1)(d_y + 1)$ many coefficients. For each i, $Q(\alpha_i, y_i) = 0$ is a homogeneous linear equation in terms of Q's coefficients. There are n such equations and $(d_x + 1)(d_y + 1)$ many unknowns. For $(d_x + 1)(d_y + 1) > n$, a non-zero solution is guaranteed to exist. □

Having made sure that Q exists, we need to answer when $y - f(x) \mid Q$.

Lemma 5 *Given a polynomial $f(x) \in \mathbb{F}[X]$ with $f(\alpha_i) = y_i$ for t different points $(\alpha_i, y_i) \in \mathbb{F}^2$: $Q(\alpha_i, y_i) = 0$, if $t > \deg_x(Q) + \underbrace{\deg_x(f)}_{k} \deg_y(Q)$, then $y - f(x) \mid Q(x, y)$.*

PROOF: Let $R(x) := Q(x, f(x))$. We have $R(\alpha_i) = Q(\alpha_i, f(\alpha_i)) = Q(\alpha_i, y_i) = 0$. Therefore $R(x)$ has t different roots. But $R(x)$ has degree at most $\deg_x(Q) + \deg_y(Q) \deg_x(f) < t$, which implies $R(x) \equiv 0 \implies Q(x, f(x)) \equiv 0$. Hence $y - f(x) \mid Q(x, y)$. □

Remember, $Q(x, y)$ can have at most $\deg_y(Q)$ many different factors of the form $y - f(x)$. Hence $\deg_y(Q) = \ell$ is our list size. By Lemma 4, we can take $d_x = \lfloor \frac{n}{\ell} \rfloor$. By Lemma 5, we need $t > \frac{n}{\ell} + \ell k$. This is minimized for $\ell = \sqrt{\frac{n}{k}}$, which yields $t > 2\sqrt{nk}$.

In order to get rid of a factor of $\sqrt{2}$, we notice that the important quantity is $d_x + k \cdot d_y$. As long as all non-zero monomials $q_{ij} x^i y^j$ of $Q(x, y)$ has $i + kj < t$, Lemma 5 will hold.

Definition 6 ($(1, k)$-weighted degree) *For a polynomial $Q(X, Y) \in \mathbb{F}[X, Y]$, its $(1, k)$-weighted degree is defined to be the maximum value of $i + kj$ over all non-zero monomials $x^i y^j$ of Q.*

Lemma 7 *Given positive integer D and $kn < \binom{D+2}{2}$ points $\{(\alpha_i, y_i)\}_{i=1}^n \subset \mathbb{F}^2$, the following holds: There exists $Q(X, Y) \not\equiv 0 \in \mathbb{F}[X, Y]$ such that $Q(\alpha_i, y_i) = 0$ for all i and $(1, k)$-weighted degree of Q is D. Moreover such Q can be found by solving a linear system.*

PROOF: The number of Q's coefficients is:

$$\sum_{j=0}^{\lfloor D/k \rfloor} (D - jk + 1) = (D + 1) \left(\left\lfloor \frac{D}{k} \right\rfloor + 1 \right) - \frac{k}{2} \left\lfloor \frac{D}{k} \right\rfloor \left(\left\lfloor \frac{D}{k} \right\rfloor + 1 \right) \geq \frac{(D + 1)(D + 2)}{2k}$$

Rest of this argument is exactly the same with Lemma 4. □

Lemma 8 *Given a degree-k polynomial $f(x) \in \mathbb{F}[X]$ with $f(\alpha_i) = y_i$ for t different points $(\alpha_i, y_i) \in \mathbb{F}^2 : Q(\alpha_i, y_i) = 0$, if $(1,k)$-weighted degree of Q is $< t$, then $y - f(x) \mid Q(x,y)$.*

PROOF: As usual, let $R(x) := Q(x, f(x))$ and note that $R(x)$ has t different roots. Now degree of R is at most $\max_{ij:q_{ij}\neq 0} i + \deg_x(f)j \leq \max_{ij:q_{ij}\neq 0} i + kj < t$. Here q_{ij} is the coefficient of monomial $x^i y^j$ of $Q(x,y)$. Hence $y - f(x) \mid Q(x,y)$. □

Putting these together, we obtain:

Theorem 9 (6) *Given parameters n,t,k and n pairs $(\alpha_i, y_i) \in \mathbb{F}^2$, if $t > \sqrt{2kn}$, we can find all degree $\leq k$ polynomials $f \in F[X]$ such that $\left|\{i | f(\alpha_i) = y_i\}\right| \geq t$ in time polynomial in n, k and $|\mathbb{F}|$. Moreover there are at most $\sqrt{2n/k}$ many such f's.*

PROOF: Let $D \leftarrow t - 1$. Then $\frac{(D+1)(D+2)}{2k} = \frac{t^2+t}{2k} > \frac{2nk+1}{2k} > n$. By Lemma **??**, a Q with $(1,k)$-weighted degree $\leq t-1$ exists. By Lemma 8, we know that if f has $\geq t$-agreement, $y - f(x) \mid Q(x,y)$. Assuming that we can find such factors in polynomial time, we are done. □

1.1.3 Finding factors of $Q(X,Y)$ of the form $Y - f(X)$

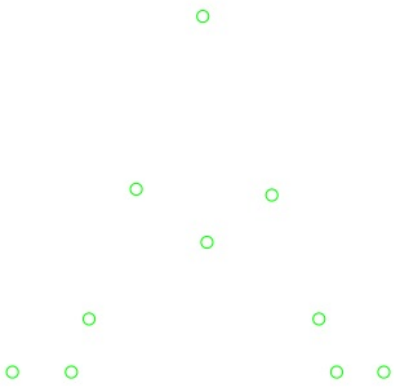
We will describe a simple randomized algorithm given in [1, Section 4.2]. We are given a bivariate polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$ with $q = |\mathbb{F}|$ and we want to find factors $y - f(x)$ with $\deg_x(f) \leq k$. If we can find an algorithm that either finds such polynomial $f(x)$ or concludes that none exists, we are done. We can run this algorithm, and if it finds $f(x)$, we can recurse on the quotient $Q(x,y)/(y - f(x))$.

Let $E(X) \in \mathbb{F}[X]$ be an irreducible polynomial with degree $\geq k+1$. An explicit choice is $E(x) = x^{q-1} - \gamma$ where γ is a generator of $\mathbb{F}$. As we have seen in Homework 2, this is irreducible. Given such $E(x)$, we can view $Q(x,y) \mod E(x)$ as a polynomial, $\widetilde{Q}(Y) \in \widetilde{\mathbb{F}}[Y]$ with coefficients in the extension field $\widetilde{\mathbb{F}} = \mathbb{F}[X]/(E(X))$ of degree $q-1$ over $\mathbb{F}$. The problem reduces to finding roots of $\widetilde{Q}(Y)$ in $\widetilde{\mathbb{F}}$, which can be done using Berlekamp's algorithm in time polynomial in $\deg\left(\widetilde{Q}\right)$ and q.

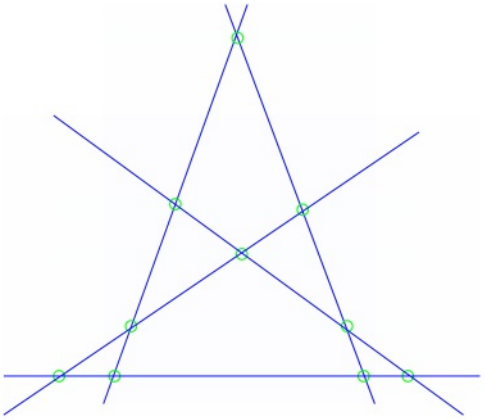
2 Method of Multiplicities

In this section, we will remove the factor $\sqrt{2}$ and obtain a list decoding algorithm with $t > \sqrt{kn}$ due to Guruswami and Sudan [4]. First we will see an example showing that we can't hope to improve the above parameters.

Example 2 *Consider the following point configuration of $n = 10$ points. For $k = 1$ (lines), having $t \geq \sqrt{nk}$ implies $t > 3$. So we want to find lines passing through ≥ 4 points.*



Next figure shows set of all lines going through at least 4 *points:*



If we want to output all these 5 *lines,* Q *must have total degree at least* 5*. But the agreement parameter* $t = 4$ *is smaller than* 5*, so* $Q(x, f(x))$ *(for one of these lines;* $y = f(x)$*) might not be* $\equiv 0$*.*

Note that in this example, each point has two lines crossing. Thus any $Q(x, y)$ *containing these lines as factors must also contain each point twice.*

Now we will try to understand what it means for $Q(x, y)$ to contain each point twice.

Example 3 *If* $Q(X, Y) \in \mathbb{F}[X, Y]$ *has* r *zeroes at point* $(0, 0)$*, then it has no monomials of total weight less than* r*.*

Since we can translate Q by any $(\alpha, \beta) \in \mathbb{F}^2$, we can generalize the above example to a formal definition:

Definition 10 (Multiplicity of zeroes) *A polynomial* $Q(X, Y)$ *is said to have a zero of multiplicity* $w \geq 1$ *at a point* $(\alpha, \beta) \in \mathbb{F}^2$ *if* $Q_{\alpha,\beta} := Q(x + \alpha, y + \beta)$ *has no monomial of total degree less than* w*.*

Armed with this definition and an idea on how to put additional constraints on Q, we will look at what happens to Lemmas 7 and 8.

Corollary 11 *Given positive integers $D, w_1, \ldots, w_n$ such that $k\sum_i \binom{w_i+1}{2} < \binom{D+2}{2}$, and points $\{(\alpha_i, y_i)\}_{i=1}^n \subset \mathbb{F}^2$, the following holds: There exists $Q(X,Y) \not\equiv 0 \in \mathbb{F}[X,Y]$ such that $Q(x,y)$ has a zero of multiplicity w_i at (α_i, y_i) for all i, and $(1,k)$-weighted degree of Q is D. Moreover such Q can be found by solving a linear system.*

Proof: Notice that coefficient of $x^i y^j$ in $Q_{\alpha,\beta}$ is

$$\sum_{i' \geq i, j' \geq j} \binom{i'}{i}\binom{j'}{j}\alpha^{i'-i}\beta^{j'-j}q_{i',j'}.$$

Hence we can express w_i-multiplicity condition as $\binom{w_i+1}{2}$ homogeneous linear equations. By previous calculation, we know that the number of Q's coefficients is $\geq \frac{(D+1)(D+2)}{2k}$. Thus a non-zero solution exists if $n\sum_i \binom{w_i+1}{2} \leq \binom{D+2}{2}$ $\square$

Lemma 12 *Suppose $f(\alpha) = \beta$ if $Q(x,y)$ has r-zeroes at (α, β). Then $(x-\alpha)^r \mid Q(x, f(x))$.*

Proof: We have

$$R(x) := Q(x, f(x)) = Q_{\alpha,\beta}(x-\alpha, f(x)-\beta) = Q_{\alpha,\beta}(x-\alpha, f(x)-f(\alpha))$$

Here $Q_{\alpha,\beta}$ has no monomials of degree $< r$. Also $(x-\alpha) \mid (f(x)-f(\alpha))$. For any non-zero monomial $x^i y^j$ of $Q_{\alpha,\beta}$, we have $(x-\alpha)^{i+j} \mid (x-\alpha)^i (f(x)-f(\alpha))^j$. Since $i+j \geq r$, we have $(x-\alpha)^r \mid Q_{\alpha,\beta}(x-\alpha, f(x)-\beta) = Q(x, f(x))$. $\square$

Lemma 13 *If a degree k polynomial f has $f(\alpha_i) = y_i$ for $\geq t$ values of i and $\sum_{i:f(\alpha_i)=y_i} w_i > D$, then $y - f(x) \mid Q(x,y)$ assuming Q has $(1,k)$ weighted degree $\leq D$.*

Proof: If $f(\alpha_i) = y_i$ and $Q(x,y)$ had a multiplicity of w_i zeroes at (α_i, y_i) then $(x-\alpha_i)^{w_i} \mid Q(x, f(x))$ by previous lemma. Therefore $\prod_{i:f(\alpha_i)=y_i}(x-\alpha_i)^{w_i} \mid Q(x, f(x))$. If $(1,k)$ weighted degree of $Q(x,y)$ is D, then $Q(x, f(x)) \equiv 0$ and $y - f(x) \mid Q(x,y)$. $\square$

Putting all together, we obtain:

Theorem 14 *The algorithm finds all polynomials $f(x)$ such that $\sum_{i:f(\alpha_i)=y_i} w_i > \sqrt{2k\sum_i \binom{w_i+1}{2}}$. In particular, if $w_i = w$, then we can decode f up to $n-t$ errors with $t > \sqrt{nk\left(1+\frac{1}{w}\right)}$.*

Corollary 15 (4) *For a Reed-Solomon code of rate R, we can list decode up to $1-\sqrt{R}$ fraction of errors in time polynomial in the block length and the field size.*

3 Soft Decoding

One reason we picked each w_i individually is because we "encode" likelihood information. One might think of w_i's representing a confidence value on the received symbol and placing more emphasis on smaller values. This connection has been made formal by Koetter-Vardy [5], where it was shown how to choose weights optimally based on channel observations and channel transition probabilities.

One important point is that this application requires dealing with non-negative real weights. This is made precise in the following Theorem which appears in [2, Chapter 6]:

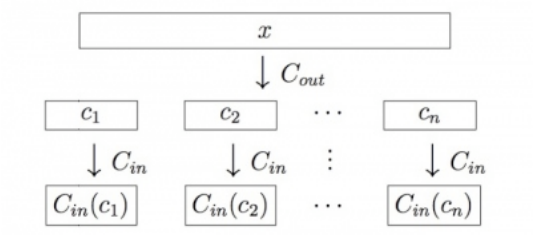
Theorem 16 *For all non-negative rationals* $\{w_{i,\alpha}\}_{i\in[n],\alpha\in\mathbb{F}}$ *and non-negative real* ϵ*, there is a* poly $\left(n, |\mathbb{F}|, \frac{1}{\epsilon}\right)$*-time algorithm to find all degree* k *polynomials* f *such that*

$$\sum_{i=1}^{n} w_{i,f(\alpha_i)} \geq \sqrt{k \sum_{i,\alpha} w_{i,\alpha}^2} + \epsilon w_{\max}.$$

4 List Decoding of Binary Concatenated Codes

The main shortcoming of the above methods is that the alphabet size is polynomial in block length. We will use concatenated codes to reduce alphabet size.

Assume we are given a Reed-Solomon code as our outer code $C_{\text{out}} : \mathbb{F}_2^n \to \mathbb{F}_2^{n/R}$ with rate R and an inner code $C_{\text{in}} : \mathbb{F}_2^m \to \mathbb{F}_2^{m/r}$ with rate $r = \text{rate}(C_{\text{in}})$. Consider the basic composition:



This code has rate $R_0 = R/r$. First we will start with the most naive idea and work our way up. Our main goal in this section is to get a binary code which is close to Zyablov's bound.

4.1 Uniquely Decoding Inner Codes

Consider uniquely decoding each inner codes and then applying list decoding algorithm on the outer code. There will be $\geq \frac{\text{h}^{-1}(1-r)}{2}$ fraction of errors if an inner block is decoded to a wrong code word. The list decoding capacity of C_{out} is $1 - \sqrt{\frac{R}{r}} = 1 - \sqrt{R_0}$. Hence this algorithm can list decode up to $\left(1 - \sqrt{R_0}\right) \frac{\text{h}^{-1}(1-r)}{2}$-fraction of errors.

However this is quite bad, as we picked up a factor of $\frac{1}{2}$ that we were trying to avoid by using list decoding at the first place.

Lemma 17 *Given positive reals $0 < r, R < 1$, there exists a binary code which can be list-decoded up to $\left(1 - \sqrt{\frac{R}{r}}\right) \frac{\mathrm{h}^{-1}(1-r)}{2}$ fraction of errors in polynomial time.*

4.2 List Decoding Inner Codes

Instead of uniquely decoding inner codes, consider list decoding them, by –say– brute force. Recall that by Johnson Bound, any binary code can be list-decoded up to $\mathrm{h}^{-1}(1-r-\epsilon)$-fraction of errors with a list size of $\ell = \ell(\epsilon) \leq O(1/\epsilon)$.

A slight complication arises on how to apply Reed-Solomon list-decoding on a message where each symbol is itself another list. During list-decoding algorithm, we only assumed all (α_i, y_i) pairs were distinct, and everything worked fine even if $\alpha_i = \alpha_j$ for different i, j as long as $y_i \neq y_j$. Using this observation, we can apply our list decoding algorithm on the set $\{(\alpha_i, y_{ij})\}_{i,j}$. This will return us a list of messages which agrees with $t > \sqrt{kN}$ points where $N < n\ell$. Therefore this method will give us list decoding up to $\left(1 - \sqrt{\frac{R}{r}}\right) \mathrm{h}^{-1}(1-r-\epsilon)$ fraction of errors.

Lemma 18 *Given positive reals $0 < r, R < 1$ and for any real $0 < \epsilon < r$, there exists a binary linear code which can be list-decoded up to $\left(1 - \sqrt{\frac{R}{r}}\right) \mathrm{h}^{-1}(1-r-\epsilon)$ fraction of errors.*

In order to get a binary code list decodable up to $\frac{1-\gamma}{2}$ fraction of errors, we can take $\ell = 1/\gamma^2$ and $r = O(\gamma^2)$, which implies $\epsilon = O(\gamma^2)$. By taking $R = r^5$, we obtain:

Theorem 19 *For any real $0 < \gamma < 1$, there exists a linear binary code of rate $\Omega(\gamma^6)$ list-decodable up to $\frac{1-\gamma}{2}$ fraction of errors with a list size of $O\left(\frac{1}{\gamma^3}\right)$.*

Remark 20 *Compare this to the optimal parameter of rate $\Omega(\gamma^2)$ and list size $O(1/\gamma^2)$.*

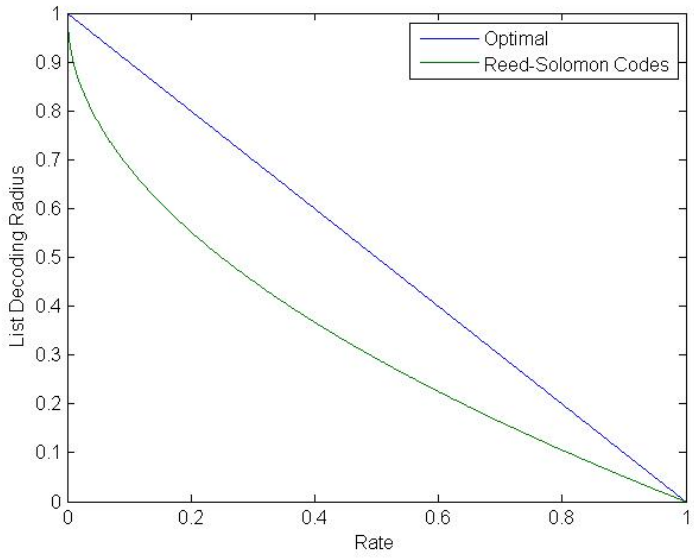
4.3 Weighted List Decoding Inner Codes

One room for improvement to the previous algorithm is that, instead of treating each code in the list equally, assigning different weights. If we remember Theorem 16, we can obtain a list decoding with a (weighted) agreement of $\sum_{i=1}^{n} w_{i,f(\alpha_i)} \geq \sqrt{k \sum_{i,\alpha} w_{i,\alpha}^2} + \epsilon w_{\max}$. In order to minimize this, we need to ensure that the weights of inner codes around a point should decay in ℓ_2 norm rapidly. Although it is not known how to get this for general binary codes, Hadamard codes have this property.

5 Going beyond Reed-Solomon codes & Johnson radius

For Reed-Solomon codes, we showed that one can efficiently list decode up to a radius (fraction of errors) equal to $1 - \sqrt{R}$. However we know that list decoding up to a fraction $1 - R - \epsilon$ of errors

is possible non-constructively. In this section, we will construct better variants of Reed-Solomon codes. The code we will describe was given by Guruswami and Rudra [3].



The main difficulty in Reed-Solomon codes is that we need to be able to deal with *any* pattern of e errors. Instead, we will "pack" parts of codeword together and decrease the error patterns we have to handle considerably.

5.1 Folded Reed-Solomon Codes

Recall that a Reed-Solomon code $\mathrm{RS}_{\mathbb{F},\mathbb{F}^*}[n,k]$ gives an encoding of a degree-k polynomial $f(X) \in \mathbb{F}[X]$ as

$$\begin{aligned} f(x) &\mapsto \left(f(\alpha)\big|_{\alpha \in \mathbb{F}^*} \right) \\ &\mapsto \left(f(1), f(\gamma), \ldots, f(\gamma^{n-1}) \right) \end{aligned}$$

where $n = |\mathbb{F}| - 1 = q - 1$ and γ is a generator of $\mathbb{F}$.

Folded Reed-Solomon code $\mathrm{FRS}^{(s)}_{\mathbb{F}}[k]$ (a code over $\mathbb{F}^s$) is the s-folded version of the above (for convenience assume $s \mid n$):

$$f(x) \mapsto \left(\begin{bmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{s-1}) \end{bmatrix}, \begin{bmatrix} f(\gamma^s) \\ f(\gamma^{s+1}) \\ \vdots \\ f(\gamma^{2s-1}) \end{bmatrix}, \ldots, \begin{bmatrix} f(\gamma^{n-s}) \\ f(\gamma^{n-s+1}) \\ \vdots \\ f(\gamma^{n-1}) \end{bmatrix}, \right).$$

The received word looks like following:

	1	γ	γ^s	$\dots$	γ^{n-s+1}
1	y_1	y_{s+1}			y_{n-s+2}
γ	y_2	y_{s+2}			$\vdots$
γ^2	y_3	y_{s+3}			$\vdots$
$\vdots$				$\ddots$	
γ^{s-1}	y_s	$\cdots$			y_n

$N = n/s$ many columns

If we think of Reed-Solomon decoding as interpolating a polynomial over a plane (which led to the $\sqrt{R}$ bound on agreement required), it might seem possible to decode with agreement fraction about $R^{s/(s+1)}$ by interpolating in $(s+1)$-dimensions.

Problem 21 *We want to find a polynomial* $Q(X, Y_1, Y_2, \dots, Y_s) \not\equiv 0 \in \mathbb{F}[X, Y_1, Y_2, \dots, Y_s]$ *such that* $Q(\gamma^{is}, y_{is+1}, y_{is+2}, \dots, y_{(i+1)s}) = 0$ *for* $0 \leq i < n/s$.

6 References

1. S. Ar, R. Lipton, R. Rubinfield and M. Sudan, "Reconstructing algebraic functions from mixed data," *SIAM Journal on Computing,* vol. 28, no. 2, pp. 488–511, 1999.

2. V. Guruswami, "List decoding of error-correcting codes," *Lecture Notes in Computer Science,* no. 3282, Springer, 2004.

3. V. Guruswami and A. Rudra, "Explicit Codes Achieving List Decoding Capacity: Error-Correction With Optimal Redundancy," *IEEE Transactions on Information Theory* 54(1): 135-150 (2008).

4. V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Transactions on Information Theory,* vol. 45, pp. 1757–1767, 1999.

5. R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Transactions on Information Theory*, 49(11): 2809-2825 (2003)

6. M. Sudan, "Decoding Reed-Solomon codes beyond the error-correction bound," *Journal of Complexity,* vol. 13, no. 1, pp. 180–193, 1997.

Introduction to Coding Theory **CMU: Spring 2010**

Notes 11: List Decoding Folded Reed-Solomon Codes

April 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

At the end of the previous notes, we defined folded Reed-Solomon codes, parameterized by an integer folding parameter $s \geq 1$, as follows.

Definition 1 (s-folded RS code) *Let $\mathbb{F}$ be a field of size q whose nonzero elements are $\{1, \gamma, \ldots, \gamma^{n-1}\}$ for $n = q - 1$. Let $s \geq 1$ be an integer which divides n. Let $1 \leq k < n$ be the degree parameter.*

The folded Reed-Solomon code $\mathrm{FRS}_{\mathbb{F}}^{(s)}[k]$ is a code over alphabet $\mathbb{F}^s$ that encodes a polynomial $f \in \mathbb{F}[X]$ of degree k as

$$f(X) \mapsto \left(\begin{bmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{s-1}) \end{bmatrix}, \begin{bmatrix} f(\gamma^s) \\ f(\gamma^{s+1}) \\ \vdots \\ f(\gamma^{2s-1}) \end{bmatrix}, \ldots, \begin{bmatrix} f(\gamma^{n-s}) \\ f(\gamma^{n-s+1}) \\ \vdots \\ f(\gamma^{n-1}) \end{bmatrix} \right) . \tag{1}$$

Observe that the folded RS code is a code of block length $N = n/s$ and rate $R = k/n$ (which is the same as the original, unfolded Reed-Solomon code).

1 List decoding FRS codes

Now suppose such a codeword was transmitted and we received a string in $\mathbf{y} \in (\mathbb{F}^s)^N$ which we view in matrix form as

$$\begin{array}{cccc} y_1 & y_{s+1} & & y_{n-s+2} \\ y_2 & y_{s+2} & & \vdots \\ y_3 & y_{s+3} & & \vdots \\ & & & \ddots \\ y_s & \cdots & & y_n \end{array} \tag{2}$$

We would like to recover a list of all polynomials $f \in \mathbb{F}[X]$ of degree k whose folded RS encoding (1) agrees with $\mathbf{y}$ in at least t columns, for some agreement parameter t. The goal would be to give an algorithm for as small a t as possible, as this corresponds to list decoding up to $n - t$ errors. Note that we can solve this problem for $t \geq \sqrt{R}N$ by simply unfolding the received word and using the algorithm for Reed-Solomon codes. Our hope is to do better by exploiting the fact that the agreements have some structure: when a column is correct we know the correct values of *all* s values in the column. In other words, the number of errors patterns we have to worry about is smaller and they have some structure, which could be potentially be exploited to correct from a larger number of errors. This is in fact what we will do, exploiting in a crucial way the algebraic structure of the folding.

As in the Reed-Solomon list decoding algorithm, we will use interpolation to fit a low-degree nonzero polynomial Q through the data. The gain will come by interpolating in more than two dimensions. This enables working with a smaller degree and offers the hope of leading to an algorithm that works with smaller agreement. The fact that an entire column is correct when there is no error will be used to argue that the interpolated polynomial Q and the message polynomial f must obey some identity (the higher-dimensional analog of the identity $Q(X, f(X)) = 0$ from the Reed-Solomon case). The algebraic crux is to argue that this identity suffices to pin down f to a small list of possibilities, and to find this list efficiently.

1.1 Interpolation step

We will describe a decoding algorithm that can be viewed as a higher-dimensional generalization of the Welch-Berlekamp algorithm. The algorithm will interpolate a "linear" polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$ which has degree 1 in the Y_i's through certain $(s+1)$-tuples. We thank Salil Vadhan for pointing out this variant of the algorithm, which is simpler to describe. The algorithm described in the paper [1] uses higher degrees in Y_i's as well as multiplicities in the interpolation. This leads to a stronger bound (as we will mention later), but the simpler "linear polynomial" based algorithm suffices for the main message of this lecture. Jumping ahead, this message is the explicit construction of codes of rate R which can be list decoded in polynomial time from a fraction $1 - R - \epsilon$ of errors, for any desired $\epsilon > 0$.

Given a received word as in (2) we will interpolate a nonzero polynomial

$$Q(X, Y_1, Y_2, \ldots, Y_s) = A_0(X) + A_1(X)Y_1 + A_2(X)Y_2 + \cdots + A_s(X)Y_s$$

with the degree restrictions $\deg(A_i) \leq D - 1$ for $i = 1, 2, \ldots, s$ and $\deg(A_0) \leq D + k - 1$ for a suitable degree parameter. The number of monomials in a polynomial Q with these degree restrictions equals $Ds + D + k$. The polynomial $Q \in \mathbb{F}[X, Y_1, \ldots, Y_s]$ must satisfy the interpolation conditions

$$Q(\gamma^{is}, y_{is+1}, y_{is+2}, \cdots, y_{(i+1)s}) = 0 \quad \text{for } i = 0, 1, \ldots, n/s - 1 \ . \tag{3}$$

Provided $Ds + D + k > \frac{n}{s} = N$, or in other words

$$D > \frac{N - k}{s + 1} \ , \tag{4}$$

such a nonzero polynomial Q must exist, and can be found in polynomial time by solving a homogeneous linear system. The following lemma shows that any such polynomial Q gives an algebraic condition that the message polynomials $f(X)$ we are interested in list decoding must satisfy.

Lemma 2 *If $f \in \mathbb{F}[X]$ is a polynomial of degree k whose FRS encoding (1) agrees with $\mathbf{y}$ in at least t columns for $t \geq D + k$, then*

$$Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) = 0 \ . \tag{5}$$

PROOF: Define $R(X) = Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X))$. Due to the degree restrictions on Q, the degree of $R(X)$ is easily seen to be at most $D + k - 1$. If the FRS encoding of f agrees with $\mathbf{y}$ in the i'th column (for some $i \in \{0, 1, \ldots, N - 1\}$), we have

$$f(\gamma^{is}) = y_{is+1}, \quad f(\gamma^{is+1}) = y_{is+2}, \quad \cdots \quad , f(\gamma^{is+s-1}) = y_{(i+1)s} \ .$$

Together with the interpolation conditions (3), this implies

$$R(\gamma^{is}) = Q(\gamma^{is}, f(\gamma^{is}), f(\gamma^{is+1}), \cdots, f(\gamma^{is+s-1})) = 0 \ .$$

Hence R has at least t zeroes. Since $\deg(R) \leq D + k - 1$, if $t \geq D + k$, we must have $R = 0$. $\square$

1.2 Root-finding step

The question that arises now is how restrictive is Equation (5) in terms of pinning down the number of possible solutions $f(X)$ to a small (polynomial in $|\mathbb{F}|$) number? For purposes of illustration and ease of notation, let us focus on the $s = 2$ case, so that we need to find the list of all degree k solutions $f(X)$ to

$$Q(X, f(X), f(\gamma X)) = 0 \ . \tag{6}$$

For this part we will have use the fact that $\gamma \in \mathbb{F}$ is not arbitrary but is a primitive element. Indeed if $\gamma = 1$, then for the polynomial $Q(X, Y, Z) = Y - Z$, *every* polynomial f will satisfy $Q(X, f(X), f(X)) = 0$. Likewise, if $\gamma = -1$, then every polynomial $f(X) = g(X^2)$ will satisfy $Q(X, f(X), f(-X)) = 0$ for $Q(X, Y, Z) = Y - Z$. This argument shows that if the order of γ is small, then there can be too many ($|\mathbb{F}|^{\Omega(k)}$) degree k solutions.

We will next prove that for primitive γ, the number of f of degree less than $q - 1$ (and hence also number with degree at most k) satisfying (6) is at most q. For this we need two lemmas.

Lemma 3 *If* $\deg(f) < q - 1$*, then* $f(\gamma X) = f(X)^q \mod (X^{q-1} - \gamma)$.

Proof: We have $X^q \equiv \gamma X \pmod{X^{q-1} - \gamma}$, and therefore

$$f(X^q) \equiv f(\gamma X) \pmod{X^{q-1} - \gamma} \ .$$

For $f \in \mathbb{F}[X]$, we have $f(X^q) = f(X)^q$. Therefore $f(X^q) \equiv f(\gamma X) \pmod{X^{q-1} - \gamma}$. Since the degree of f is less than $q - 1$, the remainder of $f(X)^q \bmod X^{q-1} - \gamma$ must equal $f(\gamma X)$. $\square$

The next lemma (in fact a more general statement) was on problem set 2.

Lemma 4 *The polynomial* $X^{q-1} - \gamma$ *is irreducible over* $\mathbb{F}_q$ *when* γ *is a primitive element of* $\mathbb{F}_q$.

Denote $E(X) = X^{q-1} - \gamma$. Denote by L the extension field $L = \mathbb{F}[X]/(E(X))$ (it is a field since $E(X)$ is irreducible), and for a polynomial $f \in \mathbb{F}[X]$, let $\bar{f}$ denote its residue (modulo $E(X)$) in L.

Theorem 5 *Given a nonzero* $Q \in \mathbb{F}[X, Y_1, Y_2]$ *which is linear in* Y_1, Y_2*, the number of polynomials* $f \in \mathbb{F}[X]$ *of degree less than* $q - 1$ *such that* $Q(X, f(X), f(\gamma X)) = 0$ *is at most* q*. Moreover the list of all such polynomials can be found in time polynomial in* q *and the degree of* Q*.*

Proof: Factor out the largest power of $E(X)$ that divides Q, so that $Q(X, Y_1, Y_2) = E(X)^a Q_1(X, Y_1, Y_2)$ for some $a \geq 0$ and polynomial Q_1 that is not divisible by $E(X)$. Define the polynomial $S \in L[Y]$ as

$$S(Y) = Q_1(X, Y, Y^q) \mod E(X) \ .$$

Since Q_1 has degree in Y_1, Y_2 less than q (in fact it is linear in Y_1, Y_2), and it is not divisible by $E(X)$, we have $S \neq 0$. Clearly if $Q(X, f(X), f(\gamma X)) = 0$, then $Q_1(X, f(X), f(\gamma X)) = 0$, so certainly $Q_1(X, f(X), f(\gamma X)) \mod E(X) = 0$. By Lemma **??**, this implies $Q_1(X, f(X), f(X)^q) \mod E(X) = 0$, i.e., $S(\bar{f}) = 0$. Thus $\bar{f}$ must be a root of S, which has degree at most q in Y. This implies that there are most q solutions for $\bar{f}$. Since f has degree less than $q-1$, f can be uniquely recovered from $\bar{f}$, and hence there are at most q solutions f to $Q(X, f(X), f(\gamma X)) = 0$, as desired.

The claim about the running time follows since the polynomial S can be computed efficiently from Q, and there are efficient root finding algorithms known over large extension fields. (One can either have a randomized algorithm running in time polynomial in the degree and log of the field size, or a deterministic algorithm running in time polynomial in the degree, log of the field size, *and* the characteristic of the field. In our case, even the deterministic runtime will be polynomial in q.) □

The above approach generalizes readily to the larger variate case. Any solution $f \in \mathbb{F}[X]$ to $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) = 0$ will have its residue $\bar{f}$ as a root of (the nonzero polynomial) $S(Y) = Q(X, Y, Y^q, \cdots, Y^{q^{s-1}}) \mod E(X)$. Therefore there will be at most q^{s-1} such polynomials f and they can all be found in polynomial (in q) time.

1.3 Decoding guarantee

Combining Lemma 2 and Theorem 5, and recalling the requirement (4) on the degree parameter D we conclude that we have a polynomial time list decoding algorithm for $\mathrm{FRS}_{\mathbb{F}}^{(s)}[k]$ to find all message polynomials whose encoding has agreement t with an input $\mathbf{y} \in (\mathbb{F}^s)^N$ provided

$$t > \frac{N-k}{s+1} + k = \frac{N}{s+1} + \frac{s}{s+1}k \ . \tag{7}$$

The fractional agreement τ required (in terms of the rate) is

$$\tau = \frac{t}{N} > \frac{1}{s+1} + \frac{k}{N}\frac{s}{(s+1)} = \frac{1}{s+1} + (sR)\frac{s}{s+1} \tag{8}$$

since $N = n/s$ and $R = k/n$.

Our goal was to decode from an agreement fraction $\tau \approx R$, but the factor s multiplying R implies that we require $\tau \approx sR$ which is much worse for large s. In particular we do not get anything meaningful for $R > 1/s$!

Remark 6 *By allowing higher degree terms in the Y_i's in the interpolated polynomial and using multiplicities in the interpolation (as in the improved Reed-Solomon list-decoding algorithm, extended to the multivariate case in the obvious manner), one can improve the above guarantee and decode from agreement fraction*

$$\tau \approx (sR)^{s/(s+1)} \ . \tag{9}$$

Note that this quantity is the geometric mean of $1, \underbrace{R, R, \cdots, R}_{s \text{ times}}$, whereas the agreement required in (8) was the arithmetic mean of these values (which is in general larger). Recall that for Reed-Solomon codes ($s = 1$) this was also exactly the case: we reduced the agreement required from $\frac{1+R}{2}$ to $\sqrt{R}$ to get an algorithm to list decode up to the Johnson radius.

Remark 7 (Parvaresh-Vardy) *The decoding guarantee (9) was obtained by Parvaresh and Vardy [2]. Their construction was somewhat different. Instead of the folding operation, they encoded a degree k message polynomial $f \in \mathbb{F}[X]$ by the evaluation of f and $(s-1)$ other polynomials $f_1, f_2, \ldots, f_{s-1}$ which are chosen to be algebraically related to $f(X)$ in the following way:*

$$f_i(X) = f(X)^{h^i} \mod E(X)$$

for $i = 1, 2, \ldots, s-1$, where $E(X)$ is an arbitrary irreducible polynomial of degree $k+1$. Note that the rate of such a code is R_0/s where R_0 is the rate of the original RS code (in particular this construction can never have rate exceed $1/s$). We used $f(\gamma^{i-1}X)$ in place of $f_i(X)$ in the above description in preparation for the optimal result which we discuss next.

2 Improving the decoding radius and correcting $1 - R - \epsilon$ errors

The idea to improve the agreement fraction required is to use folded codes with folding parameter m for $m \gg s$, but still only do $s+1$-variate interpolation. The key gain will be that each agreement location will now lead to many zeroes for the polynomial $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X))$ which ultimately ensures that this polynomial must equal zero even for a much smaller value of the agreement parameter t.

Specifically, let us consider the code $\mathrm{FRS}_{\mathbb{F}}^{(m)}[k]$, where as before $|\mathbb{F}| = q$, $n = q-1$, $m|n$, and the block length of the code equals $N = nm$. We will still interpolate a polynomial $Q \in \mathbb{F}[X, Y_1, \ldots, Y_s]$ in $s+1$ variables, but now instead of n/s tuples, we will interpolate through $(n/m)(m-s+1) = N(m-s+1)$ tuples given by

$$(\gamma^{im+j}, y_{im+j+1}, \ldots, y_{im+j+s}) \quad 0 \le i < n/m, \quad 0 \le j < m - s \ .$$

That is, for each column of m symbols, we interpolate through the $(m - s + 1)$ windows of s consecutive symbols. The rationale behind this is that if we have an agreement in location i, i.e.,

$$(y_{im+1}, y_{im+2}, \ldots, y_{(i+1)m}) = (f(\gamma^{is}), f(\gamma^{is+1}), \ldots, f(\gamma^{im+m-1}))$$

then we get $m - s + 1$ zeroes

$$Q(\gamma^{im+j}, f(\gamma^{im+j}), \ldots, f(\gamma^{im+j+s-1})) = 0 \quad j \in \{0, 1, 2, \ldots, m-s\} \ .$$

Thus t agreements leads to $t(m-s+1)$ zeroes for $R(X) = Q(X, f(X), \ldots, f(\gamma^{s-1}X))$. In place of (7), we get the new condition for successful decoding

$$t(m-s+1) > \frac{N(m-s+1)}{s+1} + \frac{s}{s+1}k$$

or equivalently

$$\tau = \frac{t}{N} > \frac{1}{s+1} + \frac{s}{s+1}\frac{m}{m-s+1}R \tag{10}$$

(recalling that $k = Rn = RNm$).

We now have our desired optimal trade-off between τ and R: picking $s > 1/\epsilon$ and $m \geq s^2$, the above condition (10) for successful decoding is met if

$$\tau \geq R + \epsilon \ .$$

Let us look at the parameters of the code. The alphabet size in $q^m \approx n^m = (Nm)^m$ which is $(N/\epsilon^2)^{O(1/\epsilon^2)}$ for the choice $m \approx 1/\epsilon^2$. The bound on list size we obtain is at most $q^{s-1} \approx (Nm)^{O(s)} = (N/\epsilon^2)^{O(1/\epsilon)}$ for the choice $s \approx 1/\epsilon$. The running time of the decoding algorithm is also dominated by the root-finding step, which takes $q^{O(s)}$ time as it has to find roots of a polynomial of degree at most q^s over an extension field of size at most q^q.

We can thus state our main theorem as follows.

Theorem 8 (Explicit codes achieving list decoding capacity) *For every $\epsilon > 0$ and $0 < R < 1$, there is a family of folded Reed-Solomon codes which have rate at least R and which can be list decoded up to a fraction $1 - R - \epsilon$ of errors in time $(N/\epsilon^2)^{O(\epsilon^{-1})}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\epsilon^2)^{O(1/\epsilon^2)}$.*

Remark 9 (Improvement using higher degree and multiplicities in interpolation) *Similar to Remark 6, by allowing higher degree terms in the Y_i's in the interpolated polynomial and using multiplicities in the interpolation, one can improve the above guarantee and decode from agreement fraction*

$$\tau \approx \left(\frac{mR}{m - s + 1} \right)^{s/(s+1)} \ . \tag{11}$$

instead of the arithmetic mean $\frac{1}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1}$. For details see the original paper [1].

References

[1] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008. 2, 6

[2] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005. 5