Information and Coding Theory **Autumn 2017**

Lecture 12: November 6, 2017

Lecturer: Madhur Tulsiani

Recall: We were looking at codes of the form $C : \mathbb{F}_p^k \to \mathbb{F}_p^n$, where p is prime, k is the message length, and n is the block length of the code. We also saw C as a set in $\mathbb{F}_p^n$ and defined the distance of the code as

$$\Delta(C) := \min_{x,y \in C, x \neq y} \{\Delta(x,y)\}$$

where $\Delta(x,y)$ is the Hamming distance between x and y. We proved that a code C can correct t errors iff $\Delta(C) \geq 2t+1$.

1 Linear codes

Definition 1.1 (Linear Codes) *A code $C \subseteq \mathbb{F}_p^n$ is a linear code C is a subspace of $\mathbb{F}_p^n$. We will also think of a C as a linear encoding $C : \mathbb{F}_p^k \to \mathbb{F}_p^n$, satisfying $C(\alpha u + v) = \alpha C(u) + C(v)$ for all $u, v \in \mathbb{F}_p^k$ and $\alpha \in \mathbb{F}_p$.*

Since a linear encoding is a linear map from a finite dimensional vector space to another, we can write it as a matrix of finite size. That is, there is a corresponding $G \in \mathbb{F}_p^{n \times k}$ s.t. $C(x) = Gx$ for all $x \in \mathbb{F}_p^k$. If the code has nonzero distance, then the rank of G must be k (otherwise there exist $x, y \in \mathbb{F}_p^k$ such that $Gx = Gy$). Hence, the null space of G^T has dimension $n-k$. This defines another useful matrix, known as the parity check matrix of the code.

Definition 1.2 (Parity Check Matrix) *Let $b_1, \ldots, b_{n-k}$ be a basis for the null space of G^T corresponding to a linear code C. Then $H \in \mathbb{F}_p^{(n-k) \times n}$, defined by*

$$H^T = \left[\; b_1 \; \middle| \; b_2 \; \middle| \; \ldots \; \middle| \; b_{n-k} \; \right]$$

is called the parity check matrix of C.

Since $G^T H^T = 0 \Leftrightarrow HG = 0$, we have $(HG)x = 0$ for all $x \in \mathbb{F}_p^k$, i.e., $Hz = 0$ for all $z \in C$. Moreover, since the columns of H^T are a basis for the null-space of G^T, we have that

$$z \in C \;\Leftrightarrow\; Hz = 0\,.$$

So the parity check matrix gives us a way to quickly check a codeword, by checking the parities of some bits of z (each row of H gives a parity constraint on z). Also, one can equivalently define a linear code by either giving G or the parity check matrix H.

The distance of linear codes, can also be characterized in terms of the number of non-zero entries in any $z \in C \setminus \{0\}$.

Exercise 1.3 *For $z \in \mathbb{F}_p^n$, let $wt(z) = |\{i \in [n] \mid z_i \neq 0\}|$. Prove that for a linear code C*

$$\Delta(C) \;=\; \min_{z \in C} wt(z)\,.$$

1.1 Hamming Code

Consider the following code from $\mathbb{F}_2^4$ to $\mathbb{F}_2^7$, known as the Hamming Code.

Example 1.4 *Let $C : \mathbb{F}_2^4 \to \mathbb{F}_2^7$, where*

$$C(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4, x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4).$$

Note that each element of the image is a linear function of the x_i's, i.e., one can express C with matrix multiplication as follows:

$$C(x_1, x_2, x_3, x_4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Example 1.5 *The parity check matrix of our example Hamming Code is:*

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

Note that the i^{th} column is the integer i in binary. One can easily check that $HG = 0$.

Now suppose $z = (z_1, \ldots, z_7)^T$ is our codeword and we make a single error in the i^{th} entry. Then the output codeword with the error is

$$z + e_i = \begin{bmatrix} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_7 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

and $H(z + e_i) = Hz + He_i = He_i = H_i$, the i^{th} column of H, which reads i in binary. So this is a very efficient decoding algorithm just based on parity checking. Since the Hamming code (C) can correct at least $t = 1$ errors, we must have that $\Delta(C) \geq 2t + 1 = 3$. Verify that the distance is exactly 3 using the following characterization of distance for linear codes. One can generalize the Hamming code to larger message and block lengths, we can create a parity matrix $H \in \mathbb{F}_p^{(n-k)\times n}$, where the i^{th} column reads i in binary.

2 Elementary bounds on codes

We consider a few bounds showing how much redundancy is *necessary* for any code with a given distance.

2.1 Hamming bound

We now show an optimality bound on the size of the code, starting with the case of distance-3 codes and then generalizing to distance-d codes.

Theorem 2.1 *Let $C : \mathbb{F}_2^k \to \mathbb{F}_2^n$ be any distance-3 code, i.e., $\Delta(C) = 3$. Then*

$$|C| = 2^k \leq \frac{2^n}{n+1}$$

Proof: For each $z \in C$, let $B(z)$ be the ball of size $n + 1$ consisting of z and the n elements in $\mathbb{F}_2^n$ (not in C), each at distance 1 from z. Then the balls formed by the codewords in C are disjoint, if $B(z)$ and $B(z')$ intersect, then $\Delta(z, z') \leq 2$ by triangle inequality. For each codeword $z \in C$, we have $|B(z)| = n + 1$ codes, so $|C|(n + 1) \leq 2^n$. ■

Note that our example hamming code from $\mathbb{F}_2^4$ to $\mathbb{F}_2^7$ satisfied $|C| = 2^4 = \frac{2^7}{8}$, so it was an optimal distance-3 code. Generalizing to distance-d codes, we have:

Theorem 2.2 (Hamming Bound) *Let $C : \mathbb{F}_2^k \to \mathbb{F}_2^n$ be any distance-d code, i.e., $\Delta(C) = d$. Then*

$$|C| = 2^k \leq \frac{2^n}{vol\left(B_{\lfloor \frac{d}{2} \rfloor}\right)}$$

where $vol\left(B_{\lfloor \frac{d}{2} \rfloor}\right) = \sum_{i=1}^{\lfloor \frac{d}{2} \rfloor} \binom{n}{i}$ is the number of codes at distance at most $\lfloor \frac{d}{2} \rfloor$ from any fixed codeword $z \in C$.

Remark 2.3 *The Hamming bound also gives us a bound on the rate of the code in terms of entropy (recall: the rate of the code is $\frac{k}{n}$). Let $d = \delta n$ for $\delta \leq \frac{1}{2}$. Since $\sum_{i=1}^{l} \binom{n}{i} \leq 2^{nH(\frac{l}{n})}$ for $l \leq \frac{n}{2}$, we have:*

$$\frac{k}{n} \leq 1 - H(\delta/2) + o(1).$$

2.2 Singleton bound

While the Hamming bound is a good bound for codes over small alphabets ($\mathbb{F}_p$), the following is a better bound when the alphabet size is large.

Theorem 2.4 (Singleton Bound) *Let $C : \mathbb{F}_p^k \to \mathbb{F}_p^n$ be a distance-d code. Then*

$$d \leq n - k + 1.$$

Proof: Consider the map $\Gamma : \mathbb{F}_p^k \to \mathbb{F}_p^{n-d+1}$, where $\Gamma(x)$ is the first $n - d + 1$ coordinates of $C(x)$. Then for all $x \neq y$, we have $\Gamma(x) \neq \Gamma(y)$, since $\Delta(C(x), C(y)) \geq d$. Therefore, $|img(\Gamma)| = |\mathbb{F}_p^{n-d+1}| \geq |\mathbb{F}_p^k|$. ■

We will next see Reed-Solomon codes, which achieve the singleton bound.

3 Reed-Solomon codes

We now look at Reed-Solomon codes over $\mathbb{F}_p$. These are optimal codes which can achieve a very large distance. However, they have a drawback that they need $p \geq n$.

Definition 3.1 (Reed-Solomon Code) *Assume $p \geq n$ and fix $S = \{a_1, \ldots, a_n\} \subseteq \mathbb{F}_p$, distinct s.t. $|S| = n$. For each message $(m_0, \ldots, m_{k-1}) \in \mathbb{F}_p^k$, consider the polynomial $P(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}$. Then the Reed-Solomon Code is defined as:*

$$C(m_0, \ldots, m_{k-1}) = (P(a_1), \ldots, P(a_n)).$$

Remark 3.2 *Reed-Solomon Codes can again be encoded using "inner codes" to create "concatenated codes", which can work with smaller p.*

Let's compute the distance of the Reed-Solomon Code:

Claim 3.3 $\Delta(C) \geq n - k + 1$.

Proof: Consider $C(m_0, \ldots, m_{k-1})$ with $P(x) = m_0 + m_1 x + \cdots + m_{k-1}x^{k-1}$ and $C(m'_0, \ldots, m'_{k-1})$ with $P'(x) = m'_0 + m'_1 x + \cdots + m'_{k-1}x^{k-1}$, where we assume $(m_0, \ldots, m_{k-1}) \neq (m'_0, \ldots, m'_{k-1})$ and hence $P \neq P'$. Then $P - P'$ is a non-zero polynomial of degree at most $k-1$ and has at most $k-1$ roots, i.e., P and P' agree on at most $k-1$ points. This implies that $C(m_0, \ldots, m_{k-1}) = (P(a_1), \ldots, P(a_n))$ and $C(m'_0, \ldots, m'_{k-1}) = (P'(a_1), \ldots, P'(a_n))$ differ on at least $n - k + 1$ places. Hence, $\Delta(C) \geq n - k + 1$. ■

Remark 3.4 *If $n = 2k$, then for $d = k + 1$, we can correct $\lfloor \frac{k}{2} \rfloor$ errors using the Reed-Solomon Code, and this is optimal (by the singleton bound). Moreover, the Reed-Solomon Code is a linear code:*

$$C(m_0, \ldots, m_{k-1}) = \begin{bmatrix} 1 & a_1 & a_1^2 & \ldots & a_1^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_n & a_n^2 & \ldots & a_n^{k-1} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{k-1} \end{bmatrix}$$

3.1 Unique decoding of Reed-Solomon codes

If the codewords output by the Reed-Solomon code did not contain any errors, we could simply use Lagrange interpolation to recover the messages. However, we must be able to handle noise, and the trick is to work with a hypothetical *error-locator polynomial* telling us where the error is.

Definition 3.5 (Error-locator Polynomial) *An error-locator polynomial, $E(x)$, is a polynomial which satisfies*

$$\forall i \in [n] \quad E(a_i) = 0 \Leftrightarrow y_i \neq P(a_i)$$

for all $i = 1, \ldots, n$, where y_i is the i^{th} element of the output Reed-Solomon Code obtained with noise.

Note that this directly implies that $y_i \cdot E(a_i) = P(a_i) \cdot E(a_i)$ for all $a_i \in S$. Let's denote Q as $Q(x) := P(x) \cdot E(x)$. The following algorithm by Berlekemp and Welch [WB86] finds the polynomials E and Q defined above, using these to to decode a message with at most $\lfloor \frac{\Delta(C)-1}{2} \rfloor = \lfloor \frac{n-k}{2} \rfloor$ errors.

Note that the message $(m_0, \ldots, m_{k-1})$ is identified with the polynomial $P(x) = \sum_{i=0}^{k-1} m_i x^i$. Let $(P(a_1), \ldots, P(a_n))$ be the original codeword and let $y = (y_1, \ldots, y_n)$ be it's corrupted version (we only know y). The distance between them is at most t.

Unique decoding for Reed-Solomon codes
Input: $\{(a_i, y_i)\}_{i=1,\ldots,n}$

1. Find $E, Q \in \mathbb{F}_p[x]$ such that $E \not\equiv 0$, $\deg(E) \leq t$, $\deg(Q) \leq k - 1 + t$

$$\forall i \in [n] \quad Q(a_i) = y_i \cdot E(a_i)\,.$$

2. Output $\frac{Q}{E}$.

We first observe that Step 1 in the algorithm can be done by solving linear system. Let $E(x) = e_0 + e_1 x + \cdots + e_t x^t$ and $Q(x) = q_0 + q_1 x + \cdots + q_{k-1+t} x^{k-1+t}$. Then for each given (a_i, y_i), the equation $Q(a_i) = y_i E(a_i)$ is linear in variables $e_0, \ldots, e_t$ and $q_0, \ldots, q_{k-1+t}$. Note that such system is homogeneous and hence it always has a trivial solution. We need to show that there is a solution with nonzero E.

Lemma 3.6 *There exists (E, Q) that satisfies the conditions in Step 1 of the algorithm.*

Proof: Let $I = \{i \in [n] \mid P(a_i) \neq y_i\}$ and $E^* = \prod_{i \in I}(x - a_i)$ (in particular, $E^* \equiv 1$ if I is empty). Let $Q^* = P \cdot E^*$. Then for all $i \in [n]$, we have $y_i E^*(a_i) = P(a_i) \cdot E^*(a_i) = Q^*(a_i)$. ■

If there is a unique nonzero solution to the linear system in Step 1, then Step 2 outputs the correct polynomial. But in general there can be more than one such solution. The following lemma guarantees the correctness of Step 2.

Lemma 3.7 *For any two solutions (Q_1, E_1) and (Q_2, E_2) that satisfy the conditions in Step 1,*

$$\frac{Q_1}{E_1} = \frac{Q_2}{E_2}.$$

Proof: It suffices to show $Q_1 E_2 = Q_2 E_1$. Indeed, since they satisfy the equation $Q(a_i) = y_i E(a_i)$ for each $i \in [n]$, we have

$$(Q_1 E_2)(a_i) = y_i E_1(a_i) E_2(a_i) = (E_1 Q_2)(a_i)$$

for each $i \in [n]$. It implies that they must be the same as there are n zeros of $Q_1 E_2 - Q_2 E_1$ but its degree is at most $k + 2t - 1 \leq k + 2\lfloor \frac{n-k}{2} \rfloor - 1 \leq n - 1$. ■

References

[WB86] L.R. Welch and E.R. Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470. URL: http://www.google.com/patents/US4633470. 5