Information and Coding Theory **Autumn 2022**

Lecture 18: December 1, 2022

Lecturer: Madhur Tulsiani

1 Error-correcting codes for the Hamming model

We now begin our discussion of codes for the Hamming model of errors, where the errors are adversarial in nature, instead of being introduced by a stochastic channel. When sending n symbols $x_1, x_2, \ldots, x_n$ from $\mathcal{X}^n$ across a channel, up to t of these symbols may be corrupted (but not lost) in some arbitrary manner. There is no way to know which of the symbols have been corrupted, and which symbols were transmitted correctly. We wish to design encoding and decoding schemes, which can recover the intended transmission $x = (x_1, \ldots, x_n))$ from up to t errors. The number t will be treated as a parameter, and the goal will be to design codes which can correct as many (adversarial) errors as possible.

Remark 1.1. *You may have noticed that we implicitly assumed that the input alphabet $\mathcal{X}$ for the channel is equal to the output alphabet $\mathcal{Y}$. Throughout our treatment of the adversarial error model, we will indeed assume that the input x and the output y for the channel, are both length-n sequences over the same alphabet $\mathcal{X}$. Moreover, we will also think of the message space as being $\mathcal{X}^k$, so that a code $C \subseteq \mathcal{X}^n$ with $|C| = |\mathcal{X}|^k$ can be identified (via an arbitrary bijection) with the encoding map $\mathsf{Enc} : \mathcal{X}^k \to \mathcal{X}^n$. As in the last lecture, if $\mathsf{Enc}(w) = x$ is transmitted and $y \in \mathcal{X}^n$ is received (with at most t errors) we will restrict our focus to on finding x. Since the codes will design will be linear, it will be simple to recover w from x.*

The parameter which governs how many errors a code can correct is known as the *distance* of the code. Before defining distance of a code, we recall that the Hamming distance of two finite sequences $x, y \in \mathcal{X}^n$ for any finite $\mathcal{X}$, is the number of positions in which the sequences differ. Denoting the Hamming distance of x and y by $\Delta(x, y)$, we can write

$$\Delta(x, y) \ := \ \{i \in [n] \mid x_i \neq y_i\} \ .$$

Exercise 1.2. *Check that Hamming distance defines a metric, i.e., it is symmetric, $\Delta(x, y) > 0$ for $x \neq y$, and it satisfies the triangle inequality*

$$\Delta(x, y) \ \leq \ \Delta(x, z) + \Delta(z, y) \, .$$

We now define the distance of a code, as the Hamming distance between the closest pair of codewords.

Definition 1.3. *Let $C \subseteq \mathcal{X}^n$ be a code. We define the distance of a code $\Delta(C)$ as*

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

The distance can be used to understand the number of errors one can correct. Note that there are no probabilities in the error correcting model. Thus, we take the meaning of "correcting" y to finding the closest $x_0 \in C$ i.e., $x_0 = argmin_{z \in C}(\Delta(y,z))$. The questions is if this correctly recovers the x that was sent (and corrupted to y by at most t errors).

Proposition 1.4. *A code $C \subseteq \mathcal{X}^n$ can correct t errors if and only if $\Delta(C) \geq 2t+1$.*

Proof: First, we prove the reverse direction. If $\Delta(C) \geq 2t+1$, then $\frac{\Delta(C)}{2} > t$. For each codeword $x \in C$ let define the Hamming ball of radius r as

$$B(x,r) \;=\; \{z \in \mathcal{X}^n \mid \Delta(x,z) \leq r\}\;.$$

In particular consider the case when we let $r = t$. Notice that for two distinct $x, x' \in C$, we must have $B(x,t) \cap B(x',t) = \emptyset$, since otherwise, $\Delta(x,x') \leq 2t < \Delta(C)$ by triangle inequality. Now, suppose a codeword x is corrupted in t positions to the string y. By the above argument, $B(x,t)$ is the *unique* Hamming ball around a codeword, in which y is contained. By symmetry of the Hamming distance, given y, we can decode y to the unique $x_0 \in C$ such that $x_0 \in B(y,t)$.

Now, we prove the forward direction. Suppose the codeword $x \in C$ is corrupted in t bits to y i.e., $\Delta(x,y) = t$. Since we can correct up to t errors, we know that we can find $x_0 \in C$ such that $x_0 = argmin_{z \in C}(\Delta(y,z))$. Moreover, we want x_0 to be equal to x. Thus, we have that

$$\delta(z,y) \;>\; \delta(x,y) \;=\; t \quad \forall z \in C\,.$$

However, if we have $\Delta(x,z) = \Delta(C)$ for codewords $x, z \in C$, we can always find y such that $\delta(x,y) = t$ and $\delta(z,y) = \Delta(C) - t$. Combining it with the above, we get $\Delta(C) - t > t$, which implies $\Delta(C) \geq 2t+1$. ∎

1.1 Basics of finite fields

The codes we will design will be linear codes over some finite field $\mathbb{F}_q$. We recall below some of the properties of finite fields that we will need. As before, we will restrict to the case when q is a prime number, so that $\mathbb{F}_q = \{0, \ldots, q-1\}$, with addition and multiplication defined modulo q.

- $\mathbb{F}_q^n$ is a vector space over the field $\mathbb{F}_q$. Note this is *not* an inner product space.
- **Fermat's little theorem:** $a^q \equiv a \mod q$, for all $a \in \{0, \ldots, q-1\}$.

- The set of all polynomials in a single variable X, over $\mathbb{F}_q$, is defined as

$$\mathbb{F}_q[X] \;:=\; \left\{ c_0 + c_1 \cdot X + \cdots + c_{q-1} \cdot X^{q-1} \;\mid\; c_0, \ldots, c_{q-1} \in \mathbb{F}_q \right\}.$$

  Note that the degree (highest power of x with a non-zero coefficient) is never more than $q-1$ (why?) We will also use the notation $\mathbb{F}_q^{\leq d}[X]$ to denote univariate polynomials in X, with degree at most d

- A polynomial of degree at most d, which is not identically zero, has at most d roots.

- **Lagrange interpolation:** Given *distinct* $a_1, \ldots, a_{d+1} \in \mathbb{F}_q$ and any $b_1, \ldots, b_{d+1} \in \mathbb{F}_q$, the *unique* polynomial f with degree at most d, satisfying $f(a_i) = b_i$ for all $i \in [d+1]$, is given by

$$f(x) \;=\; \sum_{i=1}^{d+1} b_i \cdot \prod_{j \neq i} \left(\frac{x - a_j}{a_i - a_j} \right).$$

Distance of linear codes over finite fields. Recall that a linear code $C \subseteq \mathbb{F}_q^n$ was defined to be a subspace of $\mathbb{F}_q^n$. 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 1.5. *For $z \in \mathbb{F}_q^n$, let $\mathsf{wt}(z) = |\{i \in [n] \mid z_i \neq 0\}|$. Prove that for a linear code C*

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

2 Reed-Solomon codes

We now consider an important family of (linear) codes known as the Reed-Solomon codes. These are optimal codes which can achieve a very large distance. However, they have a drawback that they need the field size q to be at least as large as the block-length n.

Definition 2.1 (Reed-Solomon Code)**.** *Assume $q \geq n$ and fix $S = \{a_1, \ldots, a_n\} \subseteq \mathbb{F}_q$, distinct s.t. $|S| = n$. For each message $(m_0, \ldots, m_{k-1}) \in \mathbb{F}_q^k$, consider the polynomial $f_m(X) = m_0 + m_1 \cdot X + \cdots + m_{k-1} X^{k-1}$. Then the Reed-Solomon Code is defined by its encoding as:*

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

Alternatively, we can also define the code directly as the subspace

$$C \;=\; \left\{ (f(a_1), \ldots, f(a_n)) \mid f \in \mathbb{F}_q^{\leq (k-1)}[X] \right\}.$$

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

Claim 2.2. $\Delta(C) \geq n - k + 1$.

Proof: We use the characterization from Exercise 1.5. Let $(f(a_1), \ldots, f(a_n)) \in C$ be a codeword for some $f \in \mathbb{F}_q^{\leq (k-1)}[X]$. We have

$$\mathsf{wt}(f) \;=\; \mathsf{wt}((f(a_1), \ldots, f(a_n))) \;=\; |\{i \in [n] \mid f(a_i) \neq 0\}| \,.$$

Since f has degree at most $k-1$, the number of points where it's zero is at most $k-1$ points, which gives $\mathsf{wt}(f) \geq n - (k-1)$. ∎

Remark 2.3. *The Reed-Solomon Code is a linear code, as can be seen from the encoding map*

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

2.1 Unique decoding of Reed-Solomon codes

Given a codeword of Reed-Solomon code of the form $(f(a_1), \ldots, f(a_n))$ for some $f \in \mathbb{F}$, if the values at t of the points a_i were corrupted, but *we knew* the locations of the corruptions, we can always recover f by Lagrange interpolation, as long as the number of correct values $n - t$ is at least k. The important idea in the decoding algorithm is to note that the information about the location of the errors can also be represented via a low-degree polynomial.

Let $(f(a_1), \ldots, f(a_n))$ be a codeword, and let $(y_1, \ldots, y_n)$ be a received word with at most t errors. We define the error locator polynomial $e(X)$ to a polynomial of degree at most t that satisfies

$$y_i \neq f(a_i) \quad \Rightarrow \quad e(a_i) = 0 \,.$$

Note that e needs to have degree at most t, by Lagrange interpolation. Also note that we may not know f, and hence may not know the error-locator polynomial e. However, the algorithm will try to find the polynomial e, as well as the correct f corresponding to the codeword, as part of the decoding procedure.

The above definition implies that

$$y_i \cdot e(a_i) \;=\; f(a_i) \cdot e(a_i) \quad \forall a_i \in S \,.$$

Let g the polynomial $g(X) := f(X) \cdot e(X)$. The following algorithm by Welch and Berlekemp [WB86] finds the polynomials e and g 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.

Unique decoding for Reed-Solomon codes

Input: $\{(a_i, y_i)\}_{i=1,\ldots,n}$

1. Find $e, g \in \mathbb{F}_q[X]$ such that $E \not\equiv 0$, $\deg(e) \leq t$, $\deg(g) \leq k-1+t$

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

2. Output $\frac{g}{e}$.

We first observe that Step 1 in the algorithm can be implemented by solving system of linear equations. Let $e(X) = e_0 + e_1 \cdot X + \cdots + e_t \cdot X^t$ and $g(x) = g_0 + g_1 \cdot X + \cdots + g_{k-1+t} \cdot X^{k-1+t}$. Then for each given (a_i, y_i), the equation $g(a_i) = y_i \cdot e(a_i)$ is linear in variables $e_0, \ldots, e_t$ and $g_0, \ldots, g_{k-1+t}$. 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 2.4. *There exists (E, Q) that satisfies the conditions in Step 1 of the algorithm.*

Proof: Let $I = \{i \in [n] \mid f(a_i) \neq y_i\}$ and $e^* = \prod_{i \in I}(X - a_i)$ (we take $e^* \equiv 1$ if I is empty). Let $g^* = f \cdot e^*$. Then for all $i \in [n]$, we have $y_i e^*(a_i) = f(a_i) \cdot e^*(a_i) = g^*(a_i)$. Also, $e^* \not\equiv 0$ by construction, and satisfies $\deg(e^*) = |I| \leq t$. Also, since $\deg(f) \leq k-1$, we gave $\deg(g^*) \leq \deg(f) + \deg(e^*) \leq k-1+t$. ■

If the above polynomials e^*, g^* were a unique nonzero solution to the linear system in Step 1, then Step 2 outputs the correct polynomial, since $g^*/e^* = f$. But in general there can be more than one such solution. The following lemma guarantees the correctness of Step 2.

Lemma 2.5. *For any two solutions (g_1, e_1) and (g_2, e_2) that satisfy the conditions in Step 1,*

$$\frac{g_1}{e_1} = \frac{g_2}{e_2}.$$

Proof: It suffices to show $g_1 \cdot e_2 = g_2 \cdot e_1$. Indeed, since they satisfy the equation $g(a_i) = y_i \cdot e(a_i)$ for each $i \in [n]$, we have

$$(g_1 \cdot e_2)(a_i) = y_i \cdot e_1(a_i) e_2(a_i) = (e_1 \cdot g_2)(a_i)$$

for each $i \in [n]$. Thus, the polynomial $g_1 \cdot e_2 - g_2 \cdot e_1$ is 0 on at least n points. However, its degree is bounded by $(k-1+t)+t = k+2t-1$, which is at most $n-1$ for $t \leq \lfloor \frac{n-k}{2} \rfloor$. Thus, the polynomial $g_2 \cdot e_1 - g_1 \cdot e_2$ must be identically zero, and we have $g_1 \cdot e_2 = g_2 \cdot e_1$ as desired. ■

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. 4