Lecture 12

Lecturer: Dan Spielman *Scribe: Sofya Raskhodnikova*

In this lecture we study Reed-Solomon and Justesen codes. Reed-Solomon codes have numerous theoretical and real life applications. They are used as building blocks of concatenated codes, an example of which is provided by Justesen codes. Justesen codes are the first constructed family of **asymptotically good** binary error-correcting codes (i.e. their rate and relative minimum distance remain positive as the length increases).

1 Reed-Solomon Codes

1.1 Construction

Reed-Solomon Codes are not binary. We extend the definition of codes to larger alphabets, allowing message symbols to be taken from a finite field $\mathbb{F} = GF(q)$ with q elements.

Definition 1 *A code is an injection $\mathcal{C}$ from $\mathbb{F}^m$ (the message space) to $\mathbb{F}^n$.*

The length of a Reed-Solomon code is $n = q-1$. We denote all distinct non-zero elements of $\mathbb{F}$ as $x_1, \ldots, x_n$.

The idea behind Reed-Solomon codes is very simple and elegant. A message $w = (w_0, w_1, \ldots, w_{m-1}) \in \mathbb{F}^m$ is associated with the polynomial $P_w(x) = w_0 + xw_1 + x^2w_2 + \cdots + x^{m-1}w_{m-1}$. To encode, we evaluate $P_w(x)$ at all non-zero elements of the field.

$$\mathcal{C}(w) = (P_w(x_1), \ldots, P_w(x_n)).$$

The polynomial is not evaluated at zero for historical reasons. If $P_w(0)$ is included, we get so-called extended Reed-Solomon code.

Reed-Solomon codes have good minimum distance properties.

Proposition 2 *The minimum distance δ of Reed-Solomon codes is $n - m + 1$.*

Proof Consider two distinct codewords P_w and P_v, $w \neq v$. We say that two polynomials differ on a symbol if they give different answers when evaluated at the corresponding element of the field $\mathbb{F}$. Otherwise, we say that the polynomials agree on that symbol. Assume by way of contradiction that P_w and P_v differ on at most $n - m$ symbols. Then they agree on at least m symbols. Consequently, $P_w - P_v$ is zero when evaluated at m elements of $\mathbb{F}$, but has degree at most $m - 1$. Hence $P_w - P_v \equiv 0$, i.e. $P_w \equiv P_v$. ■

Note that by the Singleton bound $\delta \leq n - m + 1$ for any linear code. Thus the minimum distance of the Reed-Solomon code is optimal!

1.2 Encoding

Reed-Solomon code is linear because $\mathbb{F}[x]$ is a vector space. As with all linear codes, we could encode by multiplying a message word by a generator matrix. However, we can do encoding faster than that.

Let α be a generator for $\mathbb{F}$. Then all non-zero elements of $\mathbb{F}$ can be represented as α^k, where $k = 1, \ldots, n-1$. Consider evaluating P_w at $1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{n-1}$. It can be done by multiplying the following generator matrix G by the vector of the coefficients of $P_w(x)$.

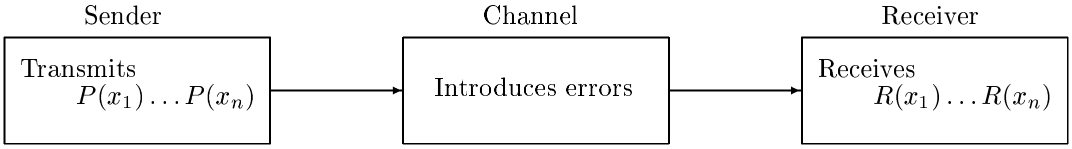
$$\begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \ldots & \alpha^{2(n-1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \alpha^{m-1} & \alpha^{2(m-1)} & \alpha^{3(m-1)} & \ldots & \alpha^{(m-1)(n-1)} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_{n-1} \end{pmatrix} = \begin{pmatrix} P_w(1) \\ P_w(\alpha) \\ P_w(\alpha^2) \\ \ldots \\ P_w(\alpha^{n-1}) \end{pmatrix}$$

G is a Fourier Transform matrix. Typically, we see it over complex numbers, where $\alpha = e^{\frac{2\pi i}{n}}$. Here the matrix is over a finite field $\mathbb{F}$. A paper by Preparata and Sarwate shows how to do FFT in $O(n \log n \log\log n)$.

1.3 Decoding

Since the minimum distance of the Reed-Solomon code is $n - m + 1$, we might hope to correct up to $\frac{n-m}{2}$ errors. It turns out that we can do that efficiently.

Let us view the n values that are received $(y_1, y_2, \ldots, y_n)$ as points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. There exists a polynomial $R(x)$ of degree at most $n - 1$ such that $R(x_i) = y_i$ for all i from 1 to n.



We show how to correct the errors, assuming that $e \leq \frac{n-m}{2}$ of them are introduced.

Definition 3 *The* **error-locator polynomial**, *$E(x)$, is the product of all terms $(x - x_i)$ such that an error occurred in character i,*

$$E(x) = \prod_{R(x_i) \neq p(x_i)} (x - x_i).$$

If we find the error-locator polynomial, we can determine error positions. After that we can determine $P(x)$ by interpolation using information from correct positions. Observe that

$$R(x_i)E(x_i) = p(x_i)E(x_i) \text{ for all } 1 \leq i \leq n.$$

Notice that

$$\begin{array}{lll} R(x) & \text{has degree at most} & n-1 \\ P(x) & \text{has degree at most} & m-1 \\ E(x) & \text{has degree at most} & e \end{array}$$

For simplicity of notation, in the rest of the lecture we denote the coefficients of polynomial $A(x)$ with $A_0, A_1, A_2, \ldots,$ starting from the free coefficient. A_i is assumed to be zero for negative i. For instance, $R(x) = \sum_{i=0}^{n-1} R_i x^i$ and $E(x) = \sum_{i=0}^{e} E_i x^i$. Evaluating the product of the above polynomials at a non-zero element x_i of $\mathbb{F}$ and using $x_i^n = 1$ (the equality holds because $\mathbb{F} \backslash \{0\}$ forms a group of size n under multiplication), we arrive at

$$R(x_i)E(x_i) = \sum_{l=0}^{e+n-1} x_i^l \sum_{\substack{j+k=l \\ 0 \leq j \leq e \\ 0 < k \leq n-1}} E_j R_k = \sum_{l=0}^{n-1} x_i^l \sum_{j=0}^{e} E_j R_{(l-j) \mod n}. \tag{1}$$

Let $Q(x) = \sum_{l=0}^{n-1} x^l \sum_{j=0}^{e} E_j R_{(l-j) \mod n}$ and $S(x) = P(x)E(x)$. Polynomials $Q(x)$ and $S(x)$ agree on all non-zero elements of $\mathbb{F}$. Both of them have degree at most $n - 1$. Therefore $Q(x) \equiv S(x)$. Therefore the degree of $Q(x)$ is the same as the degree of $S(x)$ which is at most $m + e - 1$. Hence, (1) implies that

$$\sum_{j=0}^{e} E_j R_{(i-j) \mod n} = 0 \text{ for } m + e \leq i \leq n - 1. \tag{2}$$

To find E, we solve the system of linear equations (2) with unknowns $E_0, E_1, \ldots, E_e$. The system has a non-zero solution, for instance the error locator polynomial.

Proposition 4 *If a non-zero polynomial $E'(x)$ satisfies the system (2) and* $Q'(x) = \sum_{l=0}^{n-1} x^l \sum_{j=0}^{e} E'_j R_{(l-j) \bmod n}$, *then* $P(x) \equiv \frac{Q'(x)}{E'(x)}$.

Proof Since $E'(x)$ is a solution to the system, the degree of $Q'(x)$ is at most $m+e-1$. Define $S'(x) = E'(x)P(x)$. By construction, $Q'(x_i) = E'(x_i)R(x_i)$. Therefore $Q'(x_i) \neq S'(x_i)$ only when $R(x_i) \neq P(x_i)$. Since $R(x)$ and $P(x)$ agree on at least $n-e$ symbols, $Q'(x)$ and $S'(x)$ also agree on at least $n-e$ symbols. $Q'(x)$ and $S'(x)$ both have degrees at most $m+e-1 < n-e$, since $e \leq \frac{n-m}{2}$. Consequently, $Q'(x) \equiv S'(x)$, i.e. $P(x) \equiv \frac{Q'(x)}{E'(x)}$. ■

Using the extended GCD algorithm, the linear system can be solved in $O(n \log^2 n \log\log n)$ time [Aho, Hopcroft and Ulman]. So the decoding scheme is practical, but the constant hidden in the asymptotic notation is quite large. Victor Pan '97 gives a $O(n \log n \log\log n)$ time decoding algorithm for the case when q is a prime. Interpolation can be done in $O(n \log n \log\log n)$ time.

Madhu Sudan and Venkatesan Guruswami '98 devised a scheme that returns the list of all polynomials within distance $n - \sqrt{mn}$ of the received word in polynomial time. The number of errors this algorithm can tolerate, $n - \sqrt{mn}$, is strictly better than $\frac{n-m}{2} = n - \frac{m+n}{2}$. To get a feeling of the difference, consider $m = \frac{n}{2}$. In this case, the distance between two distinct codewords is $n - m + 1 = \frac{n}{2} + 1$, and hence we can correct $\frac{n}{4}$ errors. Sudan-Guruswami algorithm, on the other hand, can tolerate $n(1 - \frac{1}{\sqrt{2}})$ errors.

2 Justesen codes.

In this section, we show how to concatenate a Reed-Solomon code with a binary code to obtain an asymptotically good code.

Take a Reed-Solomon code over $\mathbb{F} = GF(q)$ where $q = 2^a$. Each symbol in this code can be recorded with a bits. Consider a binary code $\mathcal{C} : \{0,1\}^a \to \{0,1\}^b$ with rate r and relative minimum distance δ. The Justesen code is formed by first applying Reed-Solomon encoding to the message, and then viewing each character of the resultant codeword as an a-bit string and encoding it with $\mathcal{C}$.

$$P_w \to \underbrace{P_w(x_1)}_{a \text{ bits}} \; \underbrace{P_w(x_2)}_{a \text{ bits}} \ldots \underbrace{P_w(x_{q-1})}_{a \text{ bits}} \to \underbrace{\mathcal{C}(P_w(x_1))}_{b \text{ bits}} \; \underbrace{\mathcal{C}(P_w(x_2))}_{b \text{ bits}} \ldots \underbrace{\mathcal{C}(P_w(x_{q-1})}_{b \text{ bits}}.$$

If $\mathcal{C}$ is a good code, then the concatenated code is also good. It has rate $(\frac{m}{n} r)$ and relative minimum distance $(\frac{n-m+1}{n}\delta)$.

To construct a good Justesen code we need a good code $\mathcal{C}$. However, since $a = \log_2 q$, $\mathcal{C}$ is much shorter and should be easier to find. One possible, but not the best working idea is to apply concatenation recursively. Another idea is to choose a "random code". It should not be very expensive when $\log_2 q$ is small.

Gilbert-Varshamov bound from the last lecture says that a random linear code of rate r has relative minimum distance δ with probability at least $1 - \frac{2^{nH(\delta)} \cdot 2^{rn}}{2^n}$.

Let $r = \frac{1}{2}$. Take $\mathcal{C}_\alpha : \{0,1\}^a \to \{0,1\}^{2a}$ and view it as mapping an element of $GF(2^a)$ to an element of $GF(2^a)^2$. Namely, $\mathcal{C}_\alpha(x) = (x, \alpha x)$ and $\alpha \in GF(2^a)$. We claim that this code is good for random α,

$$Pr[\mathcal{C}_\alpha \text{ has relative minimum distance at least } \delta] \geq 1 - \frac{2^{nH(\delta)} \cdot 2^{rn}}{2^n}.$$

In the proof of Gilbert-Varshamov bound we used only

$$Pr[w \in \mathcal{C}_\alpha] = \frac{2^{rn}}{2^n}.$$

It holds in our case, too. Justesen used $\mathcal{C}_\alpha$ in his concatenated code to get the first asymptotically good code.