Coding Theory

Lecture 6

Lecture by Dr. Elette Boyle
Typeset by Steven Karas

2017-12-05
Last edited 18:01:53 2017-12-05

Disclaimer These lecture notes are based on the lecture for the course Coding Theory, taught by Dr. Elette Boyle at IDC Herzliyah in the fall semester of 2017/2018. Sections may be based on the lecture notes written by Dr. Elette Boyle.

Agenda

- Efficient decoding of Reed Solomon codes

1 Recap - Stochastic noise and channel capacity

1.1 Shannon's memoryless channel model

Input alphabet $\mathcal{X}$, output alphabet $\mathcal{Y}$, transition matrix M.

1.2 BSCp, qSCp, BECa

Binary symmetric channel (BSC_p) has input and output alphabets of 0,1 with a crossover probability of p.

q-ary symmetric channel qSC_p is as above, but with $\mathcal{X} = \mathcal{Y}$, $|\mathcal{X}| = q$.

Binary erasure channel BEC_α is a binary channel that only creates erasures with some probability α.

1.3 Channel Capacity

The capacity of a channel is the value $C \in \mathbb{R}$ such that:

For any $R < C$, there exists some code E, D with rate R and negligible decoding error ($\lim_{n\to\infty} f(n) = 0$). For any $R > C$, then for every code E, D, the decoding error probability is bounded below by some constant.

Note, that the best possible capacity is 1, and the worst possible capacity is 0.

There is another proof that

1.4 Capacity of BSCp

The capacity of BSC_p is $1 - H(p)$.

The proof sketch is that there are many points on the $(1 \pm \gamma)pn$ shell around a codeword $E(\vec{m})$ must decode to $\vec{m}$ (so as to have sufficiently small decoding error). Therefore, we can't have too many $\vec{m}$, because we're limited by 2^n channel messages. Therefore, $C \leq 1 - H(p)$.

For the other direction ($C \geq 1 - H(p)$), start by considering a random code (E, D):

E: Construct as every $\vec{m}$, choose $E(\vec{m})$ at random. D: Max likelihood decoder (for BSC_p, this is the closest codeword by Hamming distance).

Show for a fixed $\vec{m}$, then over the probability of the choice of the code E and BSC_p error $\vec{e}$, the decoder fails for this $\vec{m}, \vec{e}$ is exponentially small: $< 2^{-\delta' n}$ for some $\delta' > 0$.

First, show that the probability of the Voronoi cells around $E(\vec{m})$ is bounded (union bound approach). This gives us that $\forall \vec{m}$, around $1 - 2^{-\delta' n}$ fraction of the E's are good for this $\vec{m}$.

Now, we need a single E that is simultaneously good for all $\vec{m}$s. The union bound is not enough: $1 - (2^k)(2^{-\delta' n}) < 0$ because $k > \delta' n$. So we show that there exists E with few bad $\vec{m}$s, and remove them. This keeps the rate asymptotically sufficient and decoding succeeds.

2 Efficient decoding of Reed Solomon codes

When first proposed, this Reed and Solomon's algorithm was basically a brute force approach, and an efficient algorithm was finally proposed 8 years later.

Recall that RS codes are $[n, k, d = (n-k+1)]_q$ codes with a list of n distinct evaluation points $(\alpha_1, \dots, \alpha_n)$. Where $\mathrm{RS} : {\mathbb{F}_q}^k \to {\mathbb{F}_q}^n$ which takes $\vec{m} = (m_0, \dots, m_{k-1})$ and gives $\mathrm{RS}(\vec{m}) = (f_{\vec{m}}(\alpha_1), \dots, f_{\vec{m}}(\alpha_1))$ where $f_{\vec{m}} = \sum_{i=0}^{k-1} m_i x^i$.

Recall that the distance gives us the best possible error correction of up to $\frac{d-1}{2}$ errors.

Let $e < \frac{n-k+1}{2}$.

2.1 Building blocks

Our input is a received word $\vec{y} = (y_1, \dots, y_n) \in {\mathbb{F}_q}^n$. Assume that $\vec{y}$ has at most e errors.

Recall that $RS(\vec{m})$ defines a polynomial, and we received a polynomial with some slightly different evaluation points. Notably, this means that there exists a polynomial $P(X)$ of degree $\leq k-1$ such that $\Delta(\vec{y}, (P(\alpha_i)_{i=0}^n)) \leq e$. Solving for $P(x)$ is equivalent to finding the message $\vec{m}$. We claim that $P(x)$ is unique.

If there were no errors, then we have n equations with k unknowns:

$$
\begin{aligned}
y_1 &= P_0 + P_1\alpha_1 + P_2\alpha_1^2 + \ldots + P_{k-1}\alpha_1^{k-1} \\
y_2 &= P_0 + P_1\alpha_2 + P_2\alpha_2^2 + \ldots + P_{k-1}\alpha_2^{k-1} \\
&\dots \\
y_n &= P_0 + P_1\alpha_n + P_2\alpha_n^2 + \ldots + P_{k-1}\alpha_n^{k-1}
\end{aligned}
$$

Which are linear equations in $P_0, \dots, P_{k-1}$, which is efficient using Gaussian elimination. However, we have errors, which means that up to e of these equations are wrong.

Suppose we magically receive a polynomial $E(x)$ such that $E(\alpha_i) = 0$ iff $y_i \neq P(\alpha_i)$. Note that such a polynomial must exist ($\prod_{i=y\neq P(\alpha_i)}(X - \alpha_i)$), and that it also satisfies $\deg E \leq e$. If we had such an E, then we could remove the erroneous problems, which allows us to solve the system and decode the message. We call such a polynomial the "error locater polynomial".

For $1 \leq i \leq n$, with no errors, $P(\alpha_i) = \alpha_i$, and with errors: $P(\alpha_i)E(\alpha_i) = y_i E(\alpha_i)$. This is true because if y_i doesn't have an error, then $P(\alpha_i) = y_i$, but if it does then $E(\alpha_i) = 0$.

We don't know $E(x)$, but we can think of its coefficients E_i as variables.

$P(\alpha_i)E(\alpha_i) = y_i E(\alpha_i)$ gives us n equations in $P_0, \dots, P(k-1) \times E_0, \dots, E_e$. But now our equations are quadratic[1] that include products $P_j E_l$.

So we use linearization to work around this:

$$
\overset{N_0}{P_0 E_0} + \overset{N_1}{P_0 E_1}\alpha + \ldots + P_0 E_e \alpha = EP
$$

The cross terms we simply refer to as a new variable, forgetting that it has this extra structure. This will allow us to solve, assuming that the number of unknowns is less than the number of constraints.

Define $N(x) = P(x)E(x)$. We have equations $\forall 1 \leq i \leq n$: $N(\alpha_i) = y_i E(\alpha_i)$ with $\deg N + 1 + \deg E + 1$ unknowns. Recall that $\deg N = \deg P + \deg E \leq (k-1) + e$. Therefore, unknowns $\leq ((k - 1 + e) + 1) + (e + 1) = k + 2e + 1 < k + n - k + 1 + 1 = n + 2$. Because we're talking about integers, unknowns $\leq n + 1$.

However, we're still missing a final constraint, which we take as $E_e = 1$, as the final $n + 1$st linear equation. This is possible because we can always find E of degree e where $E_e \neq 0$ and then we don't care about scaling all of E by a constant.

So we will be able to solve for the unknowns $N_0, \dots, N_{n+e-1}, E_0, \dots, E_e$. Once we know $N(x)$ and $E(x)$, we can finally $\frac{N(x)}{E(x)} = P(x)$.

The coefficients of $P(x)$ are the decoded message.

2.2 Berlekamp-Welch Algorithm

As input, we get $k \geq k \geq 1$ and $0 < e < \frac{n-k+1}{2}$ and n pairs $\{(\alpha_i, y_i)\}_{i=1}^n$ with distinct α_i.

The output is a description of a polynomial $P(x)$ of degree $\leq k - 1$.

$O(n^3)$ for Gaussian elimination; $O(n^3)$ for polynomial division

[1] We can't solve these generically as this is NP-Hard

First, use Gaussian elimination to solve for a nonzero polynomial $E(x)$ of degree e, and a nonzero polynomial $N(x)$ of degree $\leq e+k-1$ such that $E(x)$ is a monic polynomial ($E_e = 1$) and $\forall 1 \leq i \leq n : y_i E(\alpha_i) = N(\alpha_i)$. This provides us with $n+1$ constraints.

Next, if $E(x), N(x)$ as above do not exist, or if $E(x)$ does not divide $N(x)$, return failure.

Let $P(X) = \frac{E(x)}{N(x)}$.

If $\Delta(\vec{y}, (P(\alpha_i))_{i=0}^{n}) > e$, then return failure.

Return $P(x)$.

2.3 Correctness Theorem

If $(P(\alpha_i))_{i=0}^{n}$ is transmitted, then for a polynomial $P(x)$ of degree $\leq k-1$ and at most $e < \frac{n-k+1}{2}$ error occur, then the Berlekamp-Welch algorithm will output $P(x)$.

That is, this gives us correct decoding up to $e < \frac{n-k+1}{2}$ errors.

To prove this, we want to prove two lemmas, that such a pair $E(x), N(x)$ exists, and that for any pair the ratio is always the same.

2.3.1 Lemma 1

There exists polynomials $E^*(x), N^*(x)$ that satisfy $\deg E^*(x) = e$, $\deg N^*(x) \leq e+k-1$ and $n+1$ constraints. Additionally, these satisfy that $\frac{N^*(x)}{E^*(x)} = P(x)$.

Proof

$$E^*(x) = \prod_{i=y_i \neq P(\alpha_i)} (x - \alpha_i) \cdot x^{e - \Delta(\vec{y}, (P(\alpha_i))_{i=0}^{n})}$$

$$N^*(x) = P(x)E^*(x)$$

By construction, $\frac{N^*(x)}{E^*(x)} = P(x)$.

By construction, $\deg E^*(x) = e$.

By construction, $\deg N^*(x) \leq (k-1) + e$.

By construction, $E_e = 1$.

$$\forall 1 \leq i \leq n : y_i E^*(\alpha_i) = N^*(\alpha_i) \text{ when } E^*(\alpha_i) = 0 \Rightarrow N^*(\alpha_i) = y_i E^*(\alpha_i) = 0$$

$$E^*(\alpha_i) \neq 0 \Rightarrow y_i = P(\alpha_i) \underset{\text{by defn of } E^*}{\Rightarrow} y_i E^*(\alpha_i) = N^*(\alpha_i)$$

2.3.2 Lemma 2

For any 2 distinct solutions $(E(x), N(x))$ and $(\hat{E}(x), \hat{N}(x))$ that satisfy the first step of the algorithm, then they will also satisfy:

$$\frac{N(x)}{E(x)} = \frac{\hat{N}(x)}{\hat{E}(x)} = P(x)$$

Proof Consider the cross product polynomials: $N(x)\hat{E}(x)$ and $\hat{N}(x)E(x)$.

Note that the degree of each of these is $\leq (k+e-1) + e = k + 2e - 1$.

Denote the polynomial $R(x) = N(x)\hat{E}(x) - \hat{N}(x)E(x)$. We will show that $R(x)$ is the zero polynomial by showing that it has n zeros, and by elimination it must be the zero polynomial.

$$\deg R(x) \leq k + 2e - 1 < k + n - k + 1 - 1 = n$$

We know that $\forall 1 \leq i \leq n : y_i E(\alpha_i) = N(\alpha_i)$, $y_i \hat{E}(\alpha_i) = \hat{N}(\alpha_i)$.

$$\begin{aligned} R(\alpha_i) &= N(\alpha_i)\hat{E}(\alpha_i) - \hat{N}(\alpha_i)E(\alpha_i) \\ &= (y_i E(\alpha_i))\hat{E}(\alpha_i) - (y_i \hat{E}(\alpha_i))E(\alpha_i) \\ &= 0 \end{aligned}$$

So R has degree $< n$ polynomial with $\geq n$ zeroes.

Therefore, $R(x)$ is the zero polynomial, which implies that $N(x)\hat{E}(x) \equiv \hat{N}(x)E(x)$.

$$\frac{N(x)}{E(x)} = \frac{\hat{N}(x)}{\hat{E}(x)} = P(x)$$

2.3.3 Together

Lemma 1 gives us that we will find a solution N^*, E^* in step 1, and that it yields the correct ratio $\frac{N(x)}{E(x)} = P(x)$. Lemma 2 gives us that no matter what N, E we find in step 1, it will give us the same ratio $\frac{N(x)}{E(x)}$. In particular, it must give $\frac{N(x)}{E(x)} = P(x)$.

3 Next Week

Next lecture, we'll push further in the same direction, and discuss list decoding, which provides a list of possible codewords under larger than d errors.