**CS 229r Essential Coding Theory** Feb 21, 2017

# Lecture 9

*Instructor: Madhu Sudan* *Scribes: Jeffrey Ling*

## 1 Algorithms in coding theory

Now that we know about some codes, our goal now is to construct the code, i.e. give the encoding and decoding algorithm.

Given a message, we want to be able to compute encoding $E : \Sigma^k \to \Sigma^n$ efficiently. Conversely, given corrupted message, we want to compute decoding $D : \Sigma^n \to \Sigma^k$ efficiently. Here, efficiently is poly$(n)$ time[1].

## 2 Easy problems

From now on, we only consider linear codes. We then want to build the generator matrix $G$, which is equivalent to building $H$ via polynomial reduction. Any poly$(n)$ amount of information about $E, D$ is ok. (This is also known as building a polynomial sized circuit).

Note that we allow arbitrary preprocessing time to obtain these (random is ok too, such as in the GV bound).

### 2.1 Encoding

Poly time encoding is easy for linear codes, as the description can be written in poly number of bits. Later we ask if we can do this in time linear in $n$ or $k$.

### 2.2 Erasure decoding

The encoded message in $\Sigma^n$ is corrupted to $(\Sigma \cup ?)^n$. Given the generator matrix, if we remove columns of $G$ corresponding to corruptions, does there exist a message that turns into that decoding? This is solvable as a linear system: possible messages are a linear subspace.

When is this message unique? That is, how many erasures can we allow for? Answer: $d - 1$, where $d$ is distance of code.

### 2.3 Brute-force decoding algorithms $\Sigma^k \to \Sigma^n$

- Enumerate all messages, encode them, and measure distance. Runs in time poly$(n)|\Sigma^k|$. This is fine for a small number of code words (e.g. Hadamard, dual BCH).
- Enumerate all possible errors. There are $\binom{n}{t} \cdot |\Sigma^t|$ such if there are $t$ errors we wish to correct.

  We can also use the parity check matrix. Recall that with error $e$,

  $$(E(m) + e)H = eH$$

  since $E(m)H = 0$. We can then build a table of all possible results $eH$, i.e. syndromes.

  We can enumerate all errors using a lookup table of syndromes. This converts the big running time to space $\approx \Sigma^{n-k}$, which we can lookup quickly in $poly(n)$.

[1] We assume here $n$ is polynomial in $k$. In general we might want poly$(k)$ time.

# 3 General problem for linear codes

We define the nearest codeword problem (NCP), and also NCP + preprocessing (NCPP).

**Definition 1.** *Nearest codeword problem (NCP).*
*Input: $G \in \mathbb{F}_q^{k \times n}$ the generator matrix, $r \in \mathbb{F}_q^n$ the received message, $t$ bound on number of errors.*
*Output: message $m$ s.t. $\Delta(mG, r) \leq t$ if such $m$ exists. Else, "too many errors".*

We also include an additional "Promise" about $G$.

**Definition 2.** *Promise: $G$ generates code of distance $> 2t$.*

We are mostly interested in the general case where the Promise is satisfied and Preprocessing allowed. We don't know how to solve this, however! It may be polynomially solvable, but would be surprising to Madhu.

| | Preproc | No preproc |
|---|---|---|
| Promise | ? | ? |
| No promise | NP-hard | NP-complete |

# 4 General RS-decoding

We now consider Reed-Solomon codes. It turns out the general problem is NP-hard.

**Definition 3.** *NP-hard version of RS-decoding.*
*Input: $\mathbb{F}_q, n, k$ the code parameters, $x_1, \ldots, x_n \in \mathbb{F}_q$ distinct points that we evaluate the message polynomials on, $y_1, \ldots, y_n \in \mathbb{F}_q$ the received response, $t = n - k - 2$ the number of errors.*
*Decide: if there exists polynomial $p \in \mathbb{F}_q[x]$ with degree $< k$ s.t.*

$$|\{p(x_i) \neq y_i | i \in [n]\}| \leq t$$

Guruswami + Vardy and GGG give algorithms for $t = n - k - c$ for some constants $c$. However, they both require $q$ to be exponentially large in $n$.

If we set $t = (n-k)/2$, we get the easy version of RS-decoding for which there exists a polynomial time algorithm. Literature on this includes: Peterson '60, Berlekamp-Massey 70s (the most cited "source" for decoding BCH codes in the 80s and 90s), Welch-Berlekamp '86 and Gemmell-Sudan '92, Pellikaan '88 and Duursma-Kotter '94

NB: many of the early papers assume codes are cyclic, i.e. rotated codewords are also codewords. This is irrelevant to the decoding algorithm.

# 5 Easy RS-decoding

From now on assume $t \leq (n-k)/2$.

We use the key idea of an *error-locating polynomial*.

**Definition 4.** *Fix a received vector $y$, and suppose $p$ was the message polynomial. Then define an error-locating polynomial $E$ as one that satisfies*

$$E(x_i) = 0 \text{ if } p(x_i) \neq y_i, \quad \deg(E) \leq t$$

Note that by the definition, $E$ satisfies

$$E(x_i)p(x_i) = E(x_i)y_i$$

Let

$$W(x_i) = E(x_i)y_i \implies y_i = W(x_i)/E(x_i)$$

so that $y_i$ can be expressed as a rational function of $x_i$, with numerator degree $< k + t$ and denominator degree $\leq t$.

Given $x, y$, is there such a rational function? If so, we check that it's actually expressible as a polynomial. If not, we failed to find the correct message.

This gives us our decoding algorithm:

1. Find polynomials $E, W$ s.t. $\deg W < k + t$, $\deg E \leq t$, and $E$ not identically 0 s.t. $E(x_i)y_i = W(x_i) \forall i$.
   This can be done by solving a linear system.
2. If $W/E = p$ is a polynomial, output $p$, else output "too many errors".

Analysis:

(A) Does a solution to step 1 exist? Yes, if number of errors is small. We have $n$ equations in $< k + 2t$ unknowns.

(B) What if multiple solutions exist? I.e., algorithm finds other solution $(E', W')$ vs $(E, W)$.

**Claim 5.** *Any two distinct outputs $(E, W)$ and $(E', W')$ of the algorithm satisfies*

$$E/W = E'/W' \iff EW' = E'W$$

*Proof.* The idea is to evaluate both sides on enough points, since we know both sides are bounded degree polynomials. We claim that every $x_i$ satisfies $E(x_i)W'(x_i) = E'(x_i)W(x_i)$.

Indeed,

$$E(x_i)y_i = W(x_i), W'(x_i) = y_i E'(x_i) \implies EW'y_i = y_i E'W$$

on the $x_i$. We can cancel the $y_i$ if $y_i \neq 0$, and if not then by definition $W(x_i) = 0$ anyway. Thus the claim holds. □

Since the degree of both sides is $< 2t + k$, as long as $n = 2t + k$, we have enough points to make them agree. This is true by our assumption on $t$.

Here we solved decoding when number of errors is up to half the min distance. Later we see other algorithms that can solve more errors, but have weaker guarantees.

# 6 Generalizing the algorithm

We solved decoding for RS codes. What about other codes, e.g. Reed-Muller, BCH when we have few errors?

It turns out we can generalize the above algorithm to other algebraic codes!

How can we do this? We observe that polynomials in themselves are error-correcting codes, i.e. they can't agree on too many points. Before, we also use the property that the product of polynomials is another polynomial.

Let $\mathcal{E}$ be the subspace of error locator polynomials, which is also the space of polynomials with $\deg \leq t$. Let $\mathcal{M}$ be the space of messages, i.e. polynomials with $\deg < k$. Define

$$\mathcal{E} \star \mathcal{M} = \{(e_1 m_1, e_2 m_2, \ldots, e_n m_n) | (e_1, \ldots, e_n) \in \mathcal{E}, (m_1, \ldots, m_n) \in \mathcal{M}\}$$

as the coordinate-wise product (not necessarily linear). Let $\mathcal{W} \supseteq \mathcal{E} \star \mathcal{M}$ be the span of all polys $W$ that might arise in the product, i.e. those with $\deg < k + t$.

**Exercise 6.** *Show that the subspaces above actually are given by all polynomials with the specified degrees.*

Note that in general, the product would give dimension $kt$, but here we get dimension $k + t$. This is a miracle of polynomials!

We begin to see how to generalize our algorithm:

**Definition 7.** *Given $\mathcal{M}$, $(\mathcal{E}, \mathcal{W})$ form a $t$-error-locating pair for $\mathcal{M}$ if*

1. *$\Delta(\mathcal{W})$ is large enough,*
2. *$\dim(\mathcal{E})$ is large enough,*
3. *$\mathcal{W} \supseteq \mathcal{E} \star \mathcal{M}$*

*where we define "large enough" later on.*

That is, if we have a high distance space ($\mathcal{W}$) that contains a high dimensional subspace ($\mathcal{E}$), we get a decoding algorithm.

Algorithm: Given $y = (y_1, \ldots, y_n)$ the received message,

1. find $E \in \mathcal{E}$, $W \in \mathcal{W}$, $E \neq 0$ s.t. $E \star y = W$.
2. if $E_i \neq 0$, the decoded message is $Z_i = W_i / E_i$

Analysis sketch: same as before. We can find $E$ if dimension of $\mathcal{E}$ is high enough, i.e. $\dim(\mathcal{E}) \geq t + 1$. If $E \star y = W$ and $E' \star y = W'$, we can show they decode to same codeword.

**Exercise 8.** *Complete the analysis above. What should the distance of $W$ be? Hint: Maybe $t + 1$*

It turns out that every AG code fits this algorithm.

## 6.1 Abstraction of Algebraic-Geometry (Garcia-Stichtenoth/TVZ) codes

$\exists$ functions $f_1, \ldots, f_n \in \mathbb{F}_q^n$, considered as vectors (evaluations at $n$ places). some number $g$ of $f_i = 0$ (here $g$ is the genus, $g = n/(\sqrt{q} - 1)$ perhaps)

$C_i = span\{f_1, \ldots, f_i\}$ is a code of distance $n - i$. $C_i$ has dimension $\geq i - g$. $f_i \star f_j \in span\{f_1, \ldots, f_{i+j}\}$ satisfies the important product property.

This fits into the above abstraction.

Next lecture: now that we have one code we know how to decode efficiently, we can get up to the Shannon capacity efficiently. We will see how to apply concatenation.

# References