Coding Theory
Lecture 7

Lecture by Dr. Elette Boyle
Typeset by Steven Karas

2017-12-07
Last edited 18:25:49 2017-12-07

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
- Beyond the unique decoding barrier
- List decoding

1 Review

1.1 Berlekamp-Welch algorithm for uniquely decoding Reed-Solomon

Phase 1: Interpolation step Find nonzero polynomials $N(X)$ of degree $\leq k+e-1$ and $E(X)$ of degree e.

These polynomials should hold $E(\alpha_i)P(\alpha_i) = N(\alpha_i) = y_i E(\alpha_i)$ for $1 \leq i \leq n$.

Phase 2: Root finding step If $E(X)$ cleanly divides $N(X)$, then output $P(X) = \frac{N(X)}{E(X)}$.

2 List decoding

We've limited ourselves to decoding within the Hamming balls of a code, but we can extend this to the Voronoi cells around the encoding image. More interestingly, we can extend our decoder to output a list of possible answers.

Under certain conditions, we can even use additional context to determine the correct decoding of a message from a set of possible decodings.

We say that a code is list decodable if the list is polynomially sized with regards to n.

Combinatorial Definition Given $0 \leq p \leq 1$, $L \geq 1$, a code $C \subseteq \Sigma^n$ is (p, L)-list decodable if for every received word $\vec{y} \in \Sigma^n$,

$$|\{\vec{c} \in C \mid \Delta(\vec{c}, \vec{y}) \leq pn\}| \leq L$$

Degenerate Decoder If $L = q^k$, then we can trivially construct a decoding algorithm that simply outputs all possible codewords.

2.1 Asymptotic Behavior

Unique decoding is possible for $\left\lfloor \frac{d-1}{2} \right\rfloor \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ errors. This comes out to around $\frac{1-R}{2}$ fraction of errors.

Johnson Bound Turns out list decoding can achieve $1 - \sqrt{R}$. This has been proven as a tight bound, and that it is achievable, but it out of scope for today's lecture.

2.2 List decoding Reed-Solomon

Given an input $t \in \mathbb{N}$ and $\vec{y} = (y_1, \ldots, y_n) \in {\mathbb{F}_q}^n$, this algorithm will output a list of all polynomials $P(X) \in \mathbb{F}_X$ of degree $\leq k-1$ such that $P(\alpha_i) = y_i$ for at least t values of i.

Denote the error parameter as $e = n - t$.

Phase 1 Another way to think of this is finding a bivariate polynomial $Q(X, Y)$ such that $\forall 1 \leq i \leq n : Q(\alpha_i, y_i) = 0$ with a restriction on the degree of Q (to be shown later).

This is equivalent to $Q(X, Y) = YE(X) - N(X)$

Phase 2 If $Y - P(X)$ divides $Q(X, Y)$, then $P(X)$ should appear on our list[1].

Phase 1 Correctness Solve for the coefficients of $Q(X, Y)$ given the n linear constraints $Q(\alpha_i, y_i) = 0$. This will have a solution so long as the number of coefficients of Q is at least the number of constraints $= n$.

Phase 2 Correctness To make sure every desired $P(X)$ satisfies $(Y - P(X))$ divides $Q(X, Y)$, we will put some degree restrictions on Q.

Define for a polynomial $Q(X, Y) \in \mathbb{F}_q[X, Y]$, $\deg_X(Q)$ is the maximum degree of X in Q (where we view Y as a coefficient). Similarly, $\deg_Y(Q)$ is the maximum degree of Y in Q.

Suppose $Q(X, Y)$ is some polynomial with $\deg_X(Q) = a$ and $\deg_Y(Q) = b$.

e.g. $\deg_X(X^3Y^2 + XY^4) = 3$. $\deg_Y(X^3Y^2 + XY^4) = 4$

$$Q(X, Y) = \sum_{\substack{0 \leq i \leq a \\ 0 \leq j \leq b}} Q_i j X^i Y^j$$

Which gives us that Q has $(a+1)(b+1)$ coefficients.

2.2.1 Algorithm 1

Given $n \geq k \geq 1$, $\ell \geq 1$, $e = n - t$ and n pairs of $\{(\alpha_i, y_i)\}_{i=1}^n$ where $(y_1, \ldots, y_n) = \vec{y}$. Note that we will define ℓ later.

Our output is a possibly empty list of polynomials $P(X)$ of degree $\leq k - 1$.

Step 1 Find a nonzero $Q(X, Y)$ with $\deg_X(Q) \leq \ell$ and $\deg_Y(Q) \leq \frac{n}{\ell}$.

Such that $Q(\alpha_i, y_i) = 0$ for all $1 \leq i \leq n$. Construct this Q by solving the linear system.

Step 2 Factor $Q(X, Y)$ under $L \leftarrow \varnothing$ For every factor of $Q(X, Y)$ of the form $Y - P(X)$ for $\deg(P) \leq k - 1$ and that $\Delta(\vec{y}, \{P(\alpha_i)\}_{i=0}^n) \leq e$, then add P to L.

Analysis of Algorithm 1 Algorithm 1 can implemented in polynomial time, and it can list decode Reed Solomon codes of rate R up to $1 - 2\sqrt{R}$ fraction of errors.

Complexity Proof of Algorithm 1 Step 1 is solving a system of linear equations, which is $O(n^3)$, and Step 2 by efficient bivariate polynomial factorization[2].

Correctness Proof of Algorithm 1 For step 1 to be correct, we need to successfully find a $Q(X, Y)$ satisfying the requirements, we need more variables than constraints. Each coefficient is a variable, which holds because $(\ell + 1)(\frac{n}{\ell} + 1) > \ell \frac{n}{\ell} \geq n$.

For step 2, suppose that $P(X)$ is a codeword with Hamming distance e of Y. This means that $\deg(P(X)) \leq k - 1$ and $\Delta(\vec{y}, (P(\alpha_i))_{i=1}^n) \leq e$. Therefore there at least t values α_i such that $P(\alpha_i) = y_i$.

We need to show that such a $P(X)$ will appear on the list L output by algorithm 1. That is, that $P(X)$ divides whatever $Q(X)$ we found in step 1.

Recall that if $Q(X, Y) = (Y - P(X))\hat{Q}(X, Y)$, then $Q(X, P(X)) = (P(X) - P(X))\hat{Q}(X, Y)$ In other words, it holds that $(Y - P(X))$ divides $Q(X, Y)$ iff $Q(X, P(X)) = 0$. $=0$

In order to show that $Q(X, P(X)) = 0$, we will show that it has more roots than its degree.

[1] There are efficient algorithms for factoring bivariate polynomials, such algorithms are presented in the textbook

[2] Note: the number of factors is polynomial by definition

$$R(X) \stackrel{\text{defn}}{=} Q(X, P(X))$$

$$\deg(R) \leq \deg_X(Q) + \deg_Y(Q) \cdot \deg(P(X)) \leq \ell + \frac{n}{\ell} \cdot (k-1)$$

Recall that $\forall 1 \leq i \leq n : Q(\alpha_i, y_i) = 0$. Also recall that $P(\alpha_i) = y_i$ for at least t values. That means that for at least t values of $alpha_i$:

$$R(\alpha_i) = Q(\alpha_i, P(\alpha_i)) = Q(\alpha_i, y_i) = 0$$

This means that $R \equiv 0$ if $\deg(R) \leq \ell + \frac{n}{\ell} \cdot (k-1) < t \leq$ roots in R.

This also shows us how to maximize e. For any choice of $\ell > 1$, we want to minimize the bound on t, which means we choose $\ell = \sqrt{n(k-1)}$, which minimizes $\ell + \frac{n}{\ell} \cdot (k-1)$. This gives us $t > 2\sqrt{n(k-1)}$, which is equivalent to $e < n - 2\sqrt{n(k-1)}$.

Which gives us fraction error correction $< 1 - 2\sqrt{\frac{k-1}{n}} \approx 1 - 2\sqrt{R}$

Improving We want $\deg(R) = \deg(Q(X, P(X)))$ to be as small as possible, since this dictates the smallest agreement t that we can handle and still have $P(X)$s appear in step 1.

But we can't reduce the degree of Q too much, or we can't find a Q at all, because we won't have sufficient coefficients to solve the linear system.

In Algorithm 1, we bounded $\deg_X(Q)$ and $\deg_Y(Q)$ together, and took the worst case for $\deg(R)$.

Definition The $(1, w)$ weighted degree of a monomial $x^i y^j$ is equal to $i + wj$.

E.g. suppose $\deg(P) = 3$,$\deg_X(Q) \leq 2$, $\deg_Y(Q) \leq 2$: $x^0y^0 + \ldots + x^2y^2$ This is basically saying that the y's cost more than the x's by some weight factor w.

2.2.2 Algorithm 2

Instead of ℓ, have some parameter D.

Step 1 Find nonzero $Q(X, Y)$ such that $\forall 1 \leq i \leq n : Q(\alpha_i, y_i) = 0$ and $(1, (k-1))$ weighted degree of $Q(X, Y)$ is $\leq D$.

Step 2 Identical to Algorithm 1.

Correctness Proof Algorithm 2 can be efficient implemented and can list decode Reed-Solomon codes of rate R up to $1 - \sqrt{2R}$ fractional error.

From our proof of Algorithm 1, we have correctness so long as $Q(X, Y)$ has more than n coefficients, and $\deg(R) = \deg(X, P(X)) < t$.

So the only thing we need to do is to compute how many coefficients $Q(X, Y)$ can have as a function of D while satisfying $(1, k-1)$-degree of Q is $\leq D$.

$$\begin{aligned} &= |\{(i,j) \mid i + (k-1)j \leq D\}| \\ &= \ldots \\ &\geq \frac{D}{2(k-1)}(D+2) \end{aligned}$$

Note that $\deg(Q(X, P(X)))$ is effectively the $(1, k-1)$-weighted degree of Q.

So step 1 succeeds as:

$$\frac{D(D+2)}{2(k-1)} > n \Rightarrow D \geq \left\lceil \sqrt{2(k-1)n} \right\rceil$$

In order for step 2 to succeed:

$$\begin{aligned} t &> \deg(R) \\ &> (1, k-1)\text{-degree of } Q \\ &\leq D \end{aligned}$$

$$t > D > \sqrt{2(k-1)n}$$

Which gives us fractional error correction of $1 - \sqrt{2R}$.