27. List decodable codes

27.1 Introduction

In today's lecture we study list decodability of codes, where we try to correct $> \delta/2$ errors.

27.2 Going beyond unique decoding

Let us recall what Shannon's theorem tells about transmitting over a noisy channel: In the case of a binary symmetric channel (say), Shannon's theorem states that if the probability of flipping a bit is $p < 1/2$, and if $k/n = R < 1 - H(p)$, then we have encoding and decoding functions, $\text{Enc} : \{0,1\}^{Rn} \to \{0,1\}^n$ and $\text{Dec} : \{0,1\}^n \to \{0,1\}^{Rn}$ respectively, such that

$$\Pr_{\substack{m \in_r \mathcal{U}_{Rn} \\ \eta \in \text{BSC}_p}} [\text{Dec}(\text{Enc}(m) + \eta) \neq m] \leq e^{-n}.$$

On the other hand, Hamming's theory looks at worst-case errors and unique decoding guarantees the following: For every $R = k/n$, there exists a $\tau = \tau(R)$, such that there exists $\text{Enc} : \{0,1\}^{Rn} \to \{0,1\}^n$ and $\text{Dec} : \{0,1\}^n \to \{0,1\}^{Rn}$ such that for all $m \in \{0,1\}^{Rn}$ and all $\eta \in B(0, \tau n)$, $\text{Dec}(\text{Enc}(m) + \eta) = m$. Singleton bound says that $\tau < \frac{1-R}{2}$. If the size of the alphabet is larger than 2, it can be shown that reliable communication, in the sense of Shannon's theory, for relative distance $\delta = 1 - R$.

The reason why this is not possible with the concept of unique decoding is that if the relative distance of the code is δ, then if we allow for error $\geq \delta/2$, there could be more than one codeword closest to the received message. While Shannon's approach says that the decoding works most of the time, in the setting of Hamming we want to get the right message everytime. Now we see how to avoid this conflict and move to correcting errors above the $\delta/2$ bound.

27.2.1 List decoding

We now start with the definition of a list decodable code.

Definition 27.1. *For $0 < \rho < 1$, and $L > 0$, a code $\mathcal{C} \subseteq \Sigma^n$ is* (ρ, L)-list decodable *if for every $\mathbf{y} \in \Sigma^n$, $|\{\mathbf{x} \in \mathcal{C} \mid \Delta(\mathbf{y}, \mathbf{x}) \leq \rho n\}| \leq L$.*

First observe that every code $\mathcal{C}$ is $(\rho, |\Sigma|^k)$-list decodable. What we are interested in is an efficient way to get the list of codewords that are near. For this, naturally, we want the list to small so that we can list it down efficiently.

Theorem 27.2 (Johnson bound)**.** *Let $\mathcal{C}$ be an $[n,k,d]_q$ code. If $\rho < J_q(\delta)$, then $\mathcal{C}$ is (ρ, qdn)-list decodable where $J_q(\delta) = \left(1-\frac{1}{q}\right)\left(1-\sqrt{1-\frac{q\delta}{q-1}}\right)$.*

It can be shown that for any $q \geq 2$, $J_q(\delta) \geq 1-\sqrt{1-\delta}$. This gives the following corollary of the Johnson bound.

Corollary 27.3. *An $[n,k,d]_q$ code $\mathcal{C}$ is (ρ, qnd)-list decodable if $\rho < 1-\sqrt{1-\delta}$.*

The Singleton bound tells us that for Reed Solomon codes, the relative distance $\delta = 1-R$. So the natural question that arises here is whether we can list decode up to $1-\sqrt{R}$ errors for the Reed Solomon code. The Johnson bound tells us that there are only polynomially many codewords within this radius, but can we find the set efficiently.

27.3 List decoding Reed Solomon codes

Now we will see the list decoding algorithm for Reed Solomon codes. We will first see an algorithm that can correct up to $1-2\sqrt{R}$ errors, and see how to generalize that idea to get an algorithm that can correct up to $1-\sqrt{2R}$ errors. This result is due to Sudan. Guruswamy and Sudan later gave an algorithm that list decodes Reed Solomon codes up to $1-\sqrt{R}$ error.

Let's first recall what we did in the case of the unique decoding problem, now in a slightly different way. Let $(\beta_1, \beta_2, \ldots, \beta_n)$ be received message. Let $Q(x,y) = N(x) - yE(x)$ be a bivariate polynomial over $\mathbb{F}_q$. We obtained polynomials $N(x)$ and $E(x)$ such that $N(\alpha_i) - \beta_i E(\alpha_i) = 0$ for all $i \in [n]$. Then we output $P(x)$ if $y - P(x)$ is a factor of $Q(x,y)$. The final step, stated this way, requires bivariate polynomial factorization. There are polynomial-time algorithms known for this and we will not be proving that during the course.

Before we begin to apply this in list decoding RS codes, let us define the problem formally.

- **Input**: An $[n,k,d]_q$ RS code and set of tuples $\{(\alpha_1,\beta_1),(\alpha_2,\beta_2),\ldots,(\alpha_n,\beta_n)\}$.
- **Output**: The set of polynomials $P(x)$ of degree at most $k-1$ such that $P(\alpha_i) = \beta_i$ for at least t points.

To proceed as in the Welch-Berlekamp decoder, we need to do two things.

1. Put constraints on $Q(x,y)$ so that we find a solution for the linear equations that define its coefficients. If the number of coefficients is greater than n, then there is always a solution. The linear equations are obtained from the condition that $Q(\alpha_i, \beta_i) = 0$.
2. More importantly, we need $Q(x,y)$ to be divisible by $y - P(x)$ whenever $P(x)$ is a codeword that has to be output. In the Welch-Berlekamp decoder $Q(x,y)$ was of the form $N(x) - yE(x)$.

Let us try to see what constraints we want on $Q(x,y)$ so that both the above steps can be performed. Now we need to make sure that $y - P(x)$ divides $Q(x,y)$ for each of the

polynomials that are close. To that end, let $R(x) = Q(x, P(x))$. Now, if for at least t values of α_i, $P(\alpha_i) = \beta_i$, then we know that $R(x)$ has at least t roots. If the degree of $R(x)$ is at most t, then we can conclude that $R(x) \equiv 0$, and hence $y - P(x)$ divides $Q(x, y)$. Let the degree of x in $Q(x, y)$ be d_x and let the degree of y in $Q(x, y)$ be d_y. Then the degree of $R(x) \leq d_x + (k-1)d_y$ and we want this to be less than t. Also, notice that if d_x and d_y are degrees of x, y in $Q(x, y)$, then the number of unknowns in the equations that define $Q(x, y)$ is $(d_x + 1)(d_y + 1)$. If $(d_x + 1)(d_y + 1) > n$, then we know that there is a solution for the set of equations. If we set $d_x = \ell$ and $d_y = n/\ell$, for a value of ℓ to be fixed later, then $(d_x + 1)(d_y + 1) > n$. Now we want $\ell + \frac{n(k-1)}{\ell} < t$.

Algorithm 1: List decoding Reed Solomon codes

Input: An $[n, k, d]_q$ RS code and a set of tuples $\{(\alpha_i, \beta_i)\}_{1 \leq i \leq n}$, parameters ℓ, t

Output: All polynomials $P(x)$ of degree at most $k-1$ such that $P(\alpha_i) = \beta_i$ for at least t points.

1 Find a polynomial $Q(x, y)$ with $d_x \leq \ell$ and $d_y \leq n/\ell$ such that

$$\forall i \in [n], Q(\alpha_i, \beta_i) = 0.$$

2 $L \leftarrow \emptyset$.
3 **for** *each factor* $y - P(x)$ *of* $Q(x, y)$ **do**
4 **if** $P(x)$ *has degree* $\leq k-1$ *and* $P(\alpha_i) = \beta_i$ *for* $\geq t$ *points* **then**
5 $L \leftarrow L \cup P(x)$
6 **end**
7 **end**
8 Return L.

We run this algorithm with parameters $\ell = \sqrt{n(k-1)}$ and $t = 2\sqrt{nk}$. The correctness of the algorithm follows from the two lemmas below.

Lemma 27.4. *The algorithm always returns a non-zero polynomial at Step* 1*.*

Proof. Notice that in Step 1 we are solving n equations over $(\ell + 1)(\frac{n}{\ell} + 1)$ variables. So we have a homogenous system of linear equations with more number of variables than equations. So there is a non-trivial solution and hence, a non-zero $Q(x, y)$. □

Lemma 27.5. *Let $P(x)$ be a polynomial of degree at most $k-1$ such that $P(\alpha_i) = \beta_i$ for at least $t = 2\sqrt{nk}$ points. Then $y - P(x)$ divides the polynomial $Q(x, y)$.*

Proof. The polynomial $y - P(x)$ divides $Q(x, y)$, iff $Q(x, P(x)) = 0$. Let $R(x) = Q(x, P(x))$. First notice that for every α_i such that $P(\alpha_i) = \beta_i$, $R(\alpha_i) = Q(\alpha_i, \beta_i) = 0$. Therefore, $R(x)$ has $\geq t = 2\sqrt{kn}$ roots. The degree of $R(x)$ is at most $\ell + \frac{(k-1)n}{\ell} = 2\sqrt{n(k-1)} < 2\sqrt{nk}$. Therefore, $R(x)$ is the zero polynomial. □

If $t = \sqrt{nk}$, then we can correct errors up to distance $n - 2\sqrt{nk}$ up to relative distance $1 - 2\sqrt{R}$. This gives a polynomial time algorithm to list decode Reed Solomon codes up to $1 - 2\sqrt{R}$ errors.