Introduction to Coding Theory **CMU: Spring 2010**

Notes 9: Expander codes, beginning of list decoding

Mar 31, 2010

Lecturer: Venkatesan Guruswami *Scribe: Yuan Zhou*

1 Distance amplification of codes (continued from last lecture)

1.1 Code construction

Recall some definitions and lemmas introduced from last lecture.

Definition 1 (distance amplified code $G(C)$) *Let $G = (L, R, E)$ be a bipartite graph with $L = [n], R = [m]$, which is D-left-regular and d-right-regular. Let C be a binary linear code of block length $n = |L|$. For $c \in \{0,1\}^n$, define $G(c) \in (\{0,1\}^d)^m$ by*

$$G(c)_j = (c_{\Gamma_1(j)}, c_{\Gamma_2(j)}, \cdots, c_{\Gamma_d(j)}),$$

for $j \in [m]$, where $\Gamma_i(j) \in L$ denotes the i-th neighbor of $j \in R$. Now define the code $G(C)$ as

$$G(C) = \{G(c) | c \in C\}.$$

Since each bit of a codeword $c \in C$ is repeated D times in the associated codeword $G(c) \in G(C)$, we have

Lemma 2 $R(G(C)) = \dfrac{1}{D} \cdot R(C)$.

To make the (relative) distance of $G(C)$ larger than that of C, we would like to use a special class of graphs to be G, which is defined as follows.

Definition 3 (dispersers) *A bipartite graph $G = (L, R, E)$ is said to be a (γ, β)-disperser if for all subset $S \subseteq L$ with $|S| \geq \gamma n$, we have $|\Gamma(S)| \geq \beta m$.*

The following lemma follows from the definition of dispersers.

Lemma 4 *If G is a (γ, β)-disperser and $\Delta(C) \geq \gamma n$, then $\Delta(G(C)) \geq \beta m$.*

Thus, if we can get a "good" disperser (say, with large β and small γ), then we can use it to amplify the distance of a code. The follow lemma says that such a "good" disperser exists.

Lemma 5 *There exists an explicit (poly-time constructible) $(\gamma, 1-\epsilon)$-disperser with $D = d = \Theta(1/(\gamma\epsilon))$.*

Proof: Let $G = (L, R, E)$ be the double cover of Ramanujan expander (an $(n, d, \lambda \le 2\sqrt{d})$-expander). I.e. take a Ramanujan graph $H = (V, E_H)$, let L and R be two independent copies of V, and for each edge $(u, v) \in E_H$, make two edges $(u_l, v_r), (v_l, u_r)$ for E.

Now we only need to prove that for each $S \subseteq L$ with $|S| = \gamma n$, we have $\Gamma(S) \ge (1-\epsilon)n$. Fix S, let $T = R \setminus \Gamma(S)$, it suffices to prove that $|T| \le \epsilon n$. By Expander Mixing Lemma, we have

$$\begin{aligned}
& 0 = |\mathrm{E}(S,T)| \ge \frac{d|S||T|}{n} - \lambda\sqrt{|S||T|} \\
\Rightarrow\ & d^2|S||T| \le \lambda^2 n^2 \\
\Rightarrow\ & d^2|T| \le \frac{\lambda^2 n}{\gamma} && (|S| = \gamma n) \\
\Rightarrow\ & |T| \le \frac{(\lambda/d)^2}{\gamma} n \\
\Rightarrow\ & |T| \le \frac{4}{\gamma d} n && (\lambda \le 2\sqrt{d})
\end{aligned}$$

To make $|T| \le \epsilon n$, it suffices that $d \ge 4/(\gamma\epsilon)$. □

Use Tanner code (or any explicit code with positive relative rate and relative distance) as C, plug Lemma 5 into Lemma 2 and Lemma 4, we get the following codes, whose rate-distance tradeoff matches the Singleton bound, up to a constant factor.

Corollary 6 *There are explicit codes of relative distance $(1-\epsilon)$ and relative rate $\Omega(\epsilon)$, over alphabet size $2^{O(1/\epsilon)}$.*

Remark 7 *We compare the codes in Corollary 6 with algebraic geometry codes (with the same relative distance, and the same relative rate up to a constant factor), the latter one has alphabet size $O(1/\epsilon^2)$ (where the current lower bound is $\Omega(1/\epsilon)$). But the combinatorial construction of distance amplification codes is much simpler.*

If we use the code in Corollary 6 as outer code, and concatenate it with constant-size binary codes with relative rate $\Omega(\epsilon^2)$ and relative distance $(1/2 - \epsilon)$ (i.e. random code that matches Gilbert-Varshamov bound), we get a binary code with large distance.

Corollary 8 *There are explicit binary codes of relative distance $(1/2\epsilon)$ and relative rate $\Omega(\epsilon^3)$, which matches the Zyablov bound, up to a constant factor.*

1.2 Decoding algorithm for $G(C)$

Let $G = (L, R, E)$ be a double cover of Ramanujan $(n, d, \lambda \le 2\sqrt{d})$-expander constructed by Lemma 5. (Indeed we need the fact that G is a double cover of Ramanujan expanders, rather than its disperser properties). Let $C \in \{0,1\}^n$ be a binary code that can be decoded against $\gamma_0 n$ errors in linear time (say, Tanner codes).

We address our decoding problem for $G(C) \subseteq \Sigma^n$ (where $\Sigma = \{0,1\}^d$) as follows, for each $y \in \Sigma^n$ such that there exists some $c \in C$ with $\Delta(G(c), y) < (1-\epsilon)n/2$, the decoding algorithm should output $G(c)$. Note that since the distance of $G(C)$ is $(1-\epsilon)n$, if such $c \in C$ exists, it is unique.

We describe the algorithm as follows, and then explain the idea and give the proof of it.

Decoding algorithm for $G(C)$

(1) Compute $c' \in \{0,1\}^n$: for each $i \in L$, set c'_i as the majority vote for the value of i-th bit from its d-neighbors.

(2) Decode c from c' (by the decoder of C).

(3) Output $G(c)$.

The idea of the decoding algorithm is as follows. Note that $\Delta(G(c), y) < (1-\epsilon)n/2$ means that more than a half of coordinates of y are correct (i.e. agree with $G(C)$). Fix a vertex $i \in L$, think of its d neighbors $\Gamma(i)$ are randomly sampled from R (expanders simulate this in a deterministic way). Then, with high probability, more than a half of $y_{\Gamma(i)}$ will be correct. Taking the majority votes (namely c'_i) of the values of c_i recorded by $y_{\Gamma(i)}$, we will be able to recover the real c_i with high probability. Thus $\Delta(c', c)$ is small (hopefully smaller than $\gamma_0 n$), and we are able to decode c from c' by the decoder of C.

Lemma 9 *There are codes of relative rate $\Omega(\epsilon^2)$ such that we can correct a fraction $(1-\epsilon)/2$ of errors in linear time.*

Proof: Use $G(C)$ and the algorithm described above.

We claim that after the first step in the algorithm, $\Delta(c', c) \le \gamma_0 n$. To see this, let $S = \{i \in L : c'_i \neq c_i\}$, i.e., S be the set of coordinates in which the majority voting get wrong. We only need to prove that $|S| \le \gamma_0 n$. The defintion of S implies that for each $i \in S$, a majority of its neighbors are corrupted, i.e., if we let $T \subseteq R$ be the locations in which y is not corrupted ($|T| \ge (1+\epsilon)n/2$) , then

$$\forall i \in S, |\Gamma(i) \cap T| \le \frac{d}{2}.$$

Thus, by Expander Mixing Lemma, we have

$$\begin{aligned}
& \frac{d|S|}{2} \ge |\mathrm{E}(S,T)| \ge \frac{d|S||T|}{n} - \lambda\sqrt{|S||T|} \\
\Rightarrow\ & \frac{d|S|}{2} \ge \frac{d|S|(1+\epsilon)}{2} - \lambda\sqrt{|S||T|} && (|T| \ge (1+\epsilon)n/2) \\
\Rightarrow\ & \frac{\epsilon d}{2}|S| \le \lambda\sqrt{|S||T|} \\
\Rightarrow\ & \frac{\epsilon^2 d^2}{4}|S| \le \lambda^2 |T| \le \lambda^2 n \\
\Rightarrow\ & |S| \le \frac{4(\lambda/d)^2}{\epsilon^2} n \\
\Rightarrow\ & |S| \le \frac{16}{\epsilon^2 d} n && (\lambda \le 2\sqrt{d})
\end{aligned}$$

Thus, to make $|S| \le \gamma_0 n$, it suffices that $d \ge 16/(\gamma_0 \epsilon^2)$.

If we start with a graph G with $d \geq 16/(\gamma_0\epsilon^2)$, in the second step of the algorithm, the decoder finds the correct c, and then outputs the correct $G(c)$. $\square$

Exercise 1

(1) Extend the algorithm described above to correct τ errors and s erasures when $2\tau+s \leq (1-\epsilon)n$.

(2) Use GMD decoding to give binary codes of rate $\Omega(\epsilon^3)$, which are linear-time decodable up to a fraction $(1-\epsilon)/4$ of errors.

1.3 Near-optimal linear time codes

We now discuss how the above scheme can be used to achieve a near-optimal trade-off between rate and relative distance, together with linear time encoding/decoding algorithms.

Theorem 10 *[GI02] For every r, $0 < r < 1$, and all $\epsilon > 0$, there is an explicit family of codes of rate r and relative distance at least $(1-r-\epsilon)$ such that codes in the family can be encoded as well as unique decoded from a fraction $(1-r-\epsilon)/2$ of errors in time linear in the block length.*

Proof Sketch: The construction of the code looks like a combination of concatenation of Tanner codes and Reed-Solomon-based expander codes, but here the expander is used solely for simulation of a random process, rather than for distance amplification. We start with Tanner codes, which has relative rate $1/(1+\gamma)$ and can be decoded against $O(\gamma^3)$ errors in linear time for some small γ (indeed $O(\gamma^2/\log(1/\gamma))$, but $O(\gamma^3)$ is enough for the proof).

Construction of the code. Let $\gamma = \epsilon/4$. Let $G = (L, R, E)$ be a double cover of Ramanujan expanders (as constructed by Lemma 5), with degree $d = \Theta(1/(\beta\epsilon^2))$. Given a message, we encode it by Tanner codes to be C, which is guaranteed to have relative rate $1/(1+\gamma)$ and be able to decoded against a fraction $\beta > O(\gamma^3)$ of errors in linear time. Then we break C into $dr(1+\gamma)$-sized (constant-sized) blocks (say, $n = |L| = |R|$ blocks in total). Let the i-th block be $C^{(i)}$. Then encode each block $C^{(i)}$ with constant-sized Reed-Solomon codes (of relative rate $r(1+\gamma)$ and relative distance $1-r(1+\gamma)$) into $\widetilde{C^{(i)}}$. We see that the block length of each $\widetilde{C^{(i)}}$ is exactly d. We associate each $\widetilde{C^{(i)}}$ with i-th node in L. Then let each node in $j \in R$ collect the tuple

$$\mathcal{C}_j = (\widetilde{C^{(\Gamma_1(j))}}_{\Gamma^{-1}(\Gamma_1(j),j)}, \widetilde{C^{(\Gamma_2(j))}}_{\Gamma^{-1}(\Gamma_2(j),j)}, \cdots, \widetilde{C^{(\Gamma_d(j))}}_{\Gamma^{-1}(\Gamma_d(j),j)}),$$

where $\Gamma_k(j)$ is the k-th neighbor of j, and $\Gamma^{-1}(i,j)$ is the index of j among all the neighbors of i. In English, each node i on the left distributes $\widetilde{C^{(i)}}_k$ to its k-th neighbor. All the nodes on the right collect the distributed letters, and write down from whom each letter comes from.

Finally, let $\mathcal{C} = \mathcal{C}_1\mathcal{C}_2\cdots\mathcal{C}_n$ be the encoded codeword.

Rate. It is easy to see that the rate of the code is the product of the Tanner code and Reed-Solomon code, which is $1/(1+\gamma)\cdot r(1+\gamma) = r$.

Decoding algorithm. Given a (corrupted) codeword $\mathcal{C}'$, the decoding algorithm put each $\mathcal{C}'_j$ on the j-th vertex on the right side of G. Then, similar to the encoding algorithm, the right nodes distribute all the letters to the left side, and the left vertices collect the letters, getting $\widetilde{C^{(i)}}'$ for each vertex $i \in L$. Then, we run the Reed-Solomon unique decoder decode each $\widetilde{C^{(i)}}'$, getting $C^{(i)'}$. Put all the $C^{(i)'}$ together, we get C'. Run the linear-time decoder for Tanner codes to get the original message.

It is easy to see that the algorithm runs in linear time (in terms of the block length of the codeword). Now we prove that the decoding algorithm output the correct original message when at most $(1-r-\epsilon)/2$ errors occur. Indeed, we only need to prove that at most a fraction β of $\widetilde{C^{(i)}}'$ contains more than fraction $(1-r-\epsilon/4)/2 < (1-r(1+\gamma))/2$ of errors (which is the decoding capacity of Reed-Solomon unique decoder).

Let $T \in R$ be the set of corrupted entries in codeword $\mathcal{C}'$, where $|T| \leq (1-r-\epsilon)n/2$. Let $S \in L$ be the set of vertices on the left who receive more than fraction $(1-r-\epsilon/4)/2$ of corrupted letters (surely from T). Our objective is to prove that $|S| \leq \beta n$. By definition of S, we have

$$|E(S,T)| > \frac{d|S|(1-r-\epsilon/4)}{2}.$$

On the other hand, by Expander Mixing Lemma, we have

$$|E(S,T)| \leq \frac{d|S||T|}{n} + \lambda\sqrt{|S||T|}.$$

Putting two inequalities together, we have

$$\begin{aligned}
& \frac{d|S|(1-r-\epsilon/4)}{2} < \frac{d|S||T|}{n} + \lambda\sqrt{|S||T|} \\
\Rightarrow\quad & \frac{d(1-r-\epsilon/4)}{2} < \frac{d|T|}{n} + \lambda\sqrt{\frac{|T|}{|S|}} \\
\Rightarrow\quad & \frac{d(1-r-\epsilon/4)}{2} < \frac{d(1-r-\epsilon)}{2} + 2\sqrt{\frac{d|T|}{|S|}} && (|T| \leq (1-r-\epsilon)n/2 \text{ and } \lambda \leq 2\sqrt{d}) \\
\Rightarrow\quad & \sqrt{\frac{|T|}{|S|}} > \frac{3\epsilon\sqrt{d}}{16} < \frac{\epsilon\sqrt{d}}{8} \\
\Rightarrow\quad & |S| < \frac{64|T|}{\epsilon^2 d} \leq \frac{64}{\epsilon^2 d} n \leq \beta n. && (|T| \leq n \text{ and } d \geq \frac{64}{\epsilon^2\beta})
\end{aligned}$$

This finishes the proof of the correctness of the algorithm. □

Comments: The near-optimal trade-off (rate r for distance close to $(1-r)$) that almost matches the optimal Singleton bound comes from using the Reed-Solomon codes. The overall scheme can be viewed as using several independent constant-sized RS codes; the role of the expander is then to "spread out" the errors among the different copies of the RS code, so that most of these copies can be decoded, and the remaining small number of errors can be corrected by the left code C. Since only a small fraction of errors needs to be corrected by C it can have rate close to 1 and there is not much overhead in rate on top of the Reed-Solomon code.

The above codes are defined over a large alphabet (whose size depends exponentially on $1/\epsilon$). But they can be concatenated with constant-sized binary codes that lie on the Gilbert-Varshamov bound to give constructions of binary codes which meet the so-called Zyablov bound. In particular, we can have linear-time binary codes of rate $\Omega(\epsilon^3)$ to correct a fraction $(1/4-\epsilon)$ of errors for arbitrary $\epsilon > 0$ (Exercise 1 (2)). We point the reader to [GI02, GI05] for further details.

2 List decoding – introduction

2.1 Motivation and definition

Recall that a random binary code $C \in \{0,1\}^n$ with size $2^{1-h(d/n)-o(1)}$ has distance d with high probability. For such codes, we can uniquely decode against $\lfloor (d-1)/2 \rfloor$ errors. And we also know that when $\lfloor (d+1)/2 \rfloor$ errors occur, it must be possible that two codewords get to the same string – therefore we cannot decode against $\lfloor (d+1)/2 \rfloor$ errors in the worst case assumption.

But on the other hand, Shannon's theorem tells us that when d errors happens in a random way (in $\mathrm{BSC}_{d/n}$ channel), such a code is decodable against (roughly) d errors with very high probability. Thus, most d-error patterns are decodable. And this reminds us that the worst-case uniquely decodable requirement, which leads to the limitation of $\lfloor (d-1)/2 \rfloor$-error decodability, might be too pessimistic.

Since we still want to talk about worst-case model (in "Hamming world", rather than "Shannon world"), we can only choose to give up the requirement of uniqueness, and introduce the concept of list decoding.

Definition 11 (list decoding problem of a code C) *Fix a code $C \subseteq \Sigma^n$, given $y \in \Sigma^n$, the list decoding problem (against a fraction p of errors) is to output all codewords of C within Hamming distance pn from y.*

Although in list decoding problem, we allow that a message y is "close" to more than one codewords, but we still want the number of "close" codewords small – or just outputting all of them takes a long time. Thus we come up with the following definition, which measures the list-decodability of a code.

Definition 12 ((p, L)-list decodable) *A code $C \subseteq \Sigma^n$ is (p, L)-list-decodable if $\forall y \in \Sigma^n, |B(y, pn) \cap C| \le L$.*

Remark 13 *$(p, 1)$-list-decodability $\Leftrightarrow$ unique-decodability $\Leftrightarrow$ $\delta(C) > 2p$.*

2.2 Existence of codes that are decodable to a small list

In this section, we justify Definition 11 and Definition 12, by proving the following lemma, which ensures that there exist codes that are decodable to a "small" list.

Lemma 14 *Let $|\Sigma| = q$, $0 < p < 1 - 1/q$. There exists a code $C \subseteq \Sigma^n$ of relative rate $(1 - h_q(p) - 1/L)$ that is (p, L)-list-decodable. In fact, a random $C \subseteq \Sigma^n$ works with high probability.*

Proof: Let $R = 1 - h_q(p) - 1/L$, pick q^{Rn} codewords from Σ^n independently at random, and let C be the (multi-)set of these codewords. When C is not (p, L)-list-decoable, we know that there exists $y \in \Sigma^n$, and such that $|B(y, pn) \cap C| \geq L + 1$.

Therefore, fix y,

$$\begin{aligned}
\mathbf{Pr}_C[|B(y, pn) \cap C| \geq L + 1] &\leq \binom{|C|}{L+1}\Big(\frac{\mathrm{Vol}(B_q(y, pn))}{q^n}\Big)^{L+1} \\
&\leq \Big(\frac{|C|\mathrm{Vol}(B_q(y, pn))}{q^n}\Big)^{L+1} \\
&\leq q^{(R+h_q(p)-1)n(L+1)} \\
&\leq q^{-n(L+1)/L}.
\end{aligned}$$

Thus, by a union bound,

$$\begin{aligned}
\mathbf{Pr}_C[C \text{ is not } (p, L)\text{-list-decodable}] = \mathbf{Pr}_C[\exists y \in \Sigma^n, |B(y, pn) \cap C| \geq L + 1]& \\
\leq q^n q^{-n(L+1)/L} = q^{-n/L}.&
\end{aligned}$$

Thus, only with exponentially small probability C fails to be (p, L)-list-decodable. □

The code given by Lemma 14 is not ensured to be linear code. If we want to get linear list-decodable codes, we use the following fact,

Lemma 15 *For each $S \subseteq \mathbb{F}_q^n \setminus \{0\}$ with size $|S| = L$, there always exists a subset $T \subseteq S$ such that $|T| \geq \log_q(L + 1)$ and the elements in T are linearly independent.*

Then, using similar proof as in Lemma 14, we get

Lemma 16 *A random linear code $C \subseteq \mathbb{F}_q^n$ of rate $(1 - h_q(p) - \frac{1}{\log_q(L+1)}$ is (p, L)-list-decodable with probability $(1 - q^{\Omega(n)})$.*

Remark 17 *The $\frac{1}{\log_q L}$ slack has been improved to $C_{p,q}/L$ recently.*

On the other hand, assuming L is a large constant, we prove that the rate/distance tradeoff shown in Lemma 14 and Lemma 16 is tight.

Lemma 18 *A code $C \subseteq [q]^n$ of rate $(1 - h_q(p) + \epsilon)$ is not $(p, q^{\Omega_\epsilon(n)})$-list-decodable.*

Proof: Pick $y \in [q]^n$ at random. Then,

$$\mathbf{E}_y[|B(y, pn) \cap C|] = \frac{|C|\mathrm{Vol}_q(n, pn)}{q^n} \geq |C|q^{(h_q(p)-1-o(1))n} \geq q^{(\epsilon-o(1))n}.$$

Therefore, there exists $y \in [q]^n$, such that $|B(y, pn) \cap C| \geq q^{(\epsilon-o(1))n}$. □

We also remark that for large sized alphabet, say for alphabet with size $q = 2^{1/\gamma}$, the capacity (rate) of list decoding is almost

$$1 - h_q(p) = 1 - (p \log_q(q - 1) + \frac{h(p)}{\log q}) \geq 1 - (p + \frac{1}{\log q}) = 1 - p - \gamma.$$

Comparing it to Reed-Solomon codes which matches the Singleton bound, we note that unique decoding of Reed-Solomon code works for at most fraction $(1-R)/2$ of errors (where R is the relative distance). But list decoding (with constant-size list) can deal with $p \to 1-R$ errors.

3 List decoding for Reed-Solomon codes (to be continued)

We have proved the optimal rate/distance tradeoff for list decoding. Thus, the main challenge remains now is to approach the optimal rate of $(1-h_q(p))$ with

- explicit codes,
- efficient list decoding algorithms.

In this section, we will show a list decoding algorithm for Reed-Solomon codes, under a general framework using Johnson bound.

3.1 A general framework of constructing list-decodable codes

We have seen many clever construction of codes with good rate and good distance. One might ask whether we can take the advantage of these codes to get codes with nice list-decodable properties. The Johnson bound should come to mind at this stage, which allow us to say something about the list-decodability just based on the distance of a code. Recall the Johnson bound as follows,

Theorem 19 (Johnson bound) *A q-ary code with block length n, relative distance δ is always $(J_q(\delta), O(n))$-list-decodable, where*

$$J_q(\delta) = \Big(1 - \sqrt{1 - \frac{q}{q-1}\delta}\Big)\Big(1 - \frac{1}{q}\Big).$$

In particular,

$$J_2(\delta) = \frac{1 - \sqrt{1-2\delta}}{2}.$$

Indeed, we have also shown a slightly different version of Theorem 19, as follows.

Theorem 20 *A q-ary code with relative distance δ is always*

$$\Big(\Big(1 - \sqrt{1 - \frac{q\delta(1-1/L)}{q-1}}\Big)\Big(1 - 1/q\Big), L\Big)\text{-}$$

list-decodable. In particular, a binary code with relative distance δ is

$$\Big(\frac{1 - \sqrt{1 - 2\delta(1-1/L)}}{2}, L\Big)\text{-}$$

list-decodable.

For large q, we see that a code with relative distance δ is $(1-\sqrt{1-\delta(1-1/L)}, L)$-list-decodable. Thus, for Reed-Solomon codes with rate R and distance almost $(1-R)$, we want to decode it against up to fraction $(1-\sqrt{R})$ of errors.

Remark 21 *Note that since Reed-Solomon codes meet the Singleton bound, $(1-\sqrt{R})$ is the best tradeoff between the fraction of errors can be list decoded and rate, if we work under the framework of Johnson bound (only appeal to distance and Johnson bound).*

3.2 A toy problem

Before we go into the Reed-Solomon list decoding algorithm, we work on the following toy problem as warmup.

The problem. [ALRS99] Fix $p_1(x), p_2(x) \in \mathbb{F}[X]$ as two different degree-k polynomials. With a parameter n ($n > 4k$, n even), the encoding procedure selects n different elements $\alpha_1, \alpha_2, \cdots, \alpha_n \in \mathbb{F}$, chooses a set $S \in [n]$ of size $|S| = n/2$, and computes $y_1, y_2, \cdots, y_n$ as follows,

$$\forall i \in [n], y_i = \begin{cases} p_1(\alpha_i) & i \in S \\ p_2(\alpha_i) & i \in [n] \setminus S \end{cases}.$$

The problem is to design a decoding algorithm, which, given $(\alpha_1, y_1), (\alpha_2, y_2), \cdots, (\alpha_n, y_n)$ output by the encoding procedure, no matter what S the encoding procedure chooses, recovers the two polynomials $p_1(x)$ and $p_2(x)$.

First try – unique decoding. One idea could be to treat $p_1(x)$ as the real message, and all the y_i's computed according to $p_2(x)$ as noise (errors). Then use unique decoder to decode $p_1(x)$, and interpolate $p_2(x)$ from all the pairs (α_i, y_i) that don't agree with $p_1(x)$. But this idea doesn't work, because the codeword $(\alpha_1, y_1), (\alpha_2, y_2), \cdots, (\alpha_n, y_n)$ is equally far away from the message generated solely by $p_1(x)$ and the one generated solely by $p_2(x)$ (i.e., the Hamming distance are all $n/2$). Therefore no unique decoder could figure out whether $p_1(x)$ or $p_2(x)$ is the "unique" original message.

In particular, if we treat $(\alpha_1, y_1), (\alpha_2, y_2), \cdots, (\alpha_n, y_n)$ as a variant of Reed-Solomon code, the Reed-Solomon unique decoder requires at least $(n+k)/2$ agreements between $p_1(x)$ and the message to be decoded. But this is not ensured in the problem.

Solution. We encode the fact that each y_i is either $p_1(\alpha_i)$ or $p_2(\alpha_i)$ as following algebraic statement.

$$\forall i \in [n], (y_i - p_1(\alpha_i))(y_i - p_2(\alpha_i)) = 0.$$

Therefore, if we let

$$Q(x,y) = (y - p_1(x))(y - p_2(x)) = y^2 - (p_1(x) + p_2(x))y + p_1(x)p_2(x) = y^2 - B(x)y + C(x),$$

where $B(x)$ is a degree k polynomial and $C(x)$ is a degree-$2k$ polynomial, we know that $Q(x,y)$ vanishes for each $(x = \alpha_i, y = y_i)$. Thus, our decoding algorithm works as follows.

The algorithm.

(1) By solving a linear system, find a polynomial of form $Q(x,y) = y^2 - B(x)y + C(x)$ such that $B(x)$ is a degree k polynomial, $C(x)$ is a degree-$2k$ polynomial, and for each $i \in [n]$, $Q(\alpha_i, y_i) = 0$.

(2) Factorize $Q(x,y) = (y - f_1(x))(y - f_2(x))$, and output $\{f_1(x), f_2(x)\}$ as $\{p_1(x), p_2(x)\}$.

In the first step of the algorithm, we are always able to find a solution because $(y-p_1(x))(y-p_2(x))$ is a candidate solution. Thus, we need to prove that, on the other hand, we are always able to find out $p_1(x)$ and $p_2(x)$.

Lemma 22 *For any $Q(x,y)$ found in Step (1) of the algorithm, $(y - p_1(x))|Q(x,y)$, and $(y - p_2(x))|Q(x,y)$.*

Proof: By symmetry, we only need to prove that $(y - p_1(x))|Q(x,y)$. Think of Q as $Q(y)$, a univariate polynomial on y, to prove that $(y - \beta)$ is a factor of $Q(y)$, we only need to prove that $Q(\beta) = 0$. Thus, it suffices to prove $Q(x, p_1(x)) \equiv 0$, which implies $(y - p_1(x))|Q(x,y)$.

Let $R(x) = Q(x, p_1(x))$. We see that $R(x)$ has degree at most $2k$. Thus to prove $R(x) \equiv 0$, we only need to find out $(2k+1)$ roots for $R(x)$. Note that there are $n/2 > 2k$ α_i's with $y_i = p_1(\alpha_i)$. For these α_i's, we have $R(\alpha_i) = Q(\alpha_i, p_1(\alpha_i)) = Q(\alpha_i, y_i) = 0$. And this finishes the proof. □

Lemma 22 implies that $p_1(x), p_2(x) \in \{f_1(x), f_2(x)\}$. Thus, when $p_1(x)$ and $p_2(x)$ are different, the algorithm outputs the correct set.

3.3 List decode Reed-Solomon codes in general

The toy problem actually gives us an approach to decode Reed-Solomon codes into a list of size 2, against $n/2$ errors – requiring $t \geq n/2$ agreements (correct places), while unique decoder can only deal with $(n-k)/2$ errors – requiring $t > (n+k)/2$ agreements.

In general, to decode Reed-Solomon codes into a larger list, we consider the following polynomial reconstruction problem. Given n distinct pairs $(\alpha_i, y_i) \in \mathbb{F}^2$, find all polynomials $f \in \mathbb{F}[X]$ of degree at most k, such that the agreement $f(\alpha_i) = y_i$ holds for $\geq t$ values of $i \in [n]$.

In general, we would like t to be small as possible – so that we can deal with more errors. Unique decoding requires $t > (n+k)/2$. To match the Johnson bound (to decode into constant/$O(n)$-sized list), we want solve the polynomial reconstruction problem for $t > \sqrt{kn}$.

In next lecture, we will see a list decoding algorithm for $t > \sqrt{2kn}$ [Sud97].

References

[ALRS99] Sigal Ar, Richard Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488–511, 1999. 9

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 812–821, New York, NY, USA, 2002. ACM. 4, 6

[GI05] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005. 6

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. 10