Contents

1 List decoding of Reed-Solomon codes

1.1 Introduction

An error-correcting code is $C \subseteq \Sigma^n$, where $\Sigma, |\Sigma| = q$ is the alphabet and n is the encoding length.

Define the normalized Hamming distance

$$d(x, y) = \frac{1}{n}|\{i : x_i \neq y_i\}|.$$

We want

$$\delta = \min_{x,y \in C, x \neq y} d(x, y)$$

to be as large as possible (constant as $n \to \infty$). Imagine balls of radius $\delta/2$ around each point. We can send one of $|C|$ messages by mapping $[|C|] \to \Sigma^n$; they can withstand $\delta/2$ errors. The rate of the code is $\frac{\lg |C|}{n}$ (how many bits of actual information are sent compared with encoding length); we want the rate and distance to be high, but there are fundamental limits to what can be achieved.

$\delta/2$ is the unique decoding radius. Unique decoding is not possible beyond $\delta/2$.

In many cases, even if there is not a unique codeword within a given radius ($\frac{\delta}{2} + \varepsilon$, say), there may be a only a small number of codewords.

Definition 1.1: C is (δ, L)-list decodable if for all $x \in \Sigma^n$, $|B(x, \delta) \cap C| \leq L$.

If you choose your code randomly—that is, choose a codeword, removing a $\frac{\delta}{2}$ ball around it, and repeat—with high probability it will have good list decodability, up to distance $(1 - \varepsilon)\delta$. List decoding is a better way to handle talk about how good the code is than stochastically, because it's worst-case.

The maximum δ is the list-decoding radius. Here $L(n)$ is a function of n. The ideal setting is $L(n) = \text{poly}(n)$. Informally the list decoding radius is the maximum δ such that $L(n) = \text{poly}(n)$.

What does a random code give us; what is possible and not possible? What is the capacity of list decoding?

Theorem 1.2: Let $0 < \delta < 1 - \frac{1}{q}$. (Above this error, the noisy word is essentially random, so we can say nothing.) Then there exists a (δ, L)-list decodable code with rate

$$1 - H_q(\delta) - \frac{1}{L}.$$

Here H_q is defined so that

$$|B(0^n, \delta)| \sim q^{H_q(\delta)n}.$$

Proof. Choose $M = q^{Rn}$ codewords at random. We want

$$\mathbb{P}(C \text{ not } (\delta, L)\text{-list decodable}) < 1.$$

Do a union bound. The probability is at most, by the union bound,

$$\binom{M}{L+1} q^n \left(\frac{|B(x,\delta)|}{q^n}\right)^{L+1} < 1.$$

(q^n is the number of points; the probability of $L+1$ points in a ball centered at a point is $\le \binom{M}{L+1}\left(\frac{|B(x,\delta)|}{q^n}\right)^{L+1}$.) □

This is almost tight.

Theorem 1.3: If a code is (δ, L)-list decodable with

$$R \ge 1 - H_q(\delta) + \varepsilon$$

then $L(n) \ge q^{\frac{\varepsilon n}{2}}$.

A random ball will contain too many points.

Proof.

$$\begin{aligned}\mathbb{E}_{x \in_R \Sigma^n}[|B(x,\delta) \cap C|] &= |C| \frac{|B(0^n, \delta)|}{q^n} \\ &\ge 2^{Rn} q^{n(H_q(\delta)-1)} \\ &\ge q^{\varepsilon n/2}.\end{aligned}$$

□

When $q \to \infty$, $H_q(\delta) \to \delta$. We can achieve rate $R \ge 1 - \delta - \varepsilon$ with alphabet of size $2^{O\left(\frac{1}{\varepsilon}\right)}$ and list size $O\left(\frac{1}{\varepsilon}\right)$.

Graph: the achievable rates are below the line $\delta + R = 1$.

This is what is known existentially.

The current known explicit codes can achieve $R \ge 1 - \delta - \varepsilon$ with alphabet of size $2^{\text{poly}\left(\frac{1}{\varepsilon}\right)}$ and list size $n^{O\left(\frac{1}{\varepsilon}\right)}$. The idea is a folded Reed-Solomon code concatenated with an asymptotically good list-decodable code.

Theorem 1.4 (Johnson bound)**:** Given $C \subseteq \Sigma^n$ with min-distance $\delta = 1-\varepsilon$, then C is $(1-\sqrt{\varepsilon}, \text{poly}(n))$ list-decodable.

The Johnson bound says that if you slightly increase the radius, there cannot be an exponential number of codewords.

Theorem 1.5 (Singleton bound)**:** For any codewith distance δ, $R \le 1-\delta$.

MDS (maximal distance separable) codes are where $R+\delta=1$, $\delta = 1-R$, R$=\varepsilon$. MDS codes are $(1-\sqrt{R}, \text{poly}(N))$ list-decodable.

1.2 Reed-Solomon Codes

We introduce the Reed-Solomon code $RS_{k,n,\mathbb{F}}$, $k < n \le q$.

These are MDS codes. They achieve $R+\delta = 1$ so are $(1-\sqrt{R}, \text{poly}(n))$-list decodable.

Let $\Sigma = \mathbb{F}_q =: \mathbb{F}$. Let

$$C = \{\text{evaluations of degree } k \text{ polynomials on } \{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{F}\}$$

where $\alpha_1, \dots, \alpha_n$ are distinct and fixed. Codewords correspond to degree k polynomials in $\mathbb{F}[x]$. 2 distinct degree k polynomials can only agree on k points, so $\delta = \frac{n-k}{n} = 1 - \frac{k}{n}$. The rate is $\frac{\log_q(q^{k+1})}{n} = \frac{k+1}{n}$. The Reed-Solomon code is $(1-\sqrt{\frac{k}{n}}, \text{poly}(n))$-list decodable.

What is the algorithm? Given a set of points, we need to find all polynomials passing through enough of those points.

Problem 1.6: Given $(\alpha_1, y_1), \dots, (\alpha_n, y_n)$, find all polynomials of degree $\le k$ such that $p(\alpha_i) = y_i$ for at least $\sqrt{nk}$ indices $i \in [n]$.

Madha Sudan (90's) showed that you can do this with $\sqrt{2nk}$.

Theorem 1.7: There is a polynomial-time algorithm (given in the proof) that given n points as above, finds all degree $\le k$ polynomials agreeing on $\ge 2\sqrt{nk}$ points.

Proof. The algorithm is as follows. Note this algorithm will work even if the α_i are not distinct.

Define the $(1,k)$-weighted degree of a polynomial $Q(x,y)$ as $\deg Q(X, Y^k)$. The strategy is as follows.

We will find a low $(1,k)$-weighted degree polynomial $Q(x,y)$ of degree D such that $Q(\alpha_i, y_i) = 0$ for all i.

Look at $R(x) = Q(x, p(x))$ where $p \in L$. We have

$$\deg R \le D.$$

For all i such that $p(\alpha_i) = y_i$,

$$R(\alpha_i) = Q(\alpha_i, p(\alpha_i)) = Q(\alpha_i, y_i) = 0.$$

Suppose R has at least t roots and $\deg R \leq D$. If we arrange so that $t > D$, then $R(x) \equiv 0$ identically. Then $Q(x, p(x)) \equiv 0$ implies $y - p(x) \mid Q(x,y)$. (There is a deterministic algorithm to factor bivariate polynomials.) Factor R to find all factors in the form $y - p(x)$; then output those polynomials $p(x)$.

Now we just need to find the polynomial $R(x)$; this is polynomial interpolation.

The number of coefficients in $Q(x,y)$ is $\frac{1}{k}\binom{D+2}{2}$. We need to satisfy n linear constraints; they are linear homogeneous equations in the coefficients. If $\frac{\binom{D+2}{2}}{k} > n$ then there is a nonzero solution: a nonzero $\varphi(x,y)$ of $(1,k)$ weight degree $\leq D$ with $\varphi(\alpha_i) = y_i$ for all i.

If the number of roots is $t > D \approx \sqrt{2kn}$, then we can find all polynomials with agreement t. □

Guruswami and Sudan improved this to $\sqrt{kn}$.

Consider $k = 1$, $\mathbb{F} = \mathbb{R}$, find all lines which pass through at least 3 points. If you can interpolate a degree 2 polynomial through all these points, get all lines as lines within the curve. The degree 2 curve will be 2 lines.

Let's look at a small example to see how to improve the bound.

Consider the following picture (see notebook). Here $n = 10$ and $t = 4$. Here we can choose $k = 4$. However it can only have 4 linear factors. The maximum D is 3. Peculiar: through each point there are 2 lines. An algebraic curve passing all points should vanish at the points with multiplicity 2. Fit a polynomial which vanishes with degree 2 at the points.

First define multiplicity.

Definition 1.8: $Q(x,y)$ vanishes with multiplicity r at (α, β) if $Q(x+\alpha, y+\beta)$ doesn't have any monomial of degree $\leq r$.

Lemma 1.9: Let $Q(x,y)$ be with $(1,k)$ degree $\leq D$, vanishing at (α_i, y_i) with multiplicity r for all $i \in [n]$. Let P be a degree k polynomial with agreement $t > D/r$. Then $y - p(x) \mid Q(x,y)$.

Proof. Let $p \in L$, $P(\alpha_i) = y_i$. Define

$$Q^i(x,y) = Q(x+\alpha_i, y+y_i);$$

it has no monomials of degree $< r$. Then

$$\begin{aligned} R(x) &= Q(x, P(x)) \\ &= Q^i(x-\alpha_i, P(x) - y_i) \\ &= Q^i(x-\alpha_i, \underbrace{P(x) - p(\alpha_i)}_{x-\alpha_i \mid P(x)-P(\alpha_i)}). \end{aligned}$$

Q^i has no monomials of degree $< r$. Thus $(x-\alpha_i)^r \mid R(x)$. The number of linear factors is $tr > D$. Thus $R(x) \equiv 0$ and $y - p(x) \mid Q(x,y)$. □

The number of coefficients in $Q(x,y)$ of $(1,k)$-degree D is $\frac{1}{k}\binom{D+2}{D}$. The number of homogeneous linear equations is $n\binom{r+1}{2}$. So a nonzero Q exists if $\frac{1}{k}\binom{D+2}{2} > n\binom{r+1}{2}$.

Choose $D = \sqrt{knr(r+1)}$. Let $t = \frac{D}{r} = \sqrt{kn(1+\frac{1}{r})}$.
Make r large enough, we approach the Johnson bound.
The number of polynomials in the list is at most $\frac{D}{k} = \sqrt{\frac{nr(r+1)}{k}}$, the y-degree.
Conclusion:

1. If $t > \sqrt{kn}$ the list size is $\leq n^{\varepsilon}$ and we can find all of them.

2. The Reed Solomon code of rate R can be list decoded up to $1 - \sqrt{(1+\varepsilon)R}$ errors with best size $\frac{1}{\varepsilon\sqrt{R}}$.

This method (the polynomial method) is very flexible. We give another application, the list recovering problem. Given $x \in C$, a noisy channel turns each coordinate into a set s_i and spits out $s_1, \ldots, s_n$, $|s_i| \leq \ell$. We have $|s_i| \leq l$ such that for at least $1-\delta$ fraction of i's, $x_i \in s_i$. Find all x such that $x_i \in s_i$ for at least $1-\delta$ fraction of i's.

$$\left| \left(\bigcup_{y \in S_1 \times \cdots \times S_n} B(y,\delta) \right) \cap C \right| \leq L.$$

Reed-Solomon codes are also good list-recoverable codes.

If $t > \sqrt{knl}$, where t is the number of i such that $x_i \in S_i$, then the Reed-Solomon code is $(1-\sqrt{kl}, l, O(n^2 l^2) = L)$ list recoverable.

We only achieve $1 - \sqrt{R}$. We want to attain $(1-R, L)$. Guruswami and Rudra came up with folded Reed-Solomon codes. Take a generator γ of $\mathbb{F}_q^{\times}$, $n = q-1$. $\alpha_1, \ldots, \alpha_n$ are $S = \{1, \gamma, \ldots, \gamma^{q-1}\}$. Consider blocks. $\mathbb{F}^m$ by m. $P(1), \ldots P(r), \ldots, P(r^{q-1})$.

It's an open problem to make $n^{O\left(\frac{1}{\varepsilon}\right)}$ independent of n. Use concatention to get length down.

You can concatenate a list-recoverable code with a list-decodable codes to get a list-decodable code. $C_{\text{out}} \circ C_{\text{in}}$.

2 $SL = L$

Reingold, Vadhan, and Wigderson.

There are many stories in this problem.

One of the most fundamental questions in theoretical CS is the following.

Claim 2.1: Randomness is useless.

This claim comes from very recent years. 20 years ago people actually believe it's useful (Papadimitriou gave $BPP \neq P$ as a homework exercise in his book).

There are 2 questions: is randomness useful in time and in space?

1. $BPP \overset{?}{P}$

2. $RL \overset{?}{L}$

We focus on the second problem. The first problem is wide open: we don't know $BPP \overset{?}{\in} DTIME(2^{o(n)})$. We know more about the second question: Savitch proves

$$BPP \subseteq DSPACE(\ln^2 n).$$

We can do better: Saks and Zhou in 1999 show

$$RL \subseteq DSPACE(\ln^{\frac{3}{2}} n).$$

(They actual show this for BPP.) Think of RL as (undirected) $s-t$ nonconnectivity problem. $coRL$ is interesting because it contains $s-t$ connectivity. (RL doesn't have a complete problem.) For $P \overset{?}{=} NP$, just look at a complete problem; here we don't have one.

Here we show $SL = L$. Reingold showed this in 2004.

For simplicity, just think of SL as one problem, undirected $s-t$ connectivity.

Consider a graph.

Jieming starts from a vertex and wants to find a way to Nanjing. Jieming has very little memory, and cannot remember the whole structure of the graph. He can't remember much more than the name of a city.

The standard way to solve this is to take a uniform random walk on the graph. If the graph has $|V| = n$, after $n^2 \ln n$ steps, there is a high probability that he will have visited Nanjing.

But for a general graph, it's necessary to flip $n^2 \ln n$ coins: Consider 2 complete graphs with a bridge: It takes order of n time to hit the bridge vertex, and it has $O\left(\frac{1}{n}\right)$ chance of crossing the bridge.

For which graphs can we do this randomized routing faster than the worst case? A natural family is the family of expaner graphs.

Expander graphs have rapid mixing under a random walk. There are 2 definitions.

Definition 2.2: A graph $G = (V, E)$ is a **λ-edge expander** if for all sets $S \subseteq V$, $|S| \le \frac{|V|}{2}$,

$$\frac{E(S, \overline{S})}{\text{Vol}(S)} \ge \lambda.$$

Here, $\text{Vol}(S) = \sum_{v \in S} \deg(v)$. (We'll focus on the simple case when the graph is d-regular, i.e, for all $v \in V$¡ $\deg(v) = d$.) (Pictorially, a subset of $G = (V, E)$ is very spiky, like a sea urchin.)

A graph $G = (V, E)$ is a **λ-spectral expander**if $\lambda_2(L(G)) \ge \lambda$. The Laplacian $L(G)$ is defined from the adjacency matrix A_G. A_G is defined by $(A_G)_{ij} = 1$ if there is a edge between i and j. M is the random walk matrix $\frac{1}{d} A_G$, and

$$L = I - M.$$

(Thinking of this matrix as an operator, $(Lx)_i = \frac{1}{d} \sum_{(i,j) \in E} x_{ij}$, it flows from a vertex to adjacent vertices.)

The Laplacian originates in differential geometry, $L = \text{div}\,\nabla$.

These 2 definitions are quite close.

Theorem 2.3: Let $h(G), \lambda(G)$ denote the edge and spectral expansions of G. Then

$$\lambda(G) \le h(G) \le \sqrt{2\lambda(G)}.$$

The LHS is trivial, the RHS is Cheeger's inequality.

People in CS use a third definition that is more useful.

Definition 2.4: A graph is a λ-expander if for all $x \perp u = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, x \neq 0$,

$$\left| \frac{x^T M x}{x^T x} \right| \le \lambda.$$

(For edge and spectral expansion we want λ to be large. For random walk expansion we want λ to be close to 0.) $\lambda(L)$ range from 0 to 2, while $\lambda(M)$ range from 1 to -1.

Note that

$$\left| \frac{x^T (I-L) x}{x^T x} \right| = \left| 1 - \frac{x^T L x}{x^T x} \right|$$

so spectral and random-walk expansion are not quite the same. For bipartite graphs, $\lambda(M) = 1$: random walks are not mixing at all. The side you're on depends on the parity.

For the zig-zag product they consider the third definition.

2.1 Zig-zag product

Think of G_1 having large size and degree (N_1, D_1), G_2 having small size and degree (N_2, D_2), and suppose $D_1 = N_2$.

$$G_3 = G_1 \text{Ⓩ} G_2$$

has large size $N_1 N_2$ and small degree, D_2^2.

Let $\lambda_M(G_i) = \lambda_i$ denote the random walk expansion. Then (a naive bound)

$$\lambda_M(G_3) \le \lambda_1 + \lambda_2 + \lambda_2^2.$$

It's easy to construct a large degree expander. It's easy to construct small-size expanders (try all possible graphs). It's easy to construct G_1, G_2, but an expander like G_3 is hard to construct. It takes as input 2 easy-to-construct expanders and outputs a harder to construct graph that's an expander.

Replace each vertex of the original graph with a copy of the second graph G_2. Its size is hence $N_1 N_2$.

Each vertex in G_3 is labeled $(v, u) \in G_1 \times G_2$. The (i, j)th neighbor is defined as follows. Take u' the ith neighbor of u in G_2, $u \xrightarrow{i} u'$. Take the u'th neighbor of v in G_1; move to the cloud of w; $v \xrightarrow{u'} w$ in G_2. Now consider $\tilde{u}$ such that $w \xrightarrow{\tilde{u}} v$ in G_1; we move to $(w, \tilde{u})$. Now move $\tilde{u} \xrightarrow{j} \tilde{u}'$ in G_2. Define

$$(v, u) \xrightarrow{(i,j)} (w, \tilde{u}').$$

What's the magic of the zig-zag product? We claim

$$A_3 := A(G_3) = \widetilde{A_2}\widetilde{A_1}\widetilde{A_2}$$

where $\widetilde{B_2} = I \otimes A_2$ has N_1 blocks and each is a copy of A_2, and $\widetilde{A_1}$ is a permutation (actually matching) matrix where $(\widetilde{A_1})_{((v,u),(w,u'))} = 1$ if

$$v \xrightarrow{u,G_1} w \xrightarrow{u',G_1} v.$$

Consider when $(u,v) \xrightarrow{(i,j)} (w,\widetilde{u}')$:

$$u \xrightarrow{i,G_2} u'$$

$$v \xrightarrow{u',G_1} w \xrightarrow{\widetilde{u},G_1} v$$

$$\widetilde{u} \xrightarrow{j} \widetilde{u}'$$

What is the action of $\widetilde{A_2}\widetilde{A_1}\widetilde{A_2}$ on $e_{(v,u)}$? It goes to $\sum_{uu' \in G_2} e_{(v,u')} = e_v \otimes A_2 e_u$. "Stay in the block of v, move to all the neighbors of u."

In the second step, we move across blocks as given by $\widetilde{A_1}$.

We need to show

$$\forall x \in \mathbb{R}^{N_1 \times N_2}, x \perp u \implies \frac{x^T M_3 x}{x^T x} \le \lambda_1 + \lambda_2 + \lambda_2^2.$$

Decompose x as a vector that is uniform on each block,

$$x = \underbrace{\alpha \otimes u}_{\alpha^{\parallel}} + \underbrace{x'}_{\alpha^{\perp}}$$

where x' has blocks $\widetilde{\alpha_i} \perp u_i$. Then (noting $\widetilde{A_2}\alpha^{\parallel} = \alpha^{\parallel}$),

$$\begin{aligned}
\left\langle x, \widetilde{A_2}\widetilde{A_1}\widetilde{A_2}x \right\rangle &= \left\langle \widetilde{A_2}x, \widetilde{A_1}\widetilde{A_2}x \right\rangle \\
&= \left\langle \widetilde{A_2}(\alpha^{\parallel} + \alpha^{\perp}), \widetilde{A_1}\widetilde{A_2}(\alpha^{\parallel} + \alpha^{\perp}) \right\rangle \\
&= \left\langle \alpha^{\parallel} + \widetilde{A_2}\alpha^{\perp}, \widetilde{A_1}(\alpha^{\parallel} + \widetilde{A_2}\alpha^{\perp}) \right\rangle \\
&= \left\langle \alpha^{\parallel}, \widetilde{A_1}\alpha^{\parallel} \right\rangle + \left\langle \alpha^{\parallel}, \widetilde{A_1}\widetilde{A_2}\alpha^{\perp} \right\rangle + \left\langle \widetilde{A_2}\alpha^{\perp}, \widetilde{A_1}\alpha^{\parallel} \right\rangle + \left\langle \widetilde{A_2}\alpha^{\perp}, \widetilde{A_1}\widetilde{A_2}\alpha^{\perp} \right\rangle \\
&\le \lambda_1 + 2\lambda_2 + \lambda_2^2
\end{aligned}$$

Because $\alpha_1 + \cdots + \alpha_{N_1} = 0$ we can use the expansion properties of G_1 to get the λ_1 bound for the first term.

The idea:

1. Suppose G has parameters (N, D) and expansion λ. (say $1 - \frac{1}{ND}$)
2. G^2 has parameters (N, D^2), expansion λ^2.

3. Take H with parameters $(D^2, \sqrt{D}), \lambda_2$. Then

$$G^2 \textcircled{Z} H = G_{\text{new}}$$

   has parameters (ND^2, D) expansion $\lambda(G_{\text{new}}) \leq \lambda^2 + 2\lambda_2 + \lambda_2^2 \leq \lambda^2 + \varepsilon$.

We go from G with parameters (N, D, λ) to G_{new} with parameters $(ND, D, \lambda^2 + \varepsilon)$. Call this operator Z. Doing this operator t times, $Z^t G = (ND^{2t}, D, \lambda^{2^t} + \varepsilon')$. Then set $t = \lg\left(\frac{ND}{2}\right)$ to get $Z^t G$ with parameters $(N^2 D, D, \frac{1}{e} + \varepsilon)$. (We're cheating a little, using the naive bound. We need a better bound to get ε small: G_3 has expansion $\frac{1}{2}(1 - \lambda_2^2)\lambda_1 + \frac{1}{2}\sqrt{(1 - \lambda_2^2)^2 \lambda_1^2 + t\lambda_2^2}$. To go from our proof to the real proof, you just need to note $\alpha^{\parallel} \perp \alpha^{\perp}$, $\alpha^{\parallel} \perp \widetilde{A_2}\alpha^{\perp}$. Plug in this picture into the formula.