CS 229r Essential Coding Theory Feb 28, 2017

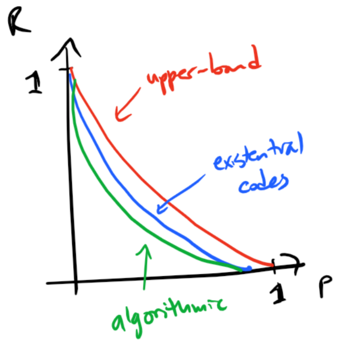
Lecture 11

Instructor: Madhu Sudan *Scribe: Preetum Nakkiran*

In this lecture, we will look at the combinatorics of list-decoding, and then introduce an algorithm for list-decoding Reed-Solomon codes.

1 Prelude: Random vs. Adversarial Errors

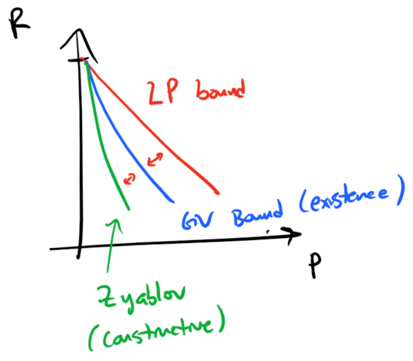
First, as motivation, let us compare our knowledge so far in the random vs. adversarial error model. In both these cases, we have the following picture of how the achievable rate varies with increasing fraction of errors p:



The three curves are (1) the upper-bound on rate, (2) the existential lower-bound, and (3) the explicit (constructive / algorithmic) lower-bound.

In the random error model, where we require the decoder to succeed with high probability, all these curves are the same: at rate $R = 1 - H(p)$ for the BSC_p channel.

In the adversarial error model, where we require unique decoding, these curves are different: There is a gap between (1) the LP upper-bound, (2) the GV greedy existential lower-bound, and (3) the constructive Zyablov lower-bound.



It may seem from the picture that this gap is arising because we're dealing with adversarial as opposed to random errors. However, the moral of this lecture is that **this gap is not a feature of the adversarial**

error model, bur rather a feature of our notion of decoder success. Specifically, if we relax the notion of unique-decoding to that of list-decoding, then we can actually achieve matching upper and lower bounds.

2 List Decoding

Let the encoder and decoder over alphabet Σ be

$$E : \Sigma^k \to \Sigma^n$$

$$D : \Sigma^n \to (\Sigma^k)^L$$

where we think of the decoder outputting a *list of messages* of size L.

We consider the decoding to be successful if the original message is in the decoded list:

$$m \in D(E(m) + \eta)$$

where η is the (adversarial) noise.

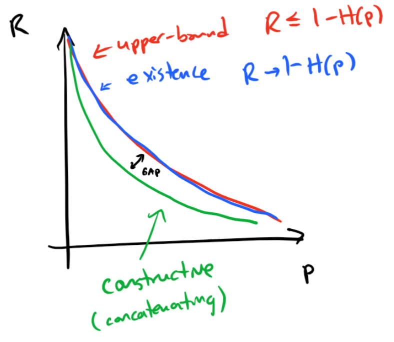
This notion is interesting when the list-size L is short: either $L =$ a constant growing very slowly, or even $L = poly(n)$.

2.1 Final Results

As a preview, here is what is achievable in the list-decodable setting.

For list sizes $L = poly(n)$, and fraction of (adversarial) errors p:

1. The rate is at most $R \leq 1 - H(p)$
2. There exist (non-constructive) codes approaching $R = 1 - H(p)$.
3. There is still a small gap for what we can achieve constructively (at least for rates bounded away from 0 and 1).



2.2 Combinatorics

Suppose we have a list-decodable code that can decode up to p-fraction of errors with list size L. The decoder would receive some word $x \in \Sigma^n$, and would need to output all the codewords within distance pn from x (by correctness: since it must output the original message, and all these codewords are potential original messages). Thus, the number of of codewords at distance $\leq pn$ from x must be at most L if the decoder is to output a short list size. We formalize this in the following definition of a list-decodable code:

Definition 1. *A code $C \subseteq \Sigma^n$, with $|C| = |\Sigma|^k$ is a (p, L)*-list-decodable code *if*

$$\forall x \in \Sigma^n : |Ball(pn, x) \cap C| \leq L$$

where $Ball(pn, x)$ is the Hamming ball of radius pn around point x.

Note that the condition holds *for all $x \in \Sigma^n$*, not just those x that are "near codewords". (But this is not a significant strenghening – only those x that are in fact near codewords are at risk of violating the list size).

Now, for a given code C, we are interested in the maximum p such that C is $(p, L = poly(n))$ list decodable. This p is known as the "list-decoding radius" of the code.

Then, we can ask how the list-decoding radius relates to other combinatorial properties of the code. Specifically:

1. **List-decoding radius vs. distance:** Here we will show that distance implies list-decodability, and get a statement like "For every code of distance δ, the list-decoding radius is at least $f(\delta)$". This is a universal bound across all codes.

2. **List-decoding radius vs. rate:** Here we ask, what is the maximum possible list-decoding radius among codes of a given rate? This is an existential statement – we require only one such code.

2.2.1 List-decoding radius vs. distance

The Johnson Bound (from pset 2) directly gives us a relation between distance and list-decodability.

The Johnson Bound states that: Suppose a q-ary code has relative distance

$$\delta = (\frac{q-1}{q})(1 - \varepsilon)$$

(which is only a factor $(1 - \varepsilon)$ below the maximum possible distance for q-ary codes). Then, the list-decoding radius (for poly-sized lists) is at least

$$p = (\frac{q-1}{q})(1 - \sqrt{\varepsilon})$$

This bound is good for small ε: we can decode up to almost the distance. But, if we want codes of good rate (eg, $\varepsilon = 1/2$), then this gap is very large.

In fact, the Johnson Bound is tight [1] so we can't expect better list-decoding in general, among all codes of a given distance. However, we will later see how to do better with specific codes (Folded Reed-Solomon Codes).

2.2.2 List-decoding radius vs. rate

Notice that Shannon's capacity upper-bound directly implies the following: The list-decoding radius p (for poly-sized lists) vs. the rate R satisfies:

$$R \leq 1 - H(p)$$

(Because, if a higher rate were possible with poly-sized lists, then we could get unique decoding with inverse-poly success probability for rates $R > 1 - H(p)$, just by outputting a random element of the list. This is impossible, since Shannon proves we must have exponentially small success in this rate regime).

Now we will show that there exist codes which approach this bound.

Theorem 2. *There exist $(p, L = poly(n))$-list-decodable codes for any rate $R < 1 - H(p)$.*

[1] For $q = 2$, consider picking a code by picking iid random codewords, with each coordinate an iid $\sqrt{\varepsilon}$-biased coin (biased towards 0). Then the weight of codewords (or, their distance from the orgin) is $\approx 1/2(1 - \sqrt{\varepsilon})$. But their pairwise distances are $\approx 1/2(1 - \varepsilon)$ (since the XOR of two $\sqrt{\varepsilon}$ biased coins is ε-biased).

In particular, random codes achieve this.

We will prove it in the binary case for clarity, but it generalizes directly.

Proof. Pick the encoding function $E : \{0,1\}^k \to \{0,1\}^n$ at random.

The idea is, we will bound the probability that there exists some point x for which "too many" ($\geq L$) codewords land in its ball of radius pn.

For a fixed $x \in \{0,1\}^n$, and fixed set of L distinct messages $m_1, \dots m_L$, we can bound the probability that all L codewords land in the ball of radius pn around x as:

$$\Pr[\forall i \in [L] : E(m_i) \in Ball(pn, x)] \leq \left(\frac{2^{nH(p)}}{2^n}\right)^L$$

(since the Ball has volume $\approx 2^{nH(p)}$).

Then, the probability there exists some $x, m_1, \dots m_L$ that violate the condition above is (by union bound):

$$\begin{aligned}\Pr[\exists x \in \{0,1\}^k, m_1, \dots m_L \in \{0,1\}^k, \text{ s.t. } \forall i \in [L] : E(m_i) \in Ball(pn, x)] &\leq 2^n (2^k)^L \left(\frac{2^{nH(p)}}{2^n}\right)^L \\ &= 2^{-(\varepsilon L - 1)n}\end{aligned}$$

where we set $R = 1 - H(p) - \varepsilon$, and $k = Rn$. Thus, we cane set L large enough to make this probability exponentially small, giving us a (p, L)-list-decodable code with rate $R = 1 - H(p) - \varepsilon$ (in particular, $L = poly(1/\varepsilon)$ is sufficient). □

Remark This result holds for random linear codes as well, with a more careful analysis.

For constructive codes, here is the current state of affairs:

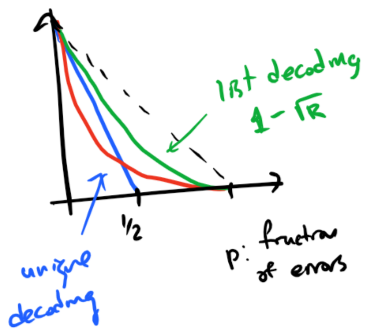
- For all algebraic codes, we have efficient list-decoding up to the Johnson Bound.
- For Reed-Muller codes, we can decode up to the distance (for RM with $r = O(1), q = O(1), m \to \infty$).
- For Reed-Solomon codes, it is still an open problem if it's possible to beat the Johnson Bound (though there are some indications that the Johnson Bound is tight here).
- For Folded Reed-Solomon Codes (which we will cover next lecture), we can list-decode up to the capacity!

3 List-Decoding Reed Solomon Codes

Suppose we have a good code for large alphabets ($q \to \infty$), with rate R and distance close to $1 - R$. This implies (by Johnson Bound) a list-decoding radius of $1 - \sqrt{R}$.

In the figure below, the green line is the list-decoding radius implied by distance ($p = 1 - \sqrt{R}$), the blue line is the radius for unique decoding ($p = 1/2(1 - R)$) – which is never better than the green line – and the red line is what we will show today: an algorithm to decode up to $p = 1 - 2\sqrt{R}$ errors.

Remark It is possible to improve the decoding algorithm to work up to $p = 1 - \sqrt{R}$, matching the Johnson Bound. We'll see this next lecture.



3.1 List Decoding Problem

We consider the following *List Decoding Problem*:

Given: Evaluation points $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$, and values $y_1, \dots y_n \in \mathbb{F}_q$.

Goal: Find all degree-k polynomials p, such that

$$|\{i : p(\alpha_i) = y_i\}| \geq 2k\sqrt{n}$$

(that is, such that p agrees with at least $k\sqrt{n}$ points (α_i, y_i)).

Note that we eventually want to achieve this goal for agreement only $O(\sqrt{kn})$ instead of $k\sqrt{n}$, since this will let us achieve constant rate (k linear in n). In particular, if we achieve the goal with agreement only $O(\sqrt{kn})$, this corresponds to $1 - O(\sqrt{k/n}) = 1 - O(\sqrt{R})$ fraction of errors, meeting the Johnson Bound (up to the constant). It turns out this will be possible with a simple modification after we achieve the easier goal above.

3.2 List-Decoding Algorithm

Here is the list-decoding algorithm for Reed-Solomon codes, by Madhu Sudan.

1. Find an algebraic "fit" for all the points. That is, find a bivariate polynomial $Q(x, y)$ such that:
   - $\deg_x Q \leq \sqrt{n}$
   - $\deg_y Q \leq \sqrt{n}$
   - $Q(\alpha_i, y_i) = 0 \quad \forall i \in [n]$
   - $Q \not\equiv 0$.
2. Factor $Q(x, y) = Q_1(x, y)Q_2(x, y) \dots$, and if some factor is of the form $Q_i(x, y) = y - p_i(x)$ for degree-k p_i, report the polynomial p_i.

Remark We can consider this a generalization of the Berlekamp-Welch algorithm: Berlekamp-Welch effectivly considers $Q(x, y) = E(x)y - W(x)$, where E is the error-locator polynomial, and tries to factor it as $Q(x, y) = E(x)(y - p(x))$.

3.3 Analysis

First, Step 1 is clearly polynomial time, since it is is simply solving a linear system (in the coefficients of Q). It turns out Step 2 also possible in polynomial time (which was known previously, and we will black-box assume here).

To prove correctness, it remains to show:

1. The linear system in Step 1 is always feasible.
2. If a "good" polynomial p (with high agreement) exists, we will actually find it in Step 2.

Point (1) is easy, since the linear system is underconstrained: we have n constraints, but there are $(\sqrt{n}+1)^2 > n$ coefficients in Q.

Point (2) follows from Bézout's theorem in the plane:

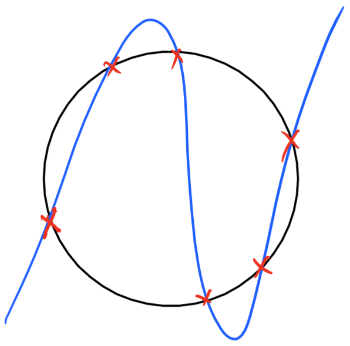
Theorem 3 (Bézout's theorem in the plane)**.** *Suppose $Q_1(x,y)$ and $Q_2(x,y)$ are of total degree d_1 and d_2 respectivly. If*

$$|\{(\alpha,\beta) : Q_1(\alpha,\beta) = Q_2(\alpha,\beta) = 0\}| > d_1 d_2$$

then

$$gcd(Q_1, Q_2) \neq 1$$

Bézout's theorem says that two coprime polynomials can't have too many common zeros. For example, a cubic and a quadratic (say, a circle in $\mathbb{R}^2$) can intersect in at most 6 points unless there is a common factor (eg, if the cubic is the product of the circle with a line).



Now we can prove (2):

Lemma 4. *If there exists a degree-k polynomial p such that*

$$|\{i : p(\alpha_i) = y_i\}| > 2k\sqrt{n}$$

then $Q(x,y)$ in the algorithm above will contain the factor $(y - p(x))$. In particular, since this factor is irreducible, the algorithm will output p.

Proof. Apply Bézout's theorem to $Q_1(x,y) := Q(x,y)$ of total degree $2\sqrt{n}$, and $Q_2(x,y) := y - p(x)$ of total degree k. Since they have $> 2k\sqrt{n}$ common zeros by assumption (Q interpolates all the points (α_i, y_i), and p agrees with at least $2k\sqrt{n}$ of these), they must have a common factor by Bézout's theorem. But $Q_2 = y - p(x)$ is irreducible, so it must be a factor of $Q(x,y)$. $\square$

This completes the analysis of the algorithm for list-decoding Reed-Solomon codes.

Remark To achieve the goal with agreement only $O(\sqrt{kn})$ instead of $2k\sqrt{n}$ as we did above, it suffices to modify the bounds on the individual degrees of Q such that $\deg_X \leq \sqrt{nk}$ and $\deg_Y \leq \sqrt{n/k}$. This better balances the degrees, and we can use a more refined version of Bézout's theorem to conclude that this requires agreement only $2\sqrt{nk}$.

To see why we may want to balance the degrees like this, note that in the ideal case (with no errors), we have $Q(x,y) = y - p(x)$ – which is of degree k in x but only degree 1 in y.

References