0D2442 AUTUMN 2014 LECTURE 13

ALGORITHMIC TASKS IN CODING THEORY

(1) Efficient encoding
(2) Efficient decoding

Task (1) divides into:

(a) Given family of codes $\mathcal{C}$ and index i, compute encoding circuit (or generator matrix for linear codes) for C_i

(b) Given encoding circuit / generator matrix and message m, compute $\mathcal{E}(m)$

Given (a), (b) is easy. Won't discuss encoding further — intuitively, it should be the easier of the two problems

Task (2): What do we mean by decoding? What is the error model?

Hamming model: - Adversary chooses message m
- After encoding, adversary can introduce up to t errors in $\mathcal{E}(m)$

Hamming problem: Given $R \in (0,1)$, find largest τ and efficiently computable families of functions $\{\mathcal{E}_n\}_{n\in\mathbb{N}}$, $\{D_n\}_{n\in\mathbb{N}}$

$$\mathcal{E}_n : \{0,1\}^{Rn} \to \{0,1\}^n \qquad D_n : \{0,1\}^n \to \{0,1\}^{Rn}$$

such that $\forall y \in B(0,\tau n)$ $D_n(\mathcal{E}_n(m)+y) = m$

DECODING PROBLEMS

Won't discuss complexity issues, but many flavors of decoding problem NP-complete in general, like finding NEAREST CODEWORD

II

(1) UNAMBIGUOUS DECODING

For fixed family of codes $\mathcal{C}$

Given index i, vector $r \in \mathbb{F}_{q_i}^{n_i}$

Find codeword $c \in C_i$ s.t. $\Delta(c,r) < \Delta(C_i)/2$ if such a codeword exists

Assumes $\Delta(C_i)$ known or can be computed

Answer is unique if it exists.

Could relax uniqueness requirement and ask for the following

(2) BOUNDED DISTANCE DECODING

Family of codes $\mathcal{C}$ fixed

Error function t fixed

Given index i and $r \in \mathbb{F}_{q_i}^{n_i}$, find any codeword $c \in C_i$ such that $\Delta(c,r) \leq t(i)$ if such a codeword exists.

There are some codes for which this is possible for $t(i) >> \Delta(C_i)/2$. This is done by solving a slightly harder problem

(3) LIST DECODING

Family of codes $\mathcal{C}$ and error function t fixed

Given index i and vector $r \in \mathbb{F}_{q_i}^{n_i}$, find list of all codewords $c \in C_i$ such that $\Delta(c,r) \leq t(i)$.

For efficient list decoding to be possible, in particular need lists to be of at most poly size
Ideally, would like to solve list-decoding problem for $t(i) = \Delta(C_i)$.
Lots of progress on this question in last 10-15 years — beyond the scope of this course

Today, want to do unambiguous decoding of RS codes

REED-SOLOMON DECODING

Input: n distinct elements $x_1, \ldots, x_n \in \mathbb{F}_q$
n elements $y_1, \ldots, y_n \in \mathbb{F}_q$
parameters k and $t \leq \frac{n-k}{2}$

Output: Polynomial $p \in \mathbb{F}_q[x]$ of degree $< k$ such that $p(x_i) \neq y_i$ for at most t values of $i \in [n]$

Let us discuss this problem a bit. Observations:

(i) Problem is clearly in NP — proposed solution can be verified efficiently

(ii) Is it in co-NP? I.e., given $\vec{y} = (y_1, \ldots, y_n)$, is there a short proof that no polynomial of degree less than k can have $p(x_i) = y_i$ for all but t points?

(iii) If error t is very small, we can just pick k points (x_{i_j}, y_{i_j}), $j \in [k]$ at random. With high probability they will all be correct, and so we can do standard interpolation

But what about the general case?
We have many more points than are needed for interpolation, but some are wrong

Could pick all subsets of k points, interpolate, and check if polynomial is OK.
But $\binom{n}{k}$ subsets — too large for $k = \Theta(n)$

Nevertheless, this can be done!
First algorithm by Peterson, Gorenstein, & Zierler in 1960s
Time $O(n^3)$
Berlekamp and Massey $O(n^2)$ late 1960s
Currently best (?) algorithm by Justesen '76
in time $O(n \text{ polylog } n)$

We will give a simpler algorithm by Welch and Berlekamp that runs in time $O(n^3)$

Let's start by some wishful thinking
Wouldn't it be great to have an auxiliary polynomial that could zero out all errors?

DEF 1 Given vectors $\vec{x}, \vec{y}$ such that $\vec{y}$ is within distance t from polynomial p of degree $< k$, a polynomial $E(x)$ is called an ERROR-LOCATING POLYNOMIAL if E has degree t and satisfies $E(x_i) = 0$ if $p(x_i) \neq y_i$

Two observations

- such a polynomial always exists.
  Take $T \subseteq [n]$ containing all errors and let $\mathcal{E}(x) = \prod_{i \in T} (x - x_i)$

- If we had such a polynomial, we would be in great shape! Just pick k non-zero points for $\mathcal{E}$ and interpolate $\vec{x}, \vec{y}$ on these points

But why would such a polynomial be any easier to find than p in the first place? So are we making progress?

Let us not worry, but instead do more wishful thinking. If we have $p(x)$ and $\mathcal{E}(x)$, we can also define

$$N(x) = p(x)\,\mathcal{E}(x)$$

Also looks hard to find... But together, they are easier to find! Let us list some of the properties of these polynomials

(1) $\mathcal{E}(x_i) = 0$ if $y_i \neq p(x_i)$
(2) $\deg(\mathcal{E}) = t$
(3) $N(x) = p(x)\,\mathcal{E}(x)$
(4) $\deg(N) < k + t$
(5) $N(x_i) = p(x_i)\,\mathcal{E}(x_i) = y_i\,\mathcal{E}(x_i)$ for all $i \in [n]$

(1) - (4) immediate

(5) clearly holds when $p(x_i) = y_i$

But if $p(x_i) \neq y_i$ we have $E(x_i) = 0$ by (1), and so have equality in this case as well

Why is this helpful?

Turns out, we can ignore p in (1) - (5) and solve for N and E.

Then we use (3) to get p. Done!

We will now:

(a) Describe the algorithm more carefully

(b) Argue efficiency (relatively straightforward)

(c) Argue correctness (bit more subtle, but not hard)

WELCH - BERLEKAMP ALGORITHM

Input: n, k, $t \leq \frac{n-k}{2}$ $\left(= \frac{d-1}{2}$; best we can hope for$\right)$

n pairs $\{(x_i, y_i)\}_{i=1}^{n}$ with x_i's distinct

Step 1 Find polynomials N and E satisfying following conditions, if such polynomials exist

(2) $\deg(E) = t$

(4) $\deg(N) < k + t$

(5) $N(x_i) = y_i E(x_i)$ for all $i \in [n]$

Step 2 Output $N(x) / E(x)$

Efficiency

Step 2 is just polynomial division

Step 1 is solving a system of linear equations

Want to find $(E_0, E_1, \ldots, E_t)$ and $(N_0, N_1, \ldots, N_{k+t-1})$ such that for all i

$$\sum_{j=0}^{k+t-1} N_j x_i^j = y_i \sum_{j=0}^{t} E_j x_i^j \quad (6)$$

This is just a linear constraint in unknowns N_j, E_j

Also need $E_t \neq 0$ — not a linear constraint

But we can force $E_t = 1$ in (6)

For solutions N, E with $E_t \neq 1$, we could just divide by E_t to get another solution satisfying this.

We now have t free variables E_j

$k+t$ free variables N_j

$n \geq k+2t$ equations

We can certainly solve this by Gaussian elimination in time $O(n^3)$ to determine whether there is a solution or not and if so find it

Correctness

Assume there is a polynomial p, $\deg(p) < k$, s.t. $p(x_i) = y_i$ for all but at most t points [and recall such a p must be unique]

Want to argue

(1) There exist $N(x)$ and $E(x)$ as above such that in addition $p(x) = N(x)/E(x)$

(2) For any two solutions $(N(x), E(x))$ and $(N'(x), E'(x))$ it holds that $\frac{N(x)}{E(x)} = \frac{N'(x)}{E'(x)}$

In general can't expect solutions to be unique, but by (1) there is one solution where the quotient gives the right value and then by (2) any solution ~~but~~ must give the correct answer.

Proof of Claim 1

Pick $E(x)$ to be an error locating polynomial $E(x) = \prod_{i \in T} (x - x_i)$ for $T \subseteq [n]$ of size $|T| = t$ containing all errors

Set $N(x) = p(x) E(x)$

Then $N(x)$ and $E(x)$ satisfy (2), (4), and (5) and $N(x)/E(x) = p(x)$

Proof of Claim 2

We can equivalently prove that

$$N(x)E'(x) = N'(x)E(x) \qquad (7)$$

The polynomials in (7) have degree $< 2t + k$

From (5) we know that for all $i \in [n]$

$$y_i E(x_i) = N(x_i)$$
$$y_i E'(x_i) = N'(x_i)$$

and multiplying we get

$$y_i N(x_i) E'(x_i) = y_i N'(x_i) E(x_i)$$

If In fact it holds that

$$N(x_i) E'(x_i) = N'(x_i) E(x_i) \qquad (8)$$

for all $i \in [n]$. ~~By~~ Obviously one if $y_i \neq 0$. But what if $y_i = 0$? Well, then by (5) (again) we have $N(x_i) = N'(x_i) = 0$

This (8) means that $N(x)E'(x)$ and $N'(x)E(x)$ agree in n distinct points. But they are both of degree at most

$$k + 2t - 1 \leq k + 2\left(\frac{n-k}{2}\right) - 1 = n - 1.$$

Hence, by the Degree Mantra they are in fact equal.

X

Summarizing we obtain

THM 2 The Welch-Berlekamp algorithm solves the unambiguous Reed-Solomon decoding problem in time $O(n^3)$.

How can this be sped up?
Looking more closely, the algorithm needs to solve

(1) rational function interpolation problem

$$\frac{N(x_i)}{E(x_i)} = y_i$$

(2) polynomial division problem $P = N(x)/E(x)$

Both of these tasks can be solved in time $O(n \operatorname{polylog} n)$.

This is nice, but we are not yet satisfied

- RS codes have (very) large alphabets
  Would like to get asymptotically good codes over binary alphabet
- Would like to push (encoding and) decoding time all the way down to linear

We will conclude our excursion into coding theory by studying a family of asymptotically good codes with linear-time encoding and decoding

Recall

Family of codes $\mathcal{C} = \{C_i\}_{i \in \mathbb{N}^+}$

C_i $(n_i, k_i, d_i)_{q_i}$ - code $\lim_{i \to \infty} n_i = \infty$

Typically want $q_i = q$ fixed (ideally $= 2$)

(MESSAGE) RATE $R(\mathcal{C}) = \liminf_{i \to \infty} \frac{k_i}{n_i}$

(RELATIVE) DISTANCE $\delta(\mathcal{C}) = \liminf_{i \to \infty} \frac{d_i}{n_i}$

A family of codes $\mathcal{C}$ is ASYMPTOTICALLY GOOD if $R(\mathcal{C}) > 0$ and $\delta(\mathcal{C}) > 0$

$\mathcal{C}$ is EFFICIENT if encoding and decoding (with $\leq \delta n/2$ errors) can be done in time polynomial in n_i.

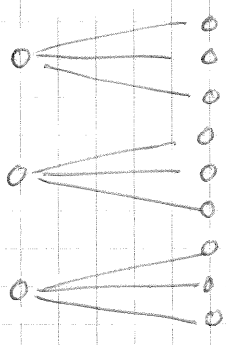
(We are looking for superefficient codes with linear-time algorithms)

These codes are built from expander graphs, but of the bipartite variety, and with extremely good expansion

DEF 3 A bipartite graph $G = (U \cup V, E)$ is a (D, s, ε)-LOSSLESS EXPANDER if the left degree is D and for all $U' \subseteq U$ of size $|U'| \leq s$ it holds that the set of neighbours on the right that

$$|N(U')| \geq (1 - \varepsilon) D |U'|$$

Note that the best possible expansion would be if every vertex on the left had D unique neighbours on the right:



This is not a terribly interesting graph, but it would have vertex expansion $D s$ for any $U' \subseteq U$ of size s.

A lossless expander is a much denser graph, with vertices on right of degree ≥ 1, but for which nevertheless the expansion is almost as good as this! Can such graphs even exist? Yes! ← (for moderately large vertex sets U')

THM [Capalbo, Reingold, Vadhan, Wigderson '02]
For any $\varepsilon > 0$ and $M \leq N$, there are explicit graphs $G = (U \cup V, E)$ with $|U| = N$, $|V| = M$, G has left degree D, and G is a $(D, \Omega(\varepsilon M / D), \varepsilon)$ - lossless expander

When M/N and ε are bounded from below by some constant, degree D can be chosen as a constant as well

Construction uses ideas related to the graph products we used in our expander construction from a few lectures back.

How do we get codes from bipartite graphs?

Given a linear code C (over $\mathbb{F}_2$, where we will stay for what remains of our coding theory survey) of blocklength n and message length k, there is a parity check matrix $H \in \mathbb{F}_2^{n \times (n-k)}$ such that

$$C = \{ x \mid xH = 0 \}$$

Can also represent this as bipartite graph $G = (U, V, E)$ with

$|U| = n$; label vertices $x_1, x_2, \ldots, x_n$ (positions in codeword)

$|V| = n-k =: m$; label by columns in H

Edge from $x_i \in U$ to $j \in V$ iff $h_{ij} \neq 0$

In this way, can construct bipartite graph from (parity check matrix of) code.

Can also start with graph G and build parity check matrix $\Rightarrow$ yields code $C(G)$

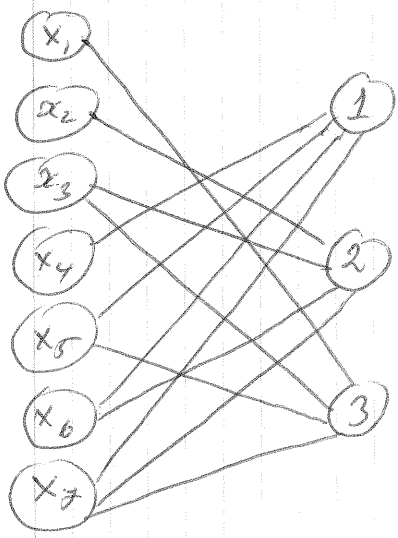
EXAMPLE $[7,4,3]_2$ Hamming code

Parity check matrix

$$H = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Corresponds to parity constraints

$$x_4 + x_5 + x_6 + x_7 = 0 \quad (1)$$
$$x_2 + x_3 + x_6 + x_7 = 0 \quad (2)$$
$$x_1 + x_3 + x_5 + x_7 = 0 \quad (3)$$



Corresponding bipartite graph G
(not regular on the left, BTW, but this is just an example)

If graph G is sparse $O(n)$ edges we get a LOW DENSITY PARITY CHECK CODE (or LDPC code, for short)

Some quick observations about LDPC codes:

- Can check if $x \in C$ in linear time (since parity constraints involve a total of $O(n)$ terms)
- If G is an expander, and x is close to codeword in C, then then most constraints on the left (connected to faulty x_i's) should have all but one x_i correct and should thus flag that something is wrong.
- Moreover two different errors x_{i^*} and x_{j^*} should be connected to different constraints and give different "error signatures"

With a little bit of luck, should be able to pinpoint errors and correct them quickly.

Turns out to be true — we will see this (and talk some more about LDPC codes) next time