COMPSCI 690T Coding Theory and Applications Feb 22, 2017

Lecture 8

Instructor: Arya Mazumdar *Scribe: Names Redacted*

1 Review

1.1 Error correcting codes

The parameters for an error correcting code are

n**:** length of the code (number of bits)

k**:** logarithm of code size

d**:** minimum (Hamming) distance between any two codewords

The rate of the code is $R = k/n$. Suppose we have a family of codes with $d = \delta n$, i.e. minimum distance scales linearly with length. We want to find the best possible value of

$$R(\delta) = \lim_{n\to\infty} \frac{k}{n}.$$

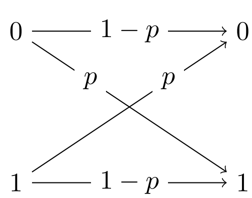
The Bassalygo-Elias bound and the Gilbert-Varshamov bound give us upper and lower bounds for $R(\delta)$:

$$\text{B-E bound} \geq R(\delta) \geq 1 - h(\delta)$$

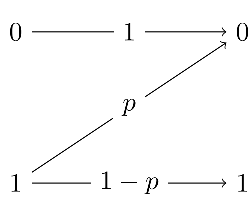
The G-V bound comes from the expected properties of a random linear code. Almost all such codes are "good", meaning achieves the G-V bound. However, verifying that a given linear code is good is computationally hard.

1.2 Communication channels

Binary symmetric channel: each bit independently can flip with probability p.



Z channel:



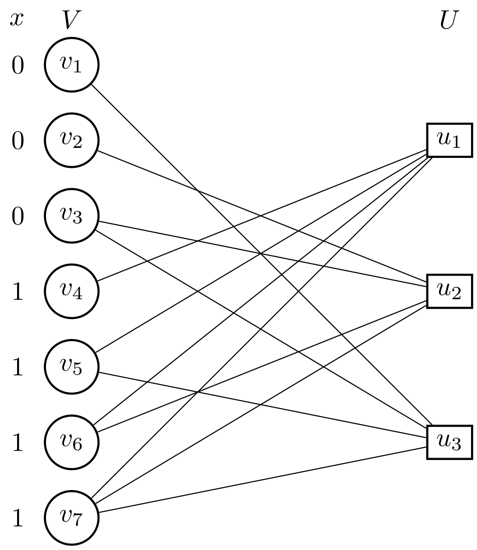
Definition 1 *The* ***capacity*** *of a channel is the best possible rate of a code in which errors from the channel can almost always be corrected.*

1.3 Decoding a linear code

A linear code C has an $(n-k)\times n$ parity check matrix like this:

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

We can decode using a **factor graph**. Make a bipartite graph with the left part containing **variable nodes** v_i representing the bits of the code and the right part containing **check nodes** u_j representing the rows of H. Draw an edge between v_i and u_j if the (j,i) entry of H is one.



To decode a vector x, we plug its components into the n variables nodes on the left side of the graph. Let $N(v)$ be the neighbors of the node v. We say that a check node u_j is **satisfied** if $\sum_{v\in N(u_j)} x_v \equiv 0 \pmod 2$. If all check nodes are satisfied, then $x \in C$.

To decode a vector x using the factor graph, plug its components into the variable nodes as shown above. Find a node v_i on the left with more unsatisfied neighbors than satisfied neighbors, and flip the component of x corresponding to this node. Repeat this step until all nodes are satisfied and output the resulting vector. (Call this **algorithm A**.)

We want to show that for a linear code C defined on a bipartite expander graph, this algorithm will terminate and correct a number of errors proportional to n.

2 Expander graphs

Definition 2 *A bipartite graph $G=(U,V,E)$ is a (Γ,A)-**expander** if for every subset $S\subset V$ of size $|S|\le\Gamma$, the set of neighbors $N(S)$ satisfies $|N(S)|>A|S|$.*

Suppose we form a random factor graph G by starting with vertex sets V and U. This gives us a linear code since the factor graph is equivalent to a parity check matrix. Let D be the left degree of the graph. With high probability there exist $\alpha\in(0,1)$ and $\varepsilon>0$ such that G is a $(\alpha n,(1-\varepsilon)D)$-expander. In order to prove that algorithm A works, we need an $(\alpha n,\frac{3}{4}D)$-expander graph.

Suppose $|S| = \alpha n$. Then there are a total of $\alpha n D$ edges touching S. The probability that it is not an $(\alpha n, (1-\varepsilon)D)$-expander is given by,

$$\begin{aligned} P_e &\le \binom{n}{\alpha n}\binom{\alpha n D}{\varepsilon \alpha n D}\left(\frac{\alpha n D}{|U|}\right)^{\varepsilon \alpha n D} \\ &\le \left(\frac{e}{\alpha}\right)^{\alpha n}\left(\frac{1}{\varepsilon}\right)^{\varepsilon \alpha n D}\left(\frac{\alpha D}{1-R}\right)^{\varepsilon \alpha n d} \\ &= \left[\frac{e}{\alpha}\left(\frac{1}{\varepsilon}\frac{\alpha D}{1-R}\right)^{\varepsilon D}\right]^{\alpha n}. \end{aligned}$$

(Note that $|U| = n - k = (1-R)n$.) We just have to choose α small enough to make the quantity inside the square brackets less than 1.

Theorem 3 (Sipser and Spielman) *If number of errors $\le \frac{\alpha n}{2}$, then algorithm A finds the correct codeword (and converges).*

Proof From the previous claim, we can assume the factor graph is a $(\alpha n, (1-\varepsilon)D)$-expander for some α and ε. Let v be the number of corrupt variables and let u be the number of unsatisfied vertices on the right. The state of the algorithm is (v, u). Let S be the set of corrupted variables and let s be the number of satisfied neighbors of corrupted variables.

If $|S| \le \alpha n$, then $|N(S)| \ge v\frac{3}{4}D$ because the graph is an expander. Therefore we have $u + s > \frac{3}{4}Dv$. Furthermore, $Dv \ge u + 2s$. Putting these facts together, we obtain

$$\begin{aligned} s &\ge \frac{3}{4}Dv - u \\ Dv &\ge u + \frac{3}{2}Dv - 2u \\ u &> \frac{Dv}{2}. \end{aligned}$$

Therefore at least one corrupt variable node is connected to at least $D/2$ unsatisfied variables. So the algorithm will continue as long as $v \le \alpha n$.

If at any stage we have $v > \alpha n$ then the algorithm fails. In order for this to happen, the state of the algorithm must first pass through $v = \alpha n$. If $v = \alpha n$, then $|N(S)| \ge \frac{3}{4}\alpha n$ which implies

$$\begin{aligned} u + s &> \frac{3}{4}\alpha n \\ Dv &\ge u + 2s \\ u &> \frac{Dv}{2} = \frac{\alpha n D}{2}. \end{aligned}$$

Therefore the state is (u, v) where $u > \alpha n D/2$ and $v = \alpha n$. But since at the start of the algorithm $u \le \alpha n D/2$, as at that stage the number of corrupted variables is $\alpha n/2$, and u can only decrease through the execution of the algorithm, we have a contradiction. ■

This is the first evidence in this class we have seen of codes that efficiently correct a number of errors linear in n.