COMPSCI 690T Coding Theory and Applications March 1, 2017

Lecture 10

Instructor: Arya Mazumdar *Scribe: Names Redacted*

1 Expander Codes

1.1 Review

Definition Given a D-regular bipartite graph G with the two parts being U and V and $|U| = |V| = m$. We pick $2m$ different local codes C_0 that has parameters (D, R_0D, δ_0D) for each vertex. For each vertex in G, since it has a degree of D, if we assign a bit to each edge coming out, we could get a D bit sequence that either belongs to or does not belong to its assigned local code. We can now construct an **Expander Code** that has code length $n = mD$, where each bit corresponds to an edge in G, and its codewords are all those assignments that ensure that the for every vertex in G the sequence from its edges is a valid codeword.

Lemma (Expander Mixing Lemma) Given a expander code with graph G, for any vertex sets S, T such that $S \subseteq V, T \subseteq U$

$$||E(S,T)| - \frac{D|S||T|}{m}| \leq \lambda\sqrt{|S||T|}$$

where λ is the 2nd largest eigenvalue of the adjacency matrix of G.

Theorem 1 (Zémor 2001) For an expander code C with length n constructed from a D-regular bipartite graph and local codes with parameters (D, R_0D, δ_0D), the rate of the code $R \geq 2R_0 - 1$ and the minimum distance $d_{min} \geq \delta_0(\delta_0 - \frac{\lambda}{D})n$.

Proof Assume that there is a codeword in C that has weight ωn, and we look at the vertex set $S \subseteq V$ that contains all the vertices with weight more than 1 on the left side and the vertex set $T \subseteq U$ that contains all the vertices with weight more than 1 on the right side. Since the minimum weight for the local codes is $\delta_0 D$, we have

$$|S| \leq \frac{\omega n}{\delta_0 n} \qquad |T| \leq \frac{\omega n}{\delta_0 D}$$

We can bound $E(S,T)$ on the left by ωn since the edges between S and T must contain all the one's and we can bound it on the right with the expander mixing lemma.

$$\omega n \leq |E(S,T)| \leq \frac{\omega n}{\delta_0 n}\frac{\omega n}{\delta_0 n}\frac{D^2}{n} + \lambda\frac{\omega n}{\delta_0 D}$$
$$1 \leq \frac{\omega}{\delta_0^2} + \frac{\lambda}{\delta_0 D} \Longrightarrow \omega \geq \delta_0(\delta_0 - \frac{\lambda}{D}) \Longrightarrow d_{min} \geq \delta_0(\delta_0 - \frac{\lambda}{D})n$$

Now, since for each local code to satisfy, we need to satisfy $D(1 - R_0)$ linear equations and we have $2\frac{n}{D}$ local codes to satisfy, the dimension of code C is at least $n - 2\frac{n}{D}D(1 - R_0)$ and therefore $R \geq 2R_0 - 1$ ■

1.2 Error Correction for Expander Codes

Algorithm (Zemor 2001, Indyk and Guruswami 2001-2005) For an expander code with graph G with the two parts being U and V, correct errors for all local codes in U by nearest neighbor, and then correct errors for all local codes in V by nearest neighbor, continue such iterations in an alternating fashion.

Theorem 2 As long as the number of errors $|E| \leq (1-\epsilon)\frac{\delta_0}{2}(\frac{\delta_0}{2} - \frac{\lambda}{D})n, \forall \epsilon > 0$, the algorithm will converge and correct these errors (takes $\frac{1}{\epsilon}$ iterations).

Proof Given the conditions that we set, we know that:

$$|E_0| \leq (1-\epsilon)\frac{\delta_0}{2}(\frac{\delta_0}{2} - \frac{\lambda}{D})n$$

where $|E_0|$ is the number of errors before any iterations of the algorithm.

Now, let S_1 be the set of vertices on the left that make mistakes in the first iteration of corrections on the left. Similarly, let T_1 be the set of vertices on the right that make mistakes in the first iterations of corrections on the right.

Since the local codes have minimum distance $\delta_0 D$ and can correct about $\frac{\delta_0 D}{2}$ errors, any correction will be wrong only if that vertex vertex contains more than $\frac{\delta_0}{2}$ errors, so we know:

$$|S_1| \leq \frac{|E_0|}{\frac{\delta_0}{2}} \leq (1-\epsilon)\frac{n}{D}(\frac{\delta_0}{2} - \frac{\lambda}{D})$$

By similar logic, every vertex of T_1 is connected to at least $\frac{\delta_0 D}{2}$ vertices of S. Hence:

$$\begin{aligned}
|T_1| &\leq \frac{2}{\delta_0 D}|E(S_1, T_1)| \\
&\leq \frac{2}{\delta_0 D}(\frac{|S_1||T_1|D^2}{n} + \lambda\frac{|S_1| + |T_1|}{2}) \\
&= |T_1|(\frac{2}{\delta_0 D}\frac{D^2}{n}|S_1| + \frac{\lambda}{\delta_0 D}) + \frac{\lambda}{2}\frac{2}{\delta_0 D}|S_1| \\
&\leq |T1|(\frac{2D}{\delta_0 n}(1-\epsilon)\frac{n}{D}(\frac{\delta_0}{2} - \frac{\lambda}{D}) + \frac{\lambda}{\delta_0 D}) + \frac{\lambda}{\delta_0 D}|S_1| \\
&\leq \frac{\frac{\lambda}{\delta_0 D}|S_1|}{1 - \frac{2D}{\delta_0 n}(1-\epsilon)\frac{n}{d}(\frac{\delta_0}{2} - \frac{\lambda}{D}) - \frac{\lambda}{\delta_0 D}} \\
&\leq \frac{\frac{\lambda}{\delta_0 D}}{1 - \frac{2}{\delta_0}(1-\epsilon)(\frac{\delta_0}{2} - \frac{\lambda}{D}) - \frac{\lambda}{\delta_0 D}}|S_1|
\end{aligned}$$

If we can show $|T_1| \leq \alpha|S_1|$ where $\alpha < 1$, then we have finished the proof because we show that the number of errors strictly decrease after each iteration. For this to be true, we must have:

$$\begin{aligned}
\frac{\lambda}{D} &< 1 - \frac{2}{\delta_0}(1-\epsilon)(\frac{\delta_0}{2} - \frac{\lambda}{D}) - \frac{\lambda}{\delta_0 D} \\
\frac{2\lambda}{\delta_0 D} &< \epsilon + \frac{2\lambda}{\delta_0 D} - \epsilon\frac{2\lambda}{\delta_0 D} \\
0 &< \epsilon(1 - \frac{2\lambda}{\delta_0 D})
\end{aligned}$$

And we can satisfy this by ensuring that $\delta_0 D > 2\lambda$ ■

2 Evaluation Codes

$v_1, v_2, ...v_m$ are boolean variables, so they are either 0 or 1, so when speaking of boolean polynomials, v_i^n is simply v_i. So binary polynomials must be multinomials. ex: $1 + v_1v_2 + v_3v_4v_5 + ...$ As an example,

let $f(v_1, v_2, v_3) = 1 + v_1v_2 + v_1v_2v_3$. Since there are 3 boolean variables, there are 8 possible inputs to the function. The results shown in the table below.

v_1	v_2	v_3	f
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

2.1 Reed-Muller Code (1954)

A (r, m)-Reed-Muller code is defined by all polynomials of degree $\leq r$ consisting of m variables, $v_1, v_2, v_3, ...v_m$
Looking at $RM(1, 3)$, so $r = 1$ and $m = 3$, we have variables, v_1, v_2, and v_3, and polynomials,

degree 0	0	1					
degree 1	v_1	v_2	v_3	$1 + v_1$	$1 + v_2$	$1 + v_3$	$v_1 + v_2$
	$v_1 + v_3$	$v_2 + v_3$	$v_1 + v_2 + v_3$	$1 + v_1 + v_2$	$1 + v_1 + v_3$	$1 + v_2 + v_3$	$1 + v_1 + v_2 + v_3$

For each of the aforementioned polynomials, we can evaluate them for all possible inputs, in the same order we did for the example function above, and those will be our codewords. So in this case we have 16 total codewords each corresponding to a unique polynomial.
In the table below we show the generation of the first 8 codewords, and the other 8 codewords would be found by evaluating the other 8 polynomials.

v_1	v_2	v_3	$f = 0$	$f = 1$	$f = v_1$	$f = v_2$	$f = v_3$	$f = 1 + v_1$	$f = 1 + v_2$	$f = 1 + v_3$
0	0	0	0	1	0	0	0	1	1	1
0	0	1	0	1	0	0	1	1	1	0
0	1	0	0	1	0	1	0	1	0	1
0	1	1	0	1	0	1	1	1	0	0
1	0	0	0	1	1	0	0	0	1	1
1	0	1	0	1	1	0	1	0	1	0
1	1	0	0	1	1	1	0	0	0	1
1	1	1	0	1	1	1	1	0	0	0

Code description For $RM(r, m)$, the length of this code is the total number of binary combinations of the m variables, so $n = 2^m$. The number of codewords, or 2^k, is just the number of multinomials consisting of m variables with degree $\leq r$.

Claim The number of multinomials using m binary variables and of max degree r is $2^{\sum_{i=0}^{r} \binom{m}{i}}$
Proof The number of multinomials using m binary variables and of max degree r is simply the product of the number of all multinomials with degrees from 0 to r, inclusive of the all 0 vector in each degree, since not all degrees must be present in each multinomial. So to get the number of x^{th} degree terms that exist when we have m variables, we simply have choose each combination of x variables from m, so $\binom{m}{x}$ terms exist, but since we can take any binary linear combination of these terms and still end with an expression of only x^{th} degree terms we have $2^{\binom{m}{x}}$. Therefore our total number of multinomials is $\prod_{i=0}^{r} 2^{\binom{m}{i}}$, which is just equivalently $2^{\sum_{i=0}^{r} \binom{m}{i}}$ ■

Since the number of multinomials is 2^k, k is just $\sum_{i=0}^{r} \binom{m}{i}$.

Now to finish desribing the code we must find d_{min}. To try to build up an intuition of this d_{min}, lets analyze it for $RM(1,3)$. Since this is a linear code, $d_{min} = \min_{\forall c \in C}\{wt(c)\}$. Additionally in this case, for all codes with a 1 in the first element there exists the same code with a 0, so for determining minimum weight, we can ignore that first element entirely, which in turn is equivalently ignoring the all 0 input. Now all multinomials of degree 1 or less are just linear combinations of the binary variables, so that means our generated code list (ignoring the 0 input) can simply be generated by using the $H^{(3)}$ parity check matrix as our generator matrix, which is just the dual of the hamming-3 code, where we have already done the analysis and arrived that is has d_{min} of 4. Using this same methodology, for any Reed-Muller code with $r = 1$, and any m, can be seen as the dual -code of a hamming code, so they have $d_{min} = 2^{m-1}$ which is equivalent to 2^{m-r} in this case. Which turns out, the expression 2^{m-r} can be generalized for any Reed-Muller code $RM(r,m)$ (also, see the $(u, u+v)$-construction from Assignment 1).

So the code description of $RM(r,m)$ is $[2^m, \sum_{i=0}^{r} \binom{m}{i}, 2^{m-r}]$

2.2 Reed Solomon Code

Reed Solomon codes use univariate polynomials, such as: $f(x) = 1 + x + x^2 + x^3 + x^4$. We will discuss this topic further in the next lecture.