1 Administration

- fill out the zoom questionnaire;
- PS3 due soon, let Madhu know if there are troubles finishing on time;
- pass/fail grading might be possible

2 Linear-time encodable & decodable codes

Last time we discussed a graph-theoretic code with a linear-time decoding algorithm but for which a linear-time encoding algorithm is unknown. Today we discuss another graph-theoretic code for which we know a linear-time algorithm for both encoding and decoding.

All codes we discuss today are said to be systematic which means they include the original message itself: $E(m) = (m, x)$.

2.1 Code $R_k$

As in the previous lecture, consider a bipartite graph with left vertices $L = \{l_1, ..., l_k\}$ and right vertices $R = \{r_1, ..., r_k\}$ and edges $E$ that is $(c, 2c)$-regular and $(\gamma, \delta)$-expander. Thinking of it as a sequence of expanders for infinitely many $k$, here $c$ is bounded by a constant and $\gamma, \delta$ are constants. Define the encoding function $E: F_2^k \rightarrow F_3^{k/2}$ by $E(m) = mx_1...x_{k/2}$ where $x_j = \bigoplus_{\{i,j\} \in E} m_i$.

This code will prove to be useful but not in its current form, since its relative distance goes to 0. Indeed, $E(0^k) = 0^{3k/2}$ and $E(10^{k-1})$ has exactly $c+1$ ones, so the distance is at most $c+1$.

2.2 $R_k$ is an error-reduction code

Despite its uselessness as it is, $R_k$ possesses the error-reduction property as defined below.

**Definition 1.** $E: m \mapsto (m, x)$ is an $\varepsilon$-error reduction code if there exists a decoder $D: (\hat{m}, \hat{x}) \mapsto \tilde{m}$ such that if $\delta(m, \hat{m}) \leq \varepsilon$ and $\delta(x, \hat{x}) \leq \varepsilon$, then $\Delta(m, \tilde{m}) \leq \Delta(x, \hat{x})/2$.

Here, $\delta(\cdot, \cdot)$ is the relative distance of two bit-strings: if $x, y \in F_2^r$, then $\delta(x, y) = \Delta(x, y)/r$.

In particular the following two properties hold for an $\varepsilon$-error reduction code.

- if $\delta(m, \hat{m}) \leq \varepsilon$ and $\delta(x, \hat{x}) = 0$, we get perfect error-correction: $\delta(m, \tilde{m}) = 0$;
- if $\Delta(m, \hat{m}) \leq \varepsilon k$ and $\Delta(x, \hat{x}) \leq \varepsilon^{k/2}$, we get a factor of 2 error-reduction: $\Delta(m, \tilde{m}) \leq \varepsilon^{k/4}$.

Note that we are using both $\delta(\cdot, \cdot)$ and $\Delta(\cdot, \cdot)$ in the definition. The reason we are using $\Delta(\cdot, \cdot)$ for the conclusion is that the code construction in the next subsection relies on the fact that we can reduce the absolute number of errors.

**Theorem 2.** Assuming $c \geq 8$ and $\gamma \geq \frac{7}{8}$, there exists $\varepsilon > 0$ such that $R_k$ is an $\varepsilon$-error reducing code.
Proof. We use the FLIP algorithm from the previous lecture to partially recover \( m \).

We know that FLIP eventually terminates since each iteration decreases the number of unsatisfied vertices. We will also argue that the algorithm continues whenever the current message \( m' \) satisfies \( \Delta(m, m') > \frac{1}{2} \Delta(x, \hat{x}) \). Given these two assertions, we can conclude that the algorithm outputs \( \tilde{m} \) such that \( \Delta(m, \tilde{m}) \leq \frac{1}{2} \Delta(x, \hat{x}) \).

Assume \( \Delta(m, m') > \frac{1}{2} \Delta(x, \hat{x}) \). Let \( S = \{ l_i : m_i \neq m_i' \} \) be the set of left vertices where \( m \) and \( m' \) disagree. Let \( U = \Gamma_{\text{unique}}(S) \), the set of right vertices that have exactly one neighbor in \( S \), and let \( V = \Gamma(S) \setminus U \).

From the previous lecture we know that \( |U| \geq (2\gamma - 1)c|S| \) if \( |S| \leq \delta k \).

Also define \( T = \{ r_j : x_j \neq \hat{x}_j \} \). We know \( |T| = \Delta(x, \hat{x}) \).

**Exercise 3.** Argue that the number of unsatisfied neighbors of \( S \) found in \( U \setminus T \) exceeds \( \frac{c}{4} |S| \), and conclude that there must be a vertex in \( S \) that can be flipped.

**Exercise 4.** Find \( \varepsilon \) that ensures \( |S| \leq \delta k \) in every iteration of the algorithm.

---

2.3 Spielman code

We now apply the error-reduction property of \( R_k \) to construct a new code that corrects a constant number of errors.

Define \( S_k : \mathbb{F}_2^k \to \mathbb{F}_4^{4k} \) recursively by \( S_k(m) = (m, x_1, x_2, x_3) \) where

- \( mx_1 = R_k(m) \),
- \( x_1x_2 = S_{k/2}(x_1) \),
- \( x_1x_2x_3 = R_{2k}(x_1x_2) \),

and we can let \( S_1(b) = bbbb \).

Having the inductive hypothesis that \( S_{k/2} \) corrects \( \varepsilon \frac{k}{2} \) errors, we can show that \( S_k \) corrects \( \varepsilon k \) errors. Indeed, if \( \Delta(\tilde{m}x_1\tilde{x}_2\tilde{x}_3, mx_1x_2x_3) \leq \varepsilon k \), we know that \( \Delta(\tilde{m}, m) \leq \varepsilon k \), \( \Delta(\tilde{x}_1\tilde{x}_2, x_1x_2) \leq \varepsilon k \), \( \Delta(\tilde{x}_3, x_3) \leq \varepsilon k \).

Reversing the \( R_{2k} \) function, we can partially recover \( x_1x_2 \), with up to \( \varepsilon \frac{k}{2} \) errors. Having that, we can recover \( x_1 \) fully by reversing \( S_{k/2} \). Finally, with the original \( x_1 \) we can fully recover \( m \).

Thus, \( S_k \) corrects \( \varepsilon k \) errors.

Finally, note that both encoding and decoding algorithms are linear-time in \( k \).