Today

- Linear time list-decodable codes.
  - (Yet another family of) Expander-based codes.
  - A "simple" decoding algorithm.
  - Towards analysis of list-decodability.
  - Best known results

- Achnowledgments: Thanks to Piotr Indyk for slides and Amir Shpilka for the lecture!

Decoding with adversarial error

- Best known results
  - RS codes of rate ϵ^2 can (list-)decode $1-\epsilon$ fraction error, but take super-linear time.
  - Can construct binary codes of rate ϵ^4 decoding $1/2 - \epsilon$ error, again in super-linear time.
  - But what about linear-time.
  - Requires simpler coding/decoding schemes.

ABNNR Codes

- Alon-Bruck-Naor-Naor-Roth '93.
- Yet another family of expander based codes.
- New elegant idea - using expansion of large sets.
- Only known construction (to Madhu) going from codes over small alphabets to codes over large alphabets. An important direction!

Code

- Ingredients:
  - Asymptotically good $[n, k, \delta n]$ binary code A.
  - (c, c)-regular (γ, δ)-weak bipartite expander G with n vertices on each side: Every set of size δn has at least $\gamma\delta n$ neighbors (no requirements on smaller sets).
- Gives: (non-linear) code over $q = 2^c$-ary alphabet, with message length k/c message and n block length and minimum distance $\gamma\delta n$.
- Construction: given message m ($= k$-bit string), encode using A first to get $c = n$

bit string. Then label left vertices with bits of c. For each right vertex now write the label of all c right neighbors (in canonical order). The labels of right vertices form a q-ary string of length n. This is the encoding of m.

Properties

- Rate $= k/(cn)$.
- Distance $= \gamma\delta n$.
- Alphabet $= 2^c$.
- How to make sense?
  - Will fix $k/n = frac14$, say.
  - Fixes $\delta = \Omega(1)$.
  - Remaining parameter c. Study behaviour of code as c grows.
  - Rate $= O(1/c)$, Alphabet size $= 2^c$. Main issue: How does distance behave?
  - Clearly $\gamma \leq c$. But we'll take $c \gg \frac{1}{\delta}$. In such case, clearly $\gamma\delta \leq 1$! Actually

it is worse: Can get $\gamma\delta \leq 1 - O(1/c)$. (Why we can't do better? Take a set of size $n/(2c)$ on right. Has at most $n/2$ neighbors on left. So exists sets of size $n/2$ on left with at most $n(1 - 1/(2c))$ neighbors on right. For random c-regular graphs - this is about right.)
  - Moral of the story: Can get q-ary codes of rate $\Omega(\epsilon)$ of rel. distance $1-\epsilon$ over $2^{O(\epsilon)}$-sized alphabet.
- Compares decently with q-ary random codes. (Similar relationship between alphabet size and distance).
- Not as good as AG codes.
- But good enough for concatenation. Also gives binary codes of rate $O(\epsilon^3)$ with rel.

distance $\frac{1}{2} - \epsilon$.

- Major novelty (partly in hindsight): Leads to linear time encoding and linear time list-decodability.
- Encoding obvious. Decoding needs more from graph.

Decoding algorithm

- Given set of assignments for right vertices.
- Will compute assignment to left vertices.
- Obvious idea: Write most popular vote for each vertex.
- Then decode left hand side.

Additional assumptions

- Code A is linear-time decodable.
- Graph is a strong weak expander - call it mixer!
- Will want: For every subset T of size $(frac12 + \epsilon)n$ on right, the set of vertices that have fewer than $c/2$ neighbors into T, is at most δn.
- Note: random vertex has most neighbors into T.
- Random graph is a $\delta, \frac{1}{2} + \epsilon$ mixer.

Expanders, Mixers, Extractors, Refrigerators

- Lots of notions of expansion.
- Should be thought of as a generic notion, not specific (α, β)-property.
- Most supposed to be some notion of "pseudorandom graph".
- E.g., Extractors: For every large enough left subset S, random neighbor of random element of S is almost uniform left neighbor.
- Mixing is a similar property.

Recent Results

- Guruswami-Indyk Constructions (a factory).
- Construction of list-decodable expander-based codes:
  - Decoding radius: $(1 - \delta)n$
  - Constant rate $r(\delta)$
  - Linear-time encoding/decoding
  - Constant alphabet
- In a sense, unifies the results of Spielman and Guruswami-Sudan
- Departs from the current list-decoding technology
  - Combinatorial construction
  - No polynomials

G-I Results ctd.

- Rate $r = 1/2^{2^{1/\delta^2}}$, i.e., pretty low
- However, for simple list decoding scenarios, matches the rate of RS

Central theme: List-Recoverable codes

A code $C \subset \Sigma^n$ is (α, l, L)-recoverable, if for any

$$\mathcal{L} = L_1 \dots L_n,\ L_i \subset \Sigma,\ |L_i| \leq l,$$

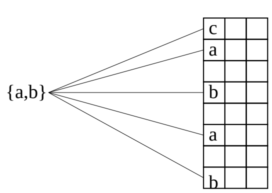
there are at most L codewords $c \in C$ such that

$$c_i \in L_i \text{ for } \geq \alpha n \text{ coordinates } i$$

- $l = 1$: list decodability
- Algorithmic version defined analogously

List Rec. $\Rightarrow$ List Dec.

- Assume we have a $(1-\epsilon, l, L)$-recoverable code C
- Take a graph $G = (A, B, E)$ such that for any $Y \subset B$, $Y| \geq |B|(\frac{1}{l+1} + \epsilon')$, the fraction of $i \in A$ for which *Neighbors*$(i) \cap Y| > \frac{D}{l+1}$ is $ge1 - \epsilon$
- Then $G(C)$ is list-decodable from $1 - (\frac{1}{l+1} + \epsilon')$ fraction of errors:



- The i-th left node creates the list L_i of l most frequent symbols
- Apply the list-recovering procedure

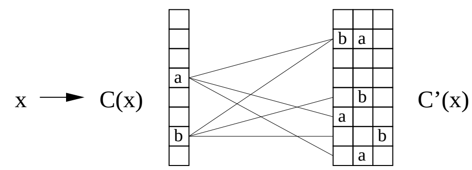
- $D = (1/\epsilon + 1/\epsilon' + l)^c$ suffices (Ramanujan graphs)

Goal: $(1-\epsilon, l, L)$-Recoverable Codes

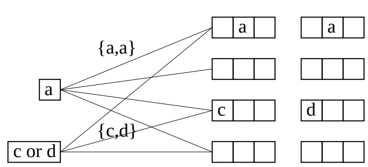
- Will give a $(1,2,2)$-list recoverable, linear-time code
- Indicate how to:
  - Handle errors
  - Allow $l > 2$

$(1,2,2)$-Recoverable Codes: Construction

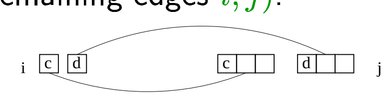
- Take any code C that:
  - Can be encoded in linear time
  - Can be decoded from, say, 90% of erasures in linear time
- Take a good expander $G = (A, B, E)$
- $C' = G(C)$



$1,2,2)$-Recoverable Codes: Decoding



- If many left-node symbols determined:
  - Replace rest by erasures
  - Decode the left-code C from erasures
- If few left-node symbols determined:
  - Remove left nodes with determined symbols
  - For all remaining edges $i, j)$:



- Find large connected component
- Use it to determine the left symbols
- Decode the left code from erasures

$(1-\epsilon, 2, 2)$-Recoverable Codes

- There will be erroneous edges between connected components
- Need to find large components with few outgoing edges
- Spectral partitioning:
  - Find the eigenvectors of the adjacency matrix
  - Use them to partition the graph
- $O(n \log n)$ time, can reduce to $O(n)$ by one level of concatenation

$l > 2$

- $O(l)$ layers of expander graphs
- Similar ideas, more messy details

Conclusions

- Can decode from 99% of errors in linear time
- Questions:
  - Can we improve the rate while preserving linear time ?
  - Can we beat RS rate while preserving polynomial time ?