Local list decoding of the Hadamard code
(Goldreich-Levin Theorem)

Given code $C \subseteq \Sigma^n$ of min dist $d$.
and a received word $r \in \Sigma^n$ s.t.
$r$ has $\leq e$ errors, where $e < d/2$

Goal: output unique $c \in C$ s.t.

$$\text{dist}(r, c) \leq e$$
Global list decoding.

- Given received $r \in \Sigma^n$ and error bound $e$ with $\frac{d}{2} \leq e$

- Goal: output list of all codewords $c_1, c_2, \ldots, c_L \in C \subseteq \Sigma^n$ s.t. $\text{dist}(r, c_i) \leq e$.

Goal: efficient list decoding.
Def: $C$ is $(e, L)$ list decodable if given received $r \in \Sigma^n$ and error bound $e$

1. **combinatorial:** # of codewords of $C$ within $e$ errors from $r$ is $\leq L$.

2. **algorithmic:** can output the list in $\circ 1$ efficiently in $n$ & $L$.

List decoding was first defined by Sudan who shows that $\text{RSC}(k, n)$ of dist $d = n - k$

is list decodable from

$$\frac{d}{2} < e < n - \sqrt{2n(n-d)}$$

Improved by Guruswami - Sudan to

$$e < n - \sqrt{n(n-d)} \rightarrow \text{Johnson bound}$$

(Technique: generalization of Berlekamp-Welch)
Interesting questions

• list decoding radius of a code (largest $\epsilon$ for which list is poly size)

• tradeoffs for list size & params of code

Obs: Many natural forms of codes are list decodable.

Some generalizations

• List learning

• List decodable lattices

• List decodable group homomorphisms.

Above is also explicit: each codeword is explicitly specified.
Implicit local list decoding.

Def: A probabilistically $M$ $\tau$-computes a function $f$ if $x$ in the domain

$$\Pr[ M(x) = f(x) ] \geq \tau$$

Def: Local list decoding (implicitly)

A probabilistic oracle $A$ is $(e, L)$-local list $f$ decodable if $x$ if $f$ $A$-distance $(f(x), C) < e$

if, when $A$ is given oracle access to $f$.

• $A^*$ outputs a list of probabilistic oracle machines $M_1, M_2, \ldots, M_L$ such that $A \in C$ with $\text{dist}(f(x), C) < e$

wp $\geq \frac{1}{2}$ over randomness of $A^*$, $f$ machine

$M_j$ that $\frac{1}{8}$ computes $c$.

• $A \times A^*$ run in time $\text{poly}(\log n, L)$

Ideally: $\text{poly}(\log n)$
We'll show Goldreich Levin's theorem: local list decoding of Hadamard code

\[ \text{N = 2}^k \]

\[ \text{Had} = \sum \text{la}(x) = \sum a_i x_i \mod 2 \]

\[ \text{la} : \{0, 1\}^k \rightarrow \{0, 1\} \]

\[ a \in \{0, 1\}^k \]

Goldreich–Levin: If probabilistic alg \( A \) st. relative

given \( f : \{0, 1\}^k \rightarrow \{0, 1\} \) with \( \text{dist}(f, \text{Had}) > \frac{1}{2} - \frac{3}{8} \)

runs in time \( \text{poly}(\frac{1}{\epsilon}, k) \), outputs a list \( \mathcal{L} \) st.

if \( \text{dist}(f, \text{la}) \leq \frac{1}{2} - \frac{3}{8} - \epsilon \) then

\( \text{la} \in \mathcal{L} \) w.p. \( > \frac{1}{2} \)

(explicit: \( a \) is output)

Obs: \( \text{dist}(\text{Had}) = \min \text{dist}(\text{la}, \text{lb}) = \frac{1}{2} \)

So unique decoding is at \( \epsilon < \frac{1}{4} \).
we can learn explicitly for Had. why (as we'll see soon)
Simple case: \( e < \frac{1}{8} < \frac{1}{4} \) (unique decoding)

Claim: If \( e \leq \frac{1}{8} \) then \( \mathbb{K} \log k \log \frac{1}{e} \) alg that outputs a \( \text{sd} \) \( \text{dist}(f, la) \leq \frac{1}{8} \) wp-\( \delta \).

Proof: Unique decoding:
For each bit \( i \in [k] \)
for \( j = 1 \) to \( t \)
pick \( x \in \{0, 1\}^k \) u.a.r.
compute \( a_{ij} = f(x+e_i) - f(x) \)
Output \( a_i = \text{maj } a_{ij} \)
i.e. output \( a = (a_1, a_2, \ldots, a_k) \in \{0, 1\}^k \)

Remark: \( l_a(x+e_i) - l_a(x) = a_i \)

Proof:
For \( j: a_{ij} = a_i \) wp \( 1- \frac{1}{4} = 3/4 \left( \frac{Pr[f(x+e_i) \neq l_a(x+e_i)]}{8} \right) \)
since \( x+e_i \) is u.a.r.
if \( x \) is u.a.r.
Hi

Pr[ maj of $a_{ij}$ is incorrect] $\leq e^{-\Omega(t)}$

(Chernoff bound)

Pr[ all a_i's are correct] $\geq (1 - e^{-\Omega(t)}) > 1 - \delta$

$t = \Theta(\log k, 1/\delta)$

Overall for all k bits of a we need $\Theta(k \log k)$ time.

(assume $\delta = \frac{1}{100}$)

So we output the correct a w.p. 99%

Here: the machine $M$ output is

st. $M(x) = l_a(x)$ if $x$. 
Case: large distance but with assumptions

Claim: If dist(la, f) < \( \frac{1}{2} - \varepsilon \) then \( \exists \) alg.

st. w.p. > \( \frac{1}{2} \) outputs \( M_a \) st. \( \Pr[ M(2) = la(2)] > \frac{3}{4} \)

in time \( O(\frac{1}{\varepsilon^2} k \log k) \)

Proof: Given oracle access to \( f \),

\( \text{Alg: } \) Pick \( x^{(1)}, x^{(2)}, \ldots, x^{(s)} \in \{0,1\}^k \) i.a.r.

(Assume can) Obtain \( la(x^{(1)}), la(x^{(2)}), \ldots, la(x^{(s)}) \)

Define func \( A \)

\[ A(x^{(1)}, x^{(2)} \ldots, x^{(s)}) = \text{maj} \left[ f(z + x^{(1)}) - la(x^{(1)}) \right] \]

We'll show: \( A \) is \( \frac{1}{8} \) close to \( la \) each diff. supposed to be equal to \( la(2) \) don't actually know these values
Analysis: Define random variables $E_1, \ldots, E_s$.

$E_i = \begin{cases} 1 & \text{when } f(z+x^{(i)}) = l_a(z+x^{(i)}) \\ 0 & \text{otherwise} \end{cases}$

Note: $E[\sum_{j} E_j] = \Pr \left( \bigcap_{x^{(1)}, x^{(2)}, \ldots, x^{(s)} \in \mathbb{X}} f(z + x^{(1)}) = l_a(z + x^{(1)}) \bigcup \ldots \bigcup f(z + x^{(s)}) = l_a(z + x^{(s)}) \right)$

by assumption of $f$ and $l_a$

We'll bound $\Pr \left( \bigcup_{x^{(1)}, x^{(2)}, \ldots, x^{(s)} \in \mathbb{X}} f(z + x^{(1)}) = l_a(z + x^{(1)}) \bigcup \ldots \bigcup f(z + x^{(s)}) = l_a(z + x^{(s)}) \right) = \Pr \left( \sum_{j} E_j < \frac{s}{2} \right)$

$\leq \Pr \left( \bigcup_{j} \left| \sum_{j} E_j - E[\sum_{j} E_j] \right| \leq 3s \right)$

$\approx (\frac{1}{2} + \varepsilon)s$

By Chebyshev's inequality:

$\Pr \left( \sum_{j} E_j > \frac{s}{2} \right) \leq \frac{\operatorname{Var}\left[\sum_{j} E_j\right]}{\left(\frac{s}{2}\right)^2}$

$\operatorname{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$
Claim: If $x$ and $y$ are pairwise independent (meaning $E[xy] = E[x]E[y]$) then 
$$\text{Var}[x+y] = \text{Var}[x] + \text{Var}[y].$$

Claim: $\text{Var}[E_i] < \frac{1}{4}$ for $\varepsilon > 0$, $\forall i \in [s]$.

Prove above claims at home.

Take $s = \frac{4}{\varepsilon^2}$

So we proved that
$$\Pr\left[ \bigwedge_{x^{(1)} \neq x^{(2)}} \Pr_{x^{(1)}} \left[ A \left( z \right) \neq l_{a}(z) \right] \leq \frac{1}{16} \right].$$

By Markov
$$\Pr\left[ \bigwedge_{x^{(1)} \neq x^{(2)}} \Pr_{x^{(1)}} \left[ \bigwedge_{x^{(3)} \neq x^{(5)}} \Pr_{x^{(3)}} \left[ A \left( z \right) \neq l_{a}(z) \right] \leq \frac{1}{8} \right] \leq \frac{1}{8} \right] \geq \frac{1}{2}$$

$$\Pr\left[ \bigwedge_{x^{(1)} \neq x^{(2)}} \Pr_{x^{(1)}} \left[ \bigwedge_{x^{(3)} \neq x^{(5)}} \Pr_{x^{(3)}} \left[ A \left( z \right) \neq l_{a}(z) \right] \leq \frac{1}{8} \right] \leq \frac{1}{16} \right] \geq \frac{1}{2}.$$
Remarks about above algorithm

• do not have access to \( l_a(x^{(i)}) \) (that's what we need to find)

• Running time \( O\left( \frac{1}{\varepsilon^2} k \log k \right) \) (if we knew \( l_a(x^{(i)}) \))

→ Can run through all possible vals

\[
l_a(x^{(1)}), l_a(x^{(2)}), \ldots, l_a(x^{(5)})
\]

So 2^5 possibilities.

Each gives a machine \( A^{b_1, \ldots, b_5}_{x^{(1)} \ldots x^{(5)}} \)

\[
s = O\left( \frac{1}{\varepsilon^2} \right) \Rightarrow 2^s \text{ machines} \Rightarrow O\left( 2^{\frac{1}{\varepsilon^2} \log k} \right) \text{alg.}
\]

Can do better: use pseudorandomness
Recall $x^{(1)}, \ldots, x^{(5)}$ are independent above but we only use **pairwise independence** in Claim 10.

New idea 1: pick a subspace of small dim (dim logs ?)

New idea 2: Fix value of possible $\lambda a$ at $x^{(1)}, \ldots, x^{(r)}$. Say $La(x^{(i)}) = b; \in \Xi_{0,1}$.
Then \( l_a \) is fixed on entire subspace \( T \); i.e.,
\[
    l_a(\mathbf{x}^S) = \sum_{i \in S} l_a(x^{(i)})
\]

Only need to guess \( l_a \) at \( \log s \) many positions. Everything else in the subspace generated by \( x^{(1)}, \ldots, x^{(\log s)} \) is obtained by taking linear combos of \( b_i \)'s.

So, we can assume known val of \( l_a \) at \( s \) points, namely at all \( x^S \), \( S \subseteq [\log s] \).

Recall \( s = O\left(\frac{1}{\epsilon^2}\right) \) (\# of machines output)
Let $E_i = \{ x : \text{if } f(2 + x^c) = l_{\infty}(x^c) \}
$.

Claim: Variables $E_i$, $i = 1, \ldots, s$ generated above are pairwise independent. (Show at home)

Final alg: (local list decoder for Had)

Pick $x^{(1)}, \ldots, x^{(r)} \ i.u.r \ \in \{0,1\}^r
$

And $(b_1, \ldots, b_r) \in \{0,1\}^r
$

For $s \subseteq \{1, \ldots, r\}$, $s \neq \emptyset$

Assign $x^s = \sum_{i \in s} x^{(i)}$

$b^s = \sum_{i \in s} b^{(i)}$

Define $A^{b_1, \ldots, b_r} (2) = \text{maj } f(x^{(s+2)} - b^s, x^{(1)}, x^{(2)}, \ldots, x^{(r)})$

For $i = 1$ to $k$

for $j = 1$ to $\log k$

pick $x \in \{0,1\}^k \text{ u.a.r.}$, $a_{ij} = A^{'x}{(2 + e^j)} - A^{'x}$

output $a'_i = \text{maj } (a_{i1}, \ldots, a_{ik})$
Proof: By Claim 10, Machine $A^{b_1, -b_2}_{x^{b_3}/x^{b_4}}$ is $\frac{7}{8}$-close to $L_a$.

By Claim 9, we can correct/decode $L_a$.

Running time: $O\left(\frac{1}{\epsilon^2} \cdot \frac{1}{\epsilon^3} \cdot k \log k\right)$

List size: $s = O\left(\frac{1}{\epsilon^3}\right)$

List decoding radius for Had is $\frac{1}{2} - \epsilon$.

(Exp many codew at dist $\frac{1}{2}$)
Applications to crypto / pseudorandomness

Boolean circuit: inputs $x_1, \ldots, x_n$
output $y_1, \ldots, y_m$
gates: AND, OR, NOT
computes function $f : \{0,1\}^n \to \{0,1\}^m$
m=1, then $f$ is called 
predicate

Pseudorandom generator (PRG) \([\text{Blum Micali Yao type PRG}]\)

Random variable $X$ over $\{0,1\}^n$, is $(S, \epsilon)$-pseudorandom

If a circuit $C$ of size $\leq S$

$$\left| \Pr_{x \sim X} [C(x) = 1] - \Pr_{y \sim \text{Un}_n} [C(y) = 1] \right| < \epsilon$$

To $C$ it is indistinguishable
if $x \sim X$ or $X \sim \text{Un}_n$

uniform over $n$ bits
Def: One-way permutation:
A bijective fun $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is $(S, \varepsilon)$-one way permutation if $\forall$ circuits $C$ of size $\leq S$, $C$ cannot succeed in inverting $f$ except on an $\leq \varepsilon$ fraction of inputs, i.e.
$$\Pr_{x\sim\{0,1\}^n}\left[ C(f(x)) = x \right] \leq \varepsilon$$

Def: Hard core predicates:
A function $B: \{0,1\}^n \rightarrow \{0,1\}$ is $(S, \varepsilon)$-hardcore predicate for perm. $f: \{0,1\}^n \rightarrow \{0,1\}^n$ if
$$\forall \text{ circuit } C \text{ of size } \leq S, \text{ (the circuit cannot compute a function of the unknown x except with}}$$
$$\Pr_{x\sim\{0,1\}^n}\left[ C(f(x)) = B(x) \right] \leq \frac{1}{2} + \varepsilon \text{ (bias } \varepsilon)$$
Thm. (Corollary to Goldreich Levin alg we saw above)

Let $p : \{0, 1\}^n \to \{0, 1\}^n$ be an $(S, \varepsilon)$ one way permutation computed by circuits of size $t$.

Let $g : \{0, 1\}^{2n} \to \{0, 1\}^{2n}$ be $g(x, y) = (p(x), y)$.

Then

1. $g$ is a one-way permutation
2. $B(x, y) = x \cdot y \mod 2$ is a $(S', \varepsilon')$-hard core predicate for $g(x, y)$.
3. $S' = \text{poly}(S, t, n)$

Punchline / Thm PRG-GL $\implies$ PRG $\implies$ to obtain pseudo random bits

Thm. (Blum Micali Yao): If $B$ is a hard core pred. for $f$, then $(f(x), B(f(x)))$ is a PRG

$\uparrow$ obtains one extra pseudo random bit
Proof of Thm. PRG - GL from the local list decoder GL.

1) easy to check.

2+3 Assume for sake of contradiction that B(x, y) is not (S', ε') hard core for g, so I a circuit C of size ≤ S' s.t.

\[ \Pr \left[ C(p(x), y) \neq B(x, y) \right] < \frac{1}{2} - \varepsilon' \]

x, y \sim \{0, 1\}^n

(for some S', ε' to be decided later)

By Markov:

\[ \Pr \left[ \Pr_x \left[ C(p(x), y) = B(x, y) \right] > \frac{1}{2} - \varepsilon'' \right] < \frac{1}{2} - \varepsilon'' \]

Hence

\[ \Pr_x \left[ \Pr_y \left[ C(p(x), y) = B(x, y) \right] < \frac{1}{2} - \varepsilon'' \right] > \varepsilon'''' \]
Hence for an $\varepsilon''$ fraction of $x$'s we have

$$\Pr_y \left[ C(p(x), y) = B(x, y) \right] \geq \frac{1}{2} + \varepsilon''$$

For such an $x$, define $f_x(y) = C(p(x), y)$

$$= x \cdot y \text{ wp} \frac{1}{2} + \varepsilon''$$

$$\downarrow = e_x(y)$$

We can now show that local list decoder above implies we can find $x \in S$

For such $x$ for which $Pr_y \left[ C(p(x), y) = B(x, y) \right] \geq \frac{1}{2} + \varepsilon''$

Use GL alg to output list $L$ of all $a$'s that agree with $f_x$ on $\geq \frac{1}{2} + \varepsilon''$ fraction of inputs.

Given value of $p(x)$.

Compare $p(x)$ with $p(a)$ for all $a \in L$.

We succeeded to invert $p(x)$ if $p(x) = p(a)$.
Parameters:

To get contradiction, need \( \frac{1}{\ell} \leq \frac{1}{3} \) and size of new circuit \( \leq S \).

E.g. take \( \varepsilon'' = 2\varepsilon \), \( \varepsilon' = 3\varepsilon \) so

\[
\varepsilon' = \frac{1}{2} - (\frac{1}{2} - 3\varepsilon)(1-2\varepsilon) = 2(3-3\varepsilon^2)
\]

Note: When we make a query to \( f \) in the GL alg above, we need to replace it with a circuit computation.

So total compact time / circuit size

\[
T = O\left( S' \frac{1}{\ell} n \log n + \frac{1}{\varepsilon^2} + \frac{1}{\varepsilon} \right)
\]

Choose \( S = \text{poly}(S', t, n, \frac{1}{\varepsilon}) \) s.t. \( T < S \), concluding the contradiction. 

\[\checkmark\]