Local List Decoding

Madhu Sudan
Microsoft Research

Overview

- Last 20 years:
  - Lots of work on List Decoding
  - Lots of work on Local Decoding

- Today:
  - A look at the intersection: Local List Decoding
    - Part I: The accidental beginnings
    - Part II: Some applications
    - Part III: Some LLD codes
    - Part IV: Current works

Part I: History

List Decodable Code

- Encoding function: $E: \Sigma^k \to \Sigma^n$
- Code: $C = \mathrm{Image}(E)$
- (ρ, L) -List-Decodable Code: $\forall\ r \in \Sigma^n$,

  $\#\{w \in C \mid \Delta(r,w) \leq \rho.n\} \leq L$.
- List-decoder: Outputs list, given r.

- [Elias '57, Wozencraft '58]

Local (Unique) Decoding

- ρ-decoder:
  - Has access to r s.t. $\Delta(r, E(m)) \leq \rho.n$
  - Outputs m.

- ρ-local decoder:
  - Has query access to $r: [n] \to \Sigma$.
  - Input: $i \in [k]$
  - Outputs: m_i

- (ρ,t)-LDC: makes $\leq t$ queries for every r,i.

Local List Decoding

- ρ-List decoder:
  - Access to $r \in \Sigma^n$
  - Outputs $\{m_1,\ldots,m_L\} = \{m \mid \Delta(r,E(m)) \leq \rho.n\}$

- (ρ,t)-list decoder:
  - Query access to $r:[n] \rightarrow \Sigma$
  - Inputs: $i \in [k]$, $j \in [L]$
  - Outputs: $(m_j)_i$

- Note: numbering $m_1,\ldots,m_L$ may be arbitrary;
  but consistent as we vary i.

(Convoluted) History

- 1950 [Reed+Muller]:
  - Code due to Muller; Decoder due to Reed.
    - “Majority-logic” decoder: Essentially a local decoder for $\rho <$ distance/2,
    - Not stated/analyzed in local terms.
- 1957 [Elias]
  - Defined List Decoding.
  - Analyzed in “random-error” setting only.
- [1980s] Many works on random-self-reducibility
  - Essentially: Local decoders (for un/natural codes).

(Convoluted) History

- 1986 [Goldreich-Levin]:
  - Local List-decoder for Hadamard code.
    - No mention of any of the words in paper.
    - “List-decoding” in acknowledgments.
    - But idea certainly there – also in [Levin 85]
    - (many variations since: KM, GRS).
- 90-92 [BeaverFeigenbaum, Lipton, GemmellLiptonRubinfeldSWigderson,GemmellS.]:
  - Local decoder for generalized RM codes.
- 96,98 [Guruswami+S]:
  - List-decoder for Reed-Solomon codes.

(Convoluted) History

- 1999 [S.TrevisanVadhan]:
  - Local List-Decoding defined
  - LLD for Generalized RM code.
- 2000 [KatzTrevisan]:
  - Local Decoding defined.
  - Lower bounds for LDCs.

Why Convoluted?

- What is convoluted?
  - Big gap (positive/negative) between definitions and algorithms
- Why?
  - Motivations/Applications changing.
  - Algorithms not crucial to early applications
  - Some applications needed specific codes
  - Different communities involved
    - Information theory/Coding theory
    - CS: Complexity/Crypto/Learning

Part II: Applications

Hardcore Predicates

- $f:\{0,1\}^n \to \{0,1\}^n$ is a owf if
  - f easy to compute
  - f^{-1} hard on random inputs:
    - random: given $y = f(x)$ for uniform x, output x' in $f^{-1}(y)$.
    - hard: every polytime alg. succeeds with negligible probabilty.

- $b:\{0,1\}^n \to \{0,1\}$ is hardcore predicate for f, if f remains hard to invert given b(x) and f(x)

Hardcore Predicates

- $b:\{0,1\}^n \times [M] \to \{0,1\}$ is a (randomized) hardcore predicate for f, if $b(x,s)$ hard to predict w.p. $\frac{1}{2} + \epsilon$, given $f(x)$ and s.

- [BlumMicali,Yao,GoldreichLevin]:
1-1 owf f + hardcore b $\Rightarrow$ pseudorandom generator.

- [GoldreichLevin,Impagliazzo]:
If $E:\{0,1\}^k \to \{0,1\}^m$ is a $(\frac{1}{2} - \epsilon, \mathrm{poly}(n))$-LLDC, then $b(x,s) = E(x)_s$ is a hardcore predicate for every owf f.

Proof of [GL,I]

- Suppose A predicts $b(x,s)$ given $f(x)$, s
- Fix $f(x)$; let $r(s) = A(f(x),s)$.
- Run Decoder(r,i,j) for all i,j to recover $\{x_1,\ldots,x_L\}$.
- Check if $f(x_j) = f(x)$!
- (Easy) Claim: This recovers $f^{-1}(f(x))$ w.h.p.

Thoughts

- Did [GL] really need Local List-Decoding?
  - No. Simple LDC mapping k to poly(k) bits would do.
  - Unfortunately, none was known with poly(k) time list-decoder.
  - GL: Designed (½ - epsilon,poly(k))-LLDC for Hadamard code (which maps k bits to 2^k bits).

Hardness amplification

- Classical quest in complexity:
  - Find hard functions (for some class). E.g.,
    - $f \in NP - P$
    - $f \in PSPACE - P$
    - Story so far: Can't find such.
- Modern question:
  - Find functions that are really hard.
    - Boolean $f \in NP$ that is hard to distinguish from random function in P.

Hardness amplification

- Thm [Lipton, ..., S. Trevisan Vadhan]:
  - Let $f:\{0,1\}^k \to \{0,1\}$ be a hard to compute in time poly(k).
  - Let $E:\{0,1\}^K \to \{0,1\}^N$ be (½-ϵ,poly(k)) locally-list-decodable with $K = 2^k$, $N = 2^n$.
  - Then $g:\{0,1\}^n \to \{0,1\}$ given by $g = E(f)$ is hard to distinguish from random for poly(k) time algorithms.

- Proof: Obvious from definitions.

Agnostic Learning

- General goal of learning theory:
  - Given a class of functions F;
  - query/sample access to $f \in F$;
  - "Learn f" (or circuit (approx.) computing it).
- Learning with Noise:
  - f not in F, but well-approximated by some function in F
- Agnostic Learning:
  - No relationship between f and F;
  - learn some approximation of f in F (if it exists).
- Useful in applications, as well as theory.

Agnostic Learning (contd.)

- GL result (Kushilevitz-Mansour interpretation):
  - Can agnostically learn linear approximations to Boolean functions, with queries.
- Kushilevitz-Mansour:
  - List-decoding helps even more: Can learn decision trees.
- Jackson:
  - Also CNF/DNF formulae ...

Part III: Some LLD Codes

Hadamard Code

- Code: Maps $\{0,1\}^k \to \{0,1\}^{2^k}$.
- Codewords:
  - functions from $\{0,1\}^k \to \{0,1\}$.
  - Encoding of m = $<m_1,\ldots,m_k>$ is the function $E_m(y_1 \ldots y_k) = \Sigma_{i=1}^{k} m_i\, y_i \pmod 2$.
  - I.e., codewords are homogenous, k-variate, degree 1 polynomials over GF(2).

Decoding Hadamard Code (GL/KM)

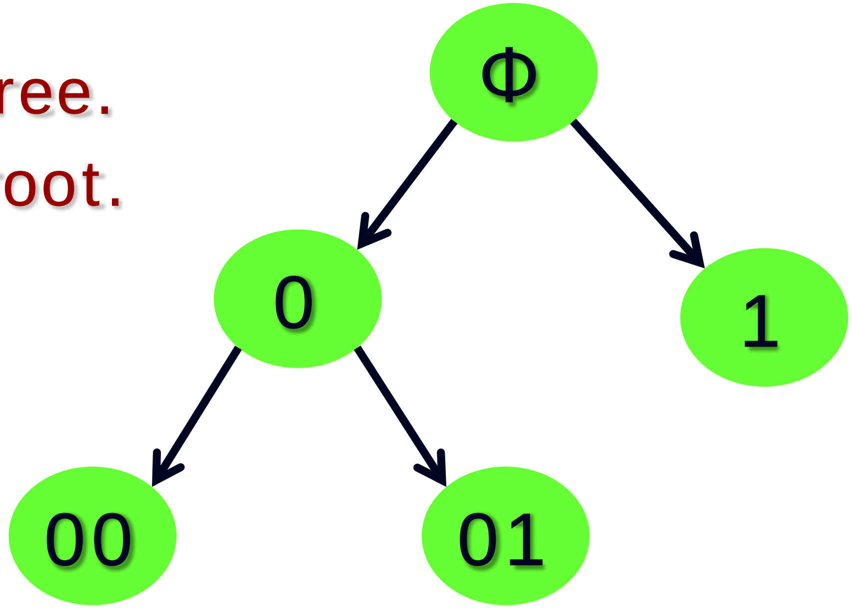
- Preliminaries:
  - View words as functions mapping to {+1,-1}.
  - $\langle f,g\rangle$ = Exp_y [f(y).g(y)].
  - $\langle E(a),E(b)\rangle = 0$ if $a \neq b$ and 1 o.w.
  - Let f_a = $\langle f,E(a)\rangle$.

    Then $f[x] = \sum_a f_a E(a)[x]$
  - For all f, $\sum_a f_a^2 = 1$.

  - $(\frac{1}{2} - \epsilon)$-List decoding: Given f, find all a such that $f_a > 2\epsilon$.

Decoding Hadamard Code [GL/KM]

- Consider 2^n sized binary tree.
- Node labelled by path to root.
- Value of leaf $a = f_a^2$
- Value of node
  = sum of children values



- Main idea: Can approximate value of any node
  - $\sum_b f_{ab}^2 = \mathrm{Exp}_{x,y,z} [f(xz).f(yz).E_a(x).E_a(y)]$
- Algorithm:
  - Explore tree root downwards.
  - Stop if node value less than ϵ^2
  - Report all leaves found.

(Generalized) Reed-Muller Code

- Message space = m-variate, degree r polynomials over GF(q).
- Encoding: Values over all points.
  - $k = \binom{m+r}{r}$
  - $n = q^m$
  - distance $= 1 - r/q$ (if $r < q$).
    $\approx q^{-r/(q-1)}$ if $r > q$.
- Decoding problem: Given query access to function that is close to polynomial, find all nearby polynomials, locally.

Decoding (contd.)

- Specifically:
  - Given query access to f, and $x \in GF(q)^m$
  - Output $p_1(x),\ldots, p_L(x)$ "consistently", where p_j's are polynomials within distance ρ of f.
- How to index the codewords?
  - By values at a few (random) points in $GF(q)^m$.
  - Claim: Specifying value of p at (roughly) $\log_q L$ points specifies it uniquely (given f).

Decoding (contd.)

- Refined question:
  - Given query access to f, and values $p_j(y_1),\ldots,p_j(y_t)$, and x;
  - Compute $p_j(x)$

- Alg [Rackoff, STV, GKZ]
  - Pick random (low-dim) subspace containing $y_1,\ldots,y_t$ and x.
  - Brute force decode f restricted to this subspace.

Part IV: Current Directions

Many interpretations of GL

- List-decoder for group homomorphisms [Dinur Grigorescu Kopparty S.]
  - Set of homomorphisms from G to H form an error-correcting code.
  - Decode upto minimum distance?
- List-decoder for sparse high-distance linear codes [Kopparty Saraf]
- List-decoder for Reed-Muller codes [Gopalan Klivans Zuckerman]

Approximate List-Decoding

- Given r, approximately compute w in C that is somewhat close to r.
- Easier problem, so should be solvable for broader class of codes C (C need not have good distance).
- [O'Donnell, Trevisan, IJK]: If encoder for C is monotone and local, then get hardness amplification for NP.
- [IJK] Give approximate-LLD for "truncated Hadamard code".

Conclusions

- Intersection of Locality and List-decoding interesting and challenging.

- Ought to be explored more?

Thank You!