CS 229r Essential Coding Theory Apr 27, 2020

Lecture 24

Instructor: Madhu Sudan *Scribe: Junu Lee*

1 Today

- Coding for Interactions
  - Setup: Interactions and Coding Protocols
  - History
  - Tree Codes
  - Braverman-Rao Protocol (2011)

2 Interactions

An interaction can be thought of as a conversation between two entities (named "Alice" and "Bob"). As with any conversation, an error caused by a misunderstanding can derail a conversation. For example, consider the following exchange:

Alice: "Please explain the proof of Fermat's [unintelligible] Theorem."
Bob: *spends four lectures going over group theory, Galois groups, semi-stable representations*
Alice: "I said Fermat's *Little* Theorem".

It seems quite possible for some adversary to make keywords of these sentences unintelligible, making communication much more inefficient. We formalize this idea of this communication, called an *interaction*, and discuss ways to protect interactions against adversaries that may change the communicated subject between Alice and Bob.

2.1 Definitions

Before we delve into the formal mathematical definition, we give the intuition for how we view a conversation. Define the graph (V, E) as a rooted binary tree with length k, i.e. $V = \{0,1\}^k$ and $E = \{(x, x0), (x, x1) \mid x \in \{0,1\}^{<k}\}$.

Definition 1. An *interaction* (between Alice and Bob) is a subgraph of the rooted binary tree (V, E) of depth k. Alice's input consists of, for every node of even depth, an edge from it to one of its children. Bob's input consists of, for every node of odd depth, an edge from it to one of its children. The message is the path from the root to a leaf formed by these inputs, and corresponds to some bit-string in $\{0,1\}^k$.

Using this notion, we formally define the idea of a protocol.

Definition 2. A *(binary) protocol* of length k is a binary rooted tree (V, E) of depth k, combined with vertex subsets $V_A, V_B \subseteq V$, edge subsets $E_A, E_B \subseteq E$, and message sets Π_A and Π_B. We can represent it as a eight-tuple $(V, E, V_A, V_B, E_A, E_B, \Pi_A, \Pi_B)$. Formally, we have the following components.

- $V = \{0,1\}^{\leq k}$, and $V_A \cup V_B = V$.
  - V_A can be seen as the inputs for Alice (i.e. even-depth nodes) and V_B can be seen as the inputs for Bob (i.e. odd-depth nodes).

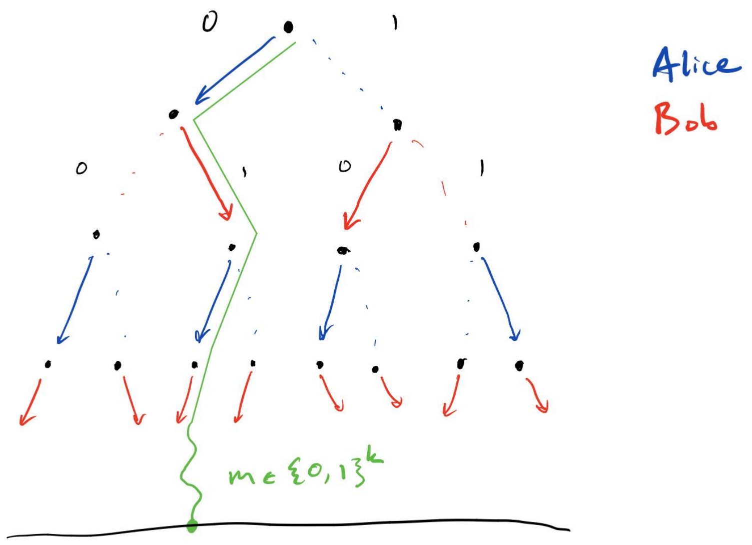

Figure 1: A visual of an interaction.

- $E = \{(x, x0), (x, x1) \mid x \in \{0,1\}^{<k}\}$, and $E_A \cup E_B = E$.
  - E_A and E_B are the directed edges that Alice and Bob can use for their interaction, respectively (starting from their respective vertex sets).
- $\Pi_A \subseteq E_A$, $\Pi_B \subseteq E_B$
  - Π_A is a subset of E_A such that for each vertex v in V_A, there is exactly one edge in Π_A such that v is its tail. The respective notion goes for Π_B.

The sets Π_A and Π_B make the direction of the conversation deterministic: for every node that A is given (i.e. B communicated an edge whose head was that node), there is exactly one possible reply by A.

Definition 3. We say that some $m \in \{0,1\}^k$, treated as a path from the root to a leaf in (V, E), is *A-valid* if $m \subseteq \Pi_A \cup E_B$. We say m is *B-valid* if $m \subseteq \Pi_B \cup E_A$. If m is both A-valid and B-valid, then we call it a *valid transcript*.

It is evident that there exists a unique valid transcript in $\Pi_A \cup \Pi_B \subseteq E$. A simple algorithm that outputs this transcript in this noiseless setting has Alice and Bob sending their collective path so far after every turn. This requires at most k bits of communication per turn, which comes out to $\frac{k(k+1)}{2} = O(k^2)$ bits required total. This is extremely inefficient, especially in the noiseless setting.

2.2 Interactive Coding

We have seen the general idea of an interaction above. Now we must introduce architecture to protect such interactions from adversaries that could lead us to believe the validity of a totally incorrect transcript. To do so, we bring our coding-theoretic intuition of mapping k bits to n bits, so to speak.

Definition 4. An *encoding scheme* is a pair of encoding functions $\mathcal{E}_A$ and $\mathcal{E}_B$, a pair of decoding functions f_A, f_B, and binary protocol $(V_n, E_n, W_A, W_B, F_A, F_B, \Gamma_A, \Gamma_B)$ of length n such that

- $\mathcal{E}_A \colon \Pi_A \mapsto \Gamma_A \subseteq F_A$, $\mathcal{E}_B \colon \Pi_B \mapsto \Gamma_B \subseteq F_B$, where by definition we have that $F_A \cup F_B = E_n$.
- $f_A, f_B \colon \{0,1\}^n \to \{0,1\}^k$.

Definition 5. We say that an encoding scheme corrects e errors if for all $a, b \in \{0,1\}^n$ such that the following three hold:

- a is A-valid, i.e. contained in $\Gamma_A \cup F_B$
- b is B-valid, i.e. contained in $\Gamma_B \cup F_A$
- $\Delta(a, b) \le e$

then $f_A(a) = f_B(b) = m$, and m is a valid transcript on the original binary protocol (of depth k).

Note that an encoding scheme can be seen as a code that maps a binary protocol of depth k to one of depth n (in this vein, the latter can be implicitly specified, but we do so explicitly to demonstrate the intuition). This metaphor is furthered when considering the role of e-error correction—any adversary who is not permitted to flip more than e bits will be outfoxed by the encoding scheme.

3 History

- Schulman '90: Introduced the problem, later came up with "tree codes" and a specific parametrization with $\Omega(1)$ rate and $\Omega(1)$-fraction error correction.
- Braverman-Rao '11: Created a protocol of rate $\Omega(1)$ correcting $\frac{1}{8}$-fraction errors.

More works have followed, but these results are the main ones in the field.

4 Tree Codes

Definition 6. A *tree code* $T \colon \{0,1\}^* \to \{0,1\}^*$ is a function such that

- $\forall x, y, |x| = |y| \implies |T(x)| = |T(y)|$.
- $\forall x$, $T(x)$ is a prefix for both $T(x0)$ and $T(x1)$

The latter property is comparable to the notion of "online coding", as adding a bit to the message corresponds to adding a set number of bits to its codeword.

Tree codes also have rate and distance parameters. Formally,

Definition 7. Given a tree code T, we say it has *rate* R if $\forall k, \forall x \in \{0,1\}^k$, we have the condition

$$|T(x)| \le \frac{k}{R}.$$

We say that T has *distance* δ if $\forall x, y \in \{0,1\}^k, \forall i \in [k]$, we have that

$$x_i \neq y_i \implies \Delta(T(x), T(y)) \ge \frac{\delta}{R}(k - i + 1).$$

In the current treatment of tree codes, we usually set $R = \frac{1}{s}$, where s is an integer. Then n is a multiple of k, which will help us visualize some deeper analysis of these codes. Building on this, we prove the existence of a "very good" tree code.

Proposition 8. *There exists a tree code T with rate R and distance δ such that $R, \delta = \Omega(1)$.*

Proof. Let s be a positive integer greater than 1. For a fixed k, take T to be a linear code with an "upper triangular" generator $G \in \mathbb{F}_2^{k\times sk}$, written as

$$G = \begin{pmatrix} *_1 & *_2 & *_3 & \cdots & *_k \\ 0 & *_1 & *_2 & \cdots & *_{k-1} \\ 0 & 0 & *_1 & \cdots & *_{k-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & *_1 \end{pmatrix}, \tag{1}$$

where $*_i \in \mathbb{F}_2^s$ are chosen at random for all $i \in [k]$. Note the Toeplitz-ness of the structure, which induces the online learning property of T. It is easy to see that this is a valid tree code, and furthermore we can construct such a generator G for any message length k.

We can quickly see that the rate of this code is $\frac{1}{s}$, a constant fraction. If we set $\delta = \Omega(1)$ we can show that the probability of "failure" for our matrix G is strictly less than 1, where failure comes from δ not satisfying the definition of distance. Thus, such a code with the desired parameters exist. $\square$

Exercise 9. *We verify the probabilistic claim above through a recursive construction. Let G_{2^t} be a $2^t \times 2^t s$ matrix built by the following recursion:*

$$G_{2k} = \begin{pmatrix} G_k & H_k \\ 0 & G_k, \end{pmatrix}$$

where H_k is a random matrix such that G_{2k} satisfies the block-Toeplitzness specified in Proposition 8.

(a) Prove that $\mathbb{P}[G_{2k} \text{ does not work} | G_k \text{ works}] = \exp(-k)$.

(b) Use the above to conclude the probabilistic claim from Proposition 8.

Exercise 10. *Over large alphabets, tree codes with δ approaching 1 and $R > 0$ exist.*

5 The Braverman-Rao Protocol

Braverman and Rao utilize tree codes in their 2011 result, which was described above. We give the protocol in more detail. We fix tree codes T^A, T^B (which are not necessarily identical). Then Alice's protocol is the following:

- Let her *state* be some sequence $S_A = (e_1, \dots, e_i) \subseteq \Pi_A$ of edges she has communicated up until her i^{th} round of communication.
- In the j^{th} round of communication:
  - Alice receives message m_j^B, which she appends to a running sequence $m_{\leq j}^B$.
  - She decodes to a sequence $R_{B,j} = \operatorname{argmin} \delta(m_{\leq j}^B, T^B(R_{B,j}))$. This can be computed through brute force.
- Let e be the successor to the path $R_{B,j} \cup S_A$.
  - If e is not in S_A, then $S_A \leftarrow S_A$ appended by e.
  - If else, do nothing.
- Send the newest bits of $T^A(S_A)$ to Bob (i.e. $T^A(S_A) \setminus T^A(S_A)$).
- We repeat the above for n steps. After n rounds, if there is a path from root to leaf in $R_{B,n} \cup S_A$, let this be Alice's output.

The analogous follows for Bob, who then outputs his path. If they match, we output it as our valid transcript.

5.1 Compression

For Alice and Bob to efficiently communicate, we cannot afford to have them send the encodings of S_A and S_B explicitly across the channel for every round. This leads to $O(n^2)$ total bits sent (treating the rate of the tree code as constant). We have two ideas for compression.

First, if we were to append e_i to S_A, then at some point previously we must have presented edge e_j such that it led Bob to communicate an edge whose head is the tail of e_i. Thus, to Bob, j restricts e_i to at most four possibilities. Two bits $b_1 b_2$ are enough to specify which of these four is e_i. Thus, communicating e_i as $(j, b_1 b_2)$ achieves $\log n$ compression, giving us a total of $O(n \log n)$ bits communicated.

However, we can do better. In general we expect $i - j$ to be small, so instead of communicating the pair $(j, b_1 b_2)$, we may communicate the pair $(i - j, b_1 b_2)$. Overall, we achieve near-linear performance.

5.2 Analysis

We provide a brief highlight of the analysis used by Braverman and Rao to prove the correctness of the algorithm. First, we define some notation:

- $N(i, j)$ is the number of errors during transmissions i to j.
- $m(i)$ is the length of prefix in both Γ_A and Γ_B agreed upon after i transmissions.
- $t(j)$ is the first time that the j^{th} edge enters S^A and is announced to Bob.

From this we have three lemmas from Braverman and Rao that prove correctness. The proofs are found in their paper, but we provide a bit of logical intuition for why these statements hold true.

Lemma 11 (Key Lemmas)**.**

(1) $m(n) \geq t(k)$

- *If we ever get the error $f_A(a) \neq f_B(b)$, then we know that $m(n) < t(k)$.*

(2) $m(t(i) - 2) < t(i - 1)$

- *If Alice and Bob have not agreed on the i^{th} edges, then they must have had miscommunication in the time prior to announcing the $(i-1)^{th}$ edge.*

(3) $N(m(i) + 1, i) \geq \frac{\delta}{2}(i - m(i))$.

-

Combining the three lemmas, Braverman and Rao conclude that the protocol above corrects $\frac{\delta}{2}$-fraction errors.

The main idea is that we measure our progress by looking at the longest path length in $S_A \cup S_B$ so far. As we never delete anything, even when a miscommunication is evident, we never get worse during a round. If Alice adds some edge to S_A and Bob does not add a next edge to S_B, then it is because of the adversary. But the adversary only has a limited number of errors it can cause, so S_B will eventually contain the next edge, and the protocol will eventually approach the true path.

6 Parting Shots

In this lecture we described the mechanism of tree codes as a way to protect against noise in an interaction. We also discussed two examples of protocols—the original by Schulman and another by Braverman and Rao—that utilize these tree codes. However, it is worth noting that none of these are constructive. It also should be said that constructive encoding and decoding of tree codes would imply that the Braverman-Rao protocol is constructive. Currently, there have been no results showing the constructiveness of the protocol, but there are work-around ideas to the protocol that are close. However, in literature there are no deterministic efficient protocols for interactive coding, nor are there explicit constructions of any tree code.