CS 2429 - Foundations of Communication Complexity

Lecture #12: 1 December 2009

Lecturer: Toniann Pitassi and Avner Magen

Scribe Notes by: Yuval Filmus

1 Unbounded-error probabilistic communication complexity

In this lecture we will describe the connection between unbounded-error probabilistic communication complexity and the sign-rank, first described in the previous lecture.

First, let us describe the model of computation. It's a two-party communication game with *private* random coins, and a protocol Π is said to compute a function $f(x, y)$ of the two players' inputs if (we use the convention that $\Pi(x, y)$ is the last bit sent)

$$f(x, y) = 0 \Longrightarrow \Pr[\Pi(x, y) = 0] > 1/2,$$
$$f(x, y) = 1 \Longrightarrow \Pr[\Pi(x, y) = 1] > 1/2.$$

In other words, we want the protocol to be correct with probability strictly greater than 1/2:

$$\Pr[\Pi(x, y) = f(x, y)] > 1/2.$$

In contrast, in the past we were concerned with protocols that were correct with probability *bounded away* from 1/2, e.g. protocols which are correct with probability at least 2/3. The new model does not require us to bound the error away from 1/2, i.e. it requires *unbounded error*.

As usual, the communication complexity is the maximum number of bits sent, given the length of the input. The class of functions computable using $O(\log n)$ communication bits is denoted $\mathrm{UPP}^{\mathrm{cc}}$.

Note that the model becomes trivial if we allow access to *public* random coins. Indeed, the following protocol calculates any given function $f(x, y)$:

1. The two players agree on an n-bit segment z of the public random coins.
2. Alice compares her input x to z. If $x = z$, see sends Bob the bit 1, otherwise she sends him the bit 0.
3. If Bob receives 1, he outputs $f(z, y)$. Otherwise, he outputs a random unbiased coin flip.

Bob outputs the correct output with probability at least

$$2^{-n} \cdot 1 + (1 - 2^{-n}) \cdot \frac{1}{2} = \frac{1}{2} + 2^{-(n+1)} > \frac{1}{2}.$$

Therefore it is essential that the randomness source be private.

2 Normal form: one way protocols

We now describe a normal form for unbounded-error protocols. This normal form will enable us to relate unbounded-error communication complexity to sign-rank. The normal form is a one-way protocol, that is a protocol in which Alice sends a bunch of bits to Bob, and then Bob simply outputs the answer. We note in passing that one-way protocols are also useful for proving lower bounds for streaming protocols.

Let Π be an unbounded-error protocol computing some function $f(x, y)$. We make the simplifying assumption that communication follows the following regime: Alice sends the first message, and Bob's last message is the protocol's output.

The communication between Alice and Bob is described by its *transcript* or *log*, which is the listing of messages sent by each party. Let us denote these messages by

$$\mathbf{a}_1,\, \mathbf{b}_1,\, \ldots,\, \mathbf{a}_\mathbf{T},\, \mathbf{b}_\mathbf{T}.$$

Here $\mathbf{a}_i$ is the i'th message sent by Alice to Bob, and $\mathbf{b}_i$ is the i'th message sent by Bob to Alice. These messages are random variables depending on each player's private random coins. They also depend on the players' inputs x, y. The number of rounds $\mathbf{T}$ is also a random variable. We denote the entire log by $\mathbf{l}$, which is also a random variable.

The protocol Π is completely specified by the following functions:

$$\alpha_t(x,\, \mathbf{l}) = \Pr[\text{Alice sends } \mathbf{a}_t \text{ given } x \text{ and } \mathbf{a}_1,\, \ldots,\, \mathbf{b}_{t-1}],$$
$$\beta_t(y,\, \mathbf{l}) = \Pr[\text{Bob sends } \mathbf{b}_t \text{ given } y \text{ and } \mathbf{a}_1,\, \ldots,\, \mathbf{a}_t].$$

These functions really only depend on the part of the log just before the corresponding message is sent, but for ease of notation we use the entire log as argument.

We immediately get

$$\Pr[\mathbf{l} \mid x, y] = \prod_{t=1}^{\mathbf{T}} \alpha_t(x,\, \mathbf{l})\beta_t(x,\, \mathbf{l}).$$

So far for the two-way protocol Π. The one-way protocol works, essentially, by Alice guessing all of Bob's responses, and Bob correcting their probabilities with his input. The actual design of the protocol is somewhat intricate. We first define the one-way probabilities of logs:

$$\alpha(x,\, \mathbf{l}) = \prod_{t=1}^{\mathbf{T}} \alpha_t(x,\, \mathbf{l}),$$
$$\beta(y,\, \mathbf{l}) = \prod_{t=1}^{\mathbf{T}} \beta_t(y,\, \mathbf{l}).$$

Next, we define a one-sided acceptance probability:

$$\alpha^{\text{acc}}(x) = \sum_{\text{accepting } \mathbf{l}} \alpha(x,\, \mathbf{l}).$$

The sum is taken over all accepting logs (logs where $\Pi(x, y) = 1$) when Alice's input is x, and Bob's input is arbitrary. A *random accepting log* given Alice's input x is obtained by choosing an accepting log $\mathbf{l}$ with probability $\alpha(x,\, \mathbf{l})/\alpha^{\text{acc}}(x)$.

We can now describe the one-way protocol.

1. Alice's actions:
   (a) With probability $\frac{1}{2} \cdot \frac{\alpha^{\mathrm{acc}}(x)}{\max_x \alpha^{\mathrm{acc}}(x)}$, Alice chooses a random accepting log $\mathbf{l}$ for x, and sends Bob $\langle 0, \mathbf{l} \rangle$.
   (b) With probability $\frac{1}{2}\left(1 - \frac{\alpha^{\mathrm{acc}}(x)}{\max_x \alpha^{\mathrm{acc}}(x)}\right)$, Alice sends Bob $\langle 1, 0 \rangle$.
   (c) With probability $\frac{1}{2}$, Alice sends Bob $\langle 1, 1 \rangle$.
2. Bob's actions:
   (a) If Bob receives $\langle 0, \mathbf{l} \rangle$, he outputs 1 with probability $\beta(y, \mathbf{l})$.
   (b) If Bob receives $\langle 1, 0 \rangle$, he outputs 0.
   (c) If Bob receives $\langle 1, 1 \rangle$, he outputs 1 with probability $1 - \frac{1}{2 \max_x \alpha^{\mathrm{acc}}(x)}$.

This protocol uses two more bits than the original protocol.

What is the probability that the new protocol outputs 1?

$$
\begin{aligned}
&\frac{1}{2 \max_x \alpha^{\mathrm{acc}}(x)} \sum_{\text{accepting } \mathbf{l}} \alpha(x, \mathbf{l}) \beta(y, \mathbf{l}) + 0 + \frac{1}{2} \cdot \left(1 - \frac{1}{2 \max_x \alpha^{\mathrm{acc}}(x)}\right) \\
&= \frac{1}{2} + \frac{1}{2 \max_x \alpha^{\mathrm{acc}}(x)} \left(\sum_{\text{accepting } \mathbf{l}} \Pr[\mathbf{l} \mid x, y] - \frac{1}{2} \right) \\
&= \frac{1}{2} + \frac{1}{2 \max_x \alpha^{\mathrm{acc}}(x)} \left(\Pr[\Pi(x, y) = 1] - \frac{1}{2} \right).
\end{aligned}
$$

Hence the new protocol calculates the same function as the old protocol.

Let's summarize what we've shown:

For every protocol in the unbounded-error communication model there corresponds a one-way unbounded-error communication protocol computing the same function with at most 2 more bits of communication.

3 Connection to the sign-rank

Recall that the sign-rank of a matrix M is the minimum rank of a matrix A such that $M_{ij} = \operatorname{sgn} A_{ij}$. It turns out that the logarithm of the sign-rank of the ± 1 matrix M_f representing a function f is essentially equal to its unbounded-error one-way communication complexity.

Indeed, any one-way protocol Π is completely described by the following two functions:

$$
\begin{aligned}
\alpha_x(m) &= \Pr[\text{Alice sends message } m \text{ given input } x], \\
\beta_y(m) &= \Pr[\text{Bob accepts given message } m \text{ and input } y].
\end{aligned}
$$

We can think of α_x, β_y as probability vectors. Notice that

$$\Pr[\Pi(x, y) = 1] = \langle \alpha_x, \beta_y \rangle.$$

Since α_x is a probability vector, we immediately get

$$\langle \alpha_x, \beta_y - \frac{1}{2} \rangle = \langle \alpha_x, \beta_y \rangle - \frac{1}{2} = \Pr[\Pi(x,y) = 1] - \frac{1}{2} = M_f(x,y).$$

Here $\beta_y - 1/2$ is the vector obtained from β_y by deducting $1/2$ from each element.

Let's pack all the α_x as the rows of some matrix A, and all the $\beta_y - 1/2$ as the columns of some matrix B. Thus

$$M_f = \operatorname{sgn}(AB).$$

The number of possible messages is 2^c, where c is the number of bits sent by Alice in Π. This is also the number of rows of A, and so $\operatorname{rank} AB \leq 2^c$. Hence the sign-rank of M_f is at most 2^c.

Conversely, suppose that M_f is sign-represented by a matrix W of rank r. By taking a basis for the row space of W, we can write W as $W = AB$, where A has r rows, and B has r columns. We will transform this representation, step by step, into a one-way protocol for calculating f:

1. Normalize each column of B so that it sums to one. The resulting matrix B_1 satisfies $M_f = \operatorname{sgn}(AB_1)$.
2. Add a big enough constant C_1 to all entries of A so that no negative entries remain. The resulting matrix A_1 satisfies $M_f = \operatorname{sgn}(A_1 B_1 - C_1)$.
3. Divide all entries of A_1 by a big enough constant C_2 so that all rows sum to at most 1. The resulting matrix A_2 satisfies $M_f = \operatorname{sgn}(A_2 B_1 - C_1/C_2)$.
4. Enlarge A_2 by adding another column, in such a way that all rows now sum exactly to 1. Enlarge B_1 accordingly by adding a corresponding row of zeroes. The resulting matrices A_3, B_2 satisfy $M_f = \operatorname{sgn}(A_3 B_2 - C_1/C_2)$.
5. Add a big enough constant C_3 to all entries of B_2 so that no negative entries remain. The resulting matrix B_3 satisfies $M_f = \operatorname{sgn}(A_3 B_3 - C_1/C_2 - C_3)$.
6. Divide all entries of B_3 by a big enough constant C_4 so that all columns sum to at most 1. The resulting matrix B_4 satisfies $M_f = \operatorname{sgn}(A_3 B_4 - C')$, where we can assume $C' = (C_1/C_2 + C_3)/C_4$ satisfies $0 \leq C' \leq 1/2$.
7. Enlarge B_4 by adding another row, in such a way that all columns now sum exactly to 1. Enlarge A_3 accordingly by adding a corresponding column of zeroes. The resulting matrices A_4, B_5 satisfy $M_f = \operatorname{sgn}(A_4 B_5 - C')$.

The matrices A_4, B_5 define a one-way communication protocol Π with a threshold of C': if $f(x,y) = 0$ then the acceptance probability is smaller than C', and if $f(x,y) = 1$ then the acceptance probability is larger than C'. In order to force a threshold of $1/2$, we slightly modify the protocol:

1. Alice emulates Π on her input, sending a message to Bob.
2. Bob emulates Π on his input and the message received from Alice.
3. If Π would have Bob accept, Bob unconditionally accepts.
4. If Π would have Bob reject, Bob accepts with probability $\dfrac{1/2 - C'}{1 - C'}$.

If the original protocol accepted with probability p, the new protocol accepts with probability

$$p + (1-p)\frac{1/2 - C'}{1 - C'} = \frac{1/2 - C' + p/2}{1 - C'} = \frac{1}{2} + \frac{p - C'}{2(1 - C')}.$$

So the threshold has moved from p to $1/2$. In total, we have transformed a sign-representation of rank r into a protocol using $\lceil \log(r+2) \rceil$ bits (ignoring Bob's bit). If $r > 1$, $\lceil \log(r+2) \rceil \le \log r + 1$. If $r = 1$, there is a corresponding deterministic 1-bit protocol (since we can assume A and B have ± 1 entries).

Summarizing, we have shown the following connection between the unbounded-error one-way communication complexity c of a function f (counting only Alice's sent bits) and the sign-rank r of the corresponding ± 1 matrix M_f:

$$\log r \le c \le \log r + 1.$$

4 Diversion: 2-bit protocol for equality

Deciding equality is extremely hard in the deterministic setting, indeed it requires the maximal number of communication bits $n + 1$. Conversely, there's a randomized algorithm that decides equality with $O(\log 1/\epsilon)$ bits, and is correct with probability $1 - \epsilon$ (and always correct when the inputs are equal). In this section we show that the sign-rank of equality is extremely small: only 2. Using the construction described in the previous section, this gives a 2-bit unbounded-error protocol for equality.

It's easy to see that if $n > 1$ then the sign-rank of the equality matrix cannot be 1. We proceed to describe a sign-rank representation of rank 2. Let $M = 2^n$, and consider the following set of vectors, defined for $0 \le x < M$:

$$a_x = \begin{pmatrix} \cos 2\pi \frac{x}{M} & \sin 2\pi \frac{x}{M} \end{pmatrix}.$$

Elementary trigonometry gives us

$$\langle a_x, a_y \rangle = \cos 2\pi \frac{x - y}{M}.$$

Since $|x - y| < M$, the following dichotomy arises:

- If $x = y$ then $\langle a_x, a_y \rangle = 1$.
- If $x \neq y$ then $\langle a_x, a_y \rangle \le \cos \frac{2\pi}{M} < 1$.

We conclude that the two sets of vectors a_x and $b_y = a_y - (1 + \cos \frac{2\pi}{M})/2$ sign-represent the equality matrix, that is $\langle a_x, b_x \rangle > 0$ while $\langle a_x, b_y \rangle < 0$ for $x \neq y$.

5 Applications

We conclude by presenting a few applications.

5.1 Communication complexity class separation

Our first application, dated 2008, is taken from a paper by Sherstov and Razborov. They consider the following function

$$f(x,y) = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{m^2} (x_{ij} \wedge y_{ij}),$$

and show that its sign-rank is $2^{\Omega(m)}$, using pattern-matrix methods. It's therefore hard to compute $f(x,y)$ in the unbounded-error model. On the other hand, there is a trivial Π_2^{cc} protocol for it requiring only $O(\log m) = O(\log n)$ communication bits (Π_2^{cc} is the natural analog of the corresponding class in the polynomial hierarchy). This shows that Π_2^{cc} is not contained in UPP^{cc}. The other direction remains open: whether there exists a function efficiently computable in the unbounded-error model but not in Π_2^{cc}.

5.2 Learning

The same function $f(x,y)$ described in the previous application is also a hard example for learning algorithms. In the learning framework, we are given a few samples of a Boolean function g taken from a space $\mathcal{F}$ of possible functions, and our goal is to find a function h which is a good estimate of g, in the sense that g and h agree with high probability. The challenge is to learn g with as few samples as possible.

Learning algorithms traditionally proceed by selecting a set of basis functions ϕ_i, posing $g = \mathrm{sgn}\left(\sum a_i \phi_i\right)$, and estimating the coefficients a_i. The size of the smallest basis that can represent all the functions in $\mathcal{F}$ is known as the *dimension complexity*. The running time of the learning algorithms is polynomial in the dimension complexity.

The class of functions $\mathcal{F} = \{f_y(x)\}$ has exponential $2^{\Omega(n)}$ dimension complexity, since the dimension complexity identifies with the sign rank of f. Thus it is exponentially hard to learn a function from $\mathcal{F}$. Note that each member of $\mathcal{F}$ is just a CNF formula.

5.3 Circuit lower bounds

Our last application is taken from a paper by Forster, Krause, Lokam, Mubarakzjanov, Schmitt and Simon. Suppose that $\varphi(x,y)$ is a Boolean function that depends only on $\sum \alpha_i x_i + g(y)$. In the corresponding matrix M_φ, all rows satisfying $\sum \alpha_i x_i = c$ are the same. Thus the rank of M_φ is at most the number of different values that $\sum \alpha_i x_i$ can take.

The functions computed by linear threshold gates are of the form just described, and so if the weights are small, the resulting matrices will have small ranks r_i. If we add an additional linear threshold gate at the top, we get a sign representation of the entire circuit with rank $\sum r_i$. Thus depth-2 linear threshold circuits computing $f(x,y)$ (defined above) are either very big or use very large weights in their bottom-level gates.