CS 229r Information Theory in Computer Science Mar 5, 2019

Lecture 11

Instructor: Madhu Sudan *Scribe: Luke Minton*

1 Administrative

Problem Set 3 is out and will be due Friday, March 15. Madhu will have Extra Office Hours this week from 1 - 3 PM on Friday. Start thinking about the final project; you're welcome to choose anything for the projects as long as it uses information theory. Topics for the final project are forthcoming; start looking for partners if you haven't already so you can work on project during Spring Break.

2 Overview

- Communication Complexity
- Low Bounds for IP
  - Distributional Complexity
  - Discrepancy
  - Spectrum
- Disjointness

3 Communication Complexity

Recall the model - we have two players, Alice and Bob, who are communicating but have private inputs. Alice has input x, Bob has input y, each is an element of $\{0,1\}^n$. We want to compute some function f from $\{0,1\}^{2n} \to Z$. Usually we want to compute boolean outputs, making our $Z = \{0,1\}$, but sometimes it is some index in n so $Z = [n]$; other times we only partially specify the function. We will consider cases where both Alice and Bob have access to shared randomness R as well as private randomness R_A, R_B. (the distinction doesn't seem to be quite important right now, but we will find cases where private randomness is helpful).

Definition 1. *The deterministic communication complexity of f is $CC(f) = min_{\pi s.t.\pi \text{ computes } f}\{\# \text{ bits exchanged}\}$*

We also have notions of random communication complexity: $CC^{Priv}(f)$ if we have private randomness, $CC^{Pub(}f)$ are the communication complexities corresponding to private and public randomness respectively and are defined analogously to $CC(f)$ (the deterministic complexity). There are obvious inequalities:

Proposition 2. $CC^{Pub}(f) \leq CC^{Priv}(f) \leq CC(f)$

Proof. This is clear because each example of a type is also an example of the types on the left. □

However we also have inequalities going the other direction:

Proposition 3. $CC^{Priv}(f) \leq CC^{Pub}(f) + O(\log n)$

Another is:

Proposition 4. $CC(f) \leq 2^{O(CC^{Priv})}$

The first of these is an exercise in the union bound (take n random strings and use Chernov bounds). The second is much simpler, we just communicate the probability of reaching each leaf under the distribution of the private randomness.

Exercise 5. *Show both of these equalities hold when real-valued probabilities are compressed to finite-length decimals*

4 A Brief Aside

4.1 Hamming Distance

Before we examine lower bounds, let's look at some interesting problems. The first of these is the Hamming Distance problem.

Definition 6. *Hamming $Dis_k(x, y) = 1$ if $\Delta(x, y) \leq k$, $Dis_k(x, y) = 0$ if $\Delta(x, y) > k$, where here $\Delta(x, y) = \#\{i|x_i \neq y_i\}$.*

Proposition 7. *The Hamming Distance problem can be solved in $\Theta(k \log k)$ bits with shared randomness.*

Remark This shouldn't be too much of a surprise because EQ is a special case. (set k equals zero). The important thing is that there is no dependence on n.

Moving for a second, there is a similar problem called small set disjointness.

Definition 8. *In the small set disjointness problem, Alice gets a subset $S \subset [n]$, Bob gets $T \subset [n]$ The question is $S \cap T = \emptyset$?*

In general, set disjointness is a hard question and takes $\Omega(n)$ communication. We will deal with this harder question more at a later time. However, today we consider case where size $|S|, |T| \leq k$ for a small k. We both have very small sets and want to know if they are overlapping or not. We have a really nice protocol that runs in $\Theta(k)$ communication. We will show something a bit weaker today:

Proposition 9. *The set disjointness problem has communication complexity at most $\Theta(k \log k)$*

Remark This is interesting because there is no n in the answer. Even though we are dealing with elements of a very large universe, the size of the universe does not play a role.

Proof. The heuristic explanation behind what we are doing here is that we are restricting our elements from the large universe of size n to a much smaller one of size k. We will use randomness to create a really nice hash function, giving $h : [n] \to [k^2]$. We don't care how much randomness we use here, so we use as much as we can! That is, we pick h at random among all possible functions (and this function is publicly agreed on by Alice and Bob).
Remark Why k^2? This is the right number to avoid the birthday paradox. If there were k elements that we want to keep collision free, the k^2 hash functions do so with high probability.

We want to achieve a small error probability, say 1/100. The feature that we want is that $Pr_h[\exists i \neq j \in W s.t. h(i) = h(j) \leq]1/100$. For small set disjointness, just send hash values of the elements, we will then apply to $W = S \cup T$ With high probability this is collision free. If Alice sends hashes of all elements, Bob would be able to see if there is something that matches his elements; if collision, likely they came from the intersecting case. If no collision then obviously they aren't. By deciding to hash everything down, we have elements of size k in universe of size k^2. High probability over the choice of h; that's how we can solve the small set disjointness problems. Again, Alice sends the hashes of her elements, $h(i)_{i \in S}$ to Bob. That is $\approx k \log k$ in complexity; there is a more clever bound in k, but that is a paper. □

Using these insights, we return to Hamming Distance and to If we have universe of size n and our hash function h; we have k^2 buckets B_i such that $B_i = \{b|h(b) = i\}$; so we have elements in up to k^2 buckets. Using her x_i bits, Alice calculates $u_j = \oplus_{b \in B_j} b$, that is the parity of all the bits hashing to each bucket. This is what she sends to Bob. Bob performs the analogous calculations on his y_i bits, say getting v_j.

If $\Delta(x, y) \leq k$, the x_i, y_i differ in $\leq k$ indices which means that $\Delta(u_i, v_i) \leq k$ as well (since each x_i, y_i is hashed to no more than one bucket). Conversely, if $\Delta(u_i, v_i) > k$, then $\Delta(x, y) > k$ with high probability as well, giving us a $\Theta(k^2)$ protocol (this could be improved with binary search).

Exercise 10. *Use a binary search-based protocol to improve our algorithm and prove Proposition 7.*

Solution: The key insight here is that we are being inefficient by sending the hashes of ALL k^2 buckets; if we send over fewer hashes, but randomly select the buckets that Alice sends over (but in a way that is also known to Bob), then for a suitably large selection we can also predict the number of differing indices in this selection as well. Specifically, Alice randomly computes a $3\log(k)$ length string (remembering that private randomness is a subset of public randomness); the first $2\log(k)$ bits encodes an integer i, defining a cyclic ordering of the k^2 buckets starting with the ith bucket. Then, grouping the k^2 buckets into k blocks of k buckets each, the next $\log(k)$ bits defines a series of "moves" following a binary search pattern, that determines which blocks are to be selected. For example if the first three bits are $(0, 1, 0)$, Alice begins by selecting block $k/2$, discarding the right half of the remaining blocks and selecting block $k/4$ (corresponding to the first 0), discarding the left half of the remaining blocks and selecting block $3k/8$ (corresponding to the first 1), and then discarding the right half of the remaining blocks and selecting block $5k/16$. Alice then sends over the hash values for each of the k buckets in each of the $\log(k)$ selected blocks, as well as her $3\log(k)$ length random string, for total message length $(k+3)\log(k) = O(k \log k)$. In the case where (x, y) differ in more than k entries, analysis of the Hypergeometric Distribution would tell us that we would expect at least $k(k \log k)/k^2 = \log(k)$ entries to differ among those selected by Alice. Moreover by analyzing the Hypergeometric CDF, as k becomes larger, with high probability the selected u_i, v_i differ in no more than $\log(k)$ with high probability if (x, y) differ in no more than k indices. By similar logic, if (x, y) differ in more than k indices, the selected u_i, v_i will differ in at least $\log(k)$ indices with high probability (the reason for this is that for large k, the Hypergeometric Distribution$(k^2, x, k \log k)$ converges in distribution to the Binomial Distribution $(k \log k, x/k^2)$; as k becomes increasingly large we get two distinct Chernoff bounds for the cases $x > k$ and $x \leq k$ respectively).

4.2 Squared Distance

All of the problems we have considered, and most of the problems we usually consider, use Boolean functions. Therefore, let's describe at least one case where we have real numbers as inputs:
Alice gets $x \in \mathbb{R}^n$,$||x||_2 = 1$ Bob gets $y \in \mathbb{R}^n$,$||y||_2 = 1$.
Remark These norm constraints aren't used directly in our proof, but the constraint allows us to have a well-defined sense of the relative error bound.

Let $f(x, y) = \sum x_i - y_i \pm \varepsilon$. Alice discretizes all her bits (so we aren't sending an infinite number of bits) and sends $(\sum_i x_i) \pm \varepsilon$ it to him. This should take around $\log(1/\varepsilon)$ bits of communication (as we get ever more precise, we require more bits). This is quite easy, so let's consider a harder problem.

Definition 11. *In the Squared Distance problem, given x and y as defined above, we want to compute $\sum_i (x_i - y_i)^2 \pm \varepsilon$ (this is just the squared Euclidean distance between two points).*

This is less trivial - clearly we do not want to send the entire x; that's too much communication.

Proposition 12. *The Squared Distance problem can be solved with CC $\Omega(1/\varepsilon)$*

Proof. Let's pick n random "bits" $R_1 ... R_n$, representing the R_i not as elements of $\{0, 1\}$ but rather as elements of $\{-1, 1\}$ (we'll see this useful technique more below), and Alice will send to Bob $(\sum (x_i)^2) \pm \varepsilon$

Remark (we're sending discretization, so $\pm\varepsilon$, this isn't the big cost here).

Alice will send $\sum_{R_i} x_i^2$. Now, if we consider $\sum_i (x_i - y_i)^2$ and expand this sum into $\sum_i x_i^2 - 2x_iy_i + y_i^2$, we note that Alice is sending the left-hand term and Bob will already have the right-hand term. Our claim is going to be the following:

Lemma 13. $E_R((\sum_{R_i} x_i \sum_{R_j} y_j)) = \sum(x_iy_i)$

This will allow Bob to finish calculating the sum.

Proof. $E_R((\sum_{R_i} x_i \sum_{R_j} y_j)) = E_R(\sum_i R_i^2 x_iy_i) + E_R(\sum_{i\neq j} R_iR_jx_iy_j)$ The same terms just evaluate to $\sum_i x_iy_i$ since R_i^2 is always 1; conversely, the cross terms $E_R(\sum_{i\neq j} R_iR_jx_iy_j)$ sum to 0 by symmetry. $\square$

Therefore, our protocol should give us a bound of $1/\varepsilon$ for our communication complexity. $\square$

Exercise 14. *Use the disjointness lower bound (Lecture 12) to prove the* $1/\varepsilon$ *lower bound.*

Solution: Assume we have a $O(1/\varepsilon)$ protocol to solve the Squared Distance problem. We consider the set $[N]$, and two subsets for which we want to calculate the answer for the set-disjointness problem. Alice transforms her subset A into a $2N$ dimensional vector, with the $2i-1, 2i$th entries equal to $(1/\sqrt{N}, 0)$ if $i \in A$ and $(\frac{1}{2\sqrt{N}}, \frac{\sqrt{3}}{2\sqrt{N}})$ otherwise. Bob does the same, except he encodes $((\frac{1}{2\sqrt{N}}, -\frac{\sqrt{3}}{2\sqrt{N}})$ if $i \notin B$. Alice sends over her vector, and using the protocol, we calculate the squared Euclidean distance. We calculate with additive error no more than $\varepsilon = \frac{3}{8N}$, noting this takes $O(1/\varepsilon) = O(N)$ bits. We note that for every i such that $i \in A, i \in B$, there is a contribution of 0 to the squared distance sum, else there is a contribution of $\frac{3}{4N}$. If our calculated squared distance is greater than $\frac{3}{4} - \frac{3}{8N}$, we note this can only arise if there are no indices i such that $i \in A, i \in B$, so this protocol solves the set disjointness problem in $O(N)$ bits.

It is actually possible to show a stronger bound of $1/\varepsilon^2$, but this requires Gap Hamming, which is not something that will be covered in the course.

5 Inner Product (IP) Lower Bound

We consider the inner product function: $IP(x,y), x, y \in \{0,1\}^n$.

Definition 15. *The inner product* $IP(x,y)$ *is defined as*

$$IP(x,y) = \bigoplus_i x_iy_i \ (mod\ 2)$$

Here is a place where we believe randomness doesn't really help us. Why is this? We have seen in the past that the rank method sometimes doesn't help us in randomized contexts. EQ is a perfect example of this. We seek to prove the following:

Theorem 16. *A lower bound for communication complexity for the Inner Product problem is* $\Omega(n)$

We will need a more "robust" metric in order to help us do so. We will want to do this with public randomness. Proving the lower bound on this is not trivial, so here we have a few ideas at play that help us out:

Remark In principle our lower bounds could be worst case lower bounds, (selecting a pathological x, y). A **distributional lower bound** specifies a distribution; i.e. we specify a distribution and find that nothing works well. We specify distribution μ over x, y and claim that no deterministic protocol π achieves small error on $(x,y) \sim \mu$.

Definition 17. *The error on a distribution is* $\delta_m(f,g) = Pr_{x,y\sim\mu}[f(x,y) \neq g(x,y)]$.

Lemma 18. *The claim that no deterministic protocol achieves error ε on $(x, y) \sim \mu$ is equivalent to $\delta_\mu(f, \pi) > \varepsilon$.*

Proof. To do this, let's try to illuminate the link between distributional lower bounds over deterministic protocols and lower bounds over randomized protocols. Suppose our protocol π does very well on every x, y and gets error less than 1/2, say 1/3. If we have something that achieves that on every x, y, then we can repeat protocol $\log(1/\varepsilon)$ times and reduce the error. So let's call our protocol a k bit communication protocol. There exists a $O(k \log(1/\varepsilon))$ bit randomized protocol that gets error less than ε on every x, y. So that's simple; so now we have protocols that make small error (but are still making error).

To make the connection, we consider distribution μ, so if it's true for every x, y, $E_R[\pi_{f(x,) \neq pi'(x,y,R)}] < \varepsilon$, where R is the randomization we are using in our protocol and $pi'(x, y, R)$ is what our randomized protocol is once the randomization has been specified. This is true for every x, y, but we can put this over any expectation over (x, y) - we just sum up over the relevant probabilities.

Therefore, we have $E_{x,y}E_R[\pi_{f(x,) \neq pi'(x,y,R)}] < \varepsilon$; but we can exchange the $E_{x,y}$ and the E_R. And the resulting gives us in the inner term $E_{x,y}[\delta(f(\pi_r))]$ π_r is a protocol with the randomness hardwired to $R = r$. This means there's an analogous deterministic protocol which achieves the ε (or else the randomized protocol wouldn't be able to achieve the bound when in expectation over R). □

Remark This makes us realize an important connection: A randomized protocol is just a distribution over deterministic protocols. Therefore there exists an r such that the protocol π_r satisfies $\delta_{x,y \sim \mu}(\pi_r) < \varepsilon$ This allows us to go to deterministic protocols. In some sense, picking the correct distribution may be challenging; but, the fact remains that so far this has been helpful in many cases to find lower bounds, and it is what we will do here.

Definition 19. *The distributional complexity of computing f with error ε over inputs from μ is $D_{\mu,\varepsilon}(f) = \min_{\pi_i}(\#$ bits exchanged by $\pi_i)$ where π_i is in the set of protocols transmitting no more than k bits.*

Capitalizing on the previous lemma, we want to prove an equivalent statement to Theorem 16:

Corollary 20. $D_{\mu,\varepsilon}(IP) = \Omega(n)$

Proof. What distribution do we want to deal with? The thing that makes the problem so easy is the fact that we can just use as μ the uniform distribution, $1/4^n$ in each entry corresponding to the x, y (there are 2^n possible strings each for Alice and Bob so 2^{2n} possible combinations for the inputs). First time we looked at deterministic protocols, we saw how communication of bits separates our matrix into monochromatic pieces; the final pieces that we see should have all values either 0 or 1.
Remark WLOG the final bit communicated by π is the value of the function. (this adds no more than one round of communication, so we won't care about that much difference today).

If we consider our protocol, what happens to the matrix? This matrix is going to get partitioned into smaller pieces. After a while, we are left with a bunch of small rectangles; the protocol is constant within each rectangle. Lets label these rectangles $R_1..R_K$ (remember when we say rectangles, we do not mean necessarily, geometrically contiguous, just that they satisfy the definition of rectangles as the Cartesian product of two sets). Some fraction of rectangles has correct values; some fraction has errors.

So let's say we measure everything in absolute numbers: $p_1...p_K$ are correct fractions in rectangles $\varepsilon_1...\varepsilon_k$ are error fraction in rectangles. We easily arrive at $\sum_i \varepsilon_i \leq \varepsilon$. $\sum p_i \geq 1 - \varepsilon$ This means $\sum p_i - e_i \geq 1 - 2\varepsilon$. For the K rectangles, there must be at least one rectangle which at least achieves the average, so there is an R_i s.t. $p_i - \varepsilon_i \geq \frac{1-2\varepsilon}{K}$ or $p_i - \varepsilon \geq \frac{1-2\varepsilon}{2^k}$ since there are 2^k rectangles (remember, this lowercase k is the number of bits communicated).

When we consider distributional complexity, it's helpful to talk about a slightly different matrix than the one we usually use (similar to what we used above for the Squared Distance problem). Confer last lecture, $f \to M_f$ s.t. that $M_{xy} = 0$ if $f(x, y) = 0$ and $M_{xy} = 1$ if $f(x, y) = 1$. Instead, we make our new matrix $M_{xy} = \mu(x, y)$ if $f(x, y) = 0$ and $M_{xy} = -\mu(x, y)$ if $f(x, y) = 1$ according to the distribution we have picked.

Definition 21. *The Discrepancy of f is defined as $Disc(f) = \max_{S,T} \mid \sum_{x\in S,y\in T} M_{x,y} \mid$, where M is our modified matrix.*

Remark When we use our modified matrix, the Discrepancy is achieved for some S and T that induce a large monochromatic rectangle; why is this?

In the 0 error case, within each rectangle we have a constant value, so the absolute value of the sum of the $M_{x,y}$ is just the appropriate probability mass. Therefore must be at least one with 2^{-k} value (one must be at least as great as average). When we introduce ε error, we can analogously assert

Lemma 22. *There exist S and T such that $| \sum_{x\in S,y\in T} M_{xy}| \geq \frac{1-2\varepsilon}{2^k}$*

Proof. The sum $\mid \sum_{ij} M_{ij} \mid$ with i, j over the entire matrix is $1 - 2\varepsilon$, and there are 2^k rectangles each of which is monochromatic, so at least one must achieve the average of $\frac{1-2\varepsilon}{2^k}$. □

Applying this fact and the definitions of distributional complexity and discrepancy, we note the following

Proposition 23. *$D_{\mu,\varepsilon}(t) \leq k \implies Disc_\mu(f) \geq (1 - 2\varepsilon)/2^k$*

We want some methods to upper bound the discrepancy (which then gives lower bound on k). This intuitively makes sense, for example EQ is a terrible function - large block of zeros are all constant, which is why we don't get a good bound. So we're going to try to make this a bit more linearly algebraic.

From our Discrepancy definition, we recall that $Disc(f) = \max_{S,T} \mid \sum_{x\in S,y\in T} M_{x,y} \mid$ however we can re-interpret summing over the relevant indices in S, T as applying indicator linear operators I_S, I_P (that is, coordinate i is 1 iff $i \in S$) This gives us $\max_{U,V} U^\perp MV$ with $U = I_S, V = I_P$.

We have coordinate restrictions here (U,V can only have 0 or 1 at each index), and these are strange in linear algebra. However, norm restrictions are not strange! So we will expand the set, noting that $\max ||U||_2, ||V||_2 \leq 2^{n/2}$ (when all coordinates are included). so we can scale these vectors down and have this is $2^n \max_{||U||_2,||V||_2=1} U^\perp MV = 2^n\lambda_{max}(M)$. We need this eigenvalue to be small, or else this doesn't really help us. We can show inductively that $M_{IP_N,\mu_N} = (M_{IP_1,\mu_1})^{\otimes N}$

Proposition 24. *The matrix of M_{IP_1,P_1} is*

$$\begin{bmatrix} 1/4 & 1/4 \\ 1/4 & -1/4 \end{bmatrix}$$

Exercise 25. *Prove Proposition 24. Then, show $M_{IP_N,\mu_N} = (M_{IP_1,\mu_1})^{\otimes N}$ using induction.*

Solution: For $n = 1$ This is a linear map over $\mathbb{R}^2$, with 0 bits corresponding to the first standard basis vector $(1, 0)$ and 1 bits corresponding to the second standard basis vector $(0, 1)$. The IP of two bits is 1 (which is mapped to $-\mu(x, y)$) iff both bits are 1. Alternatively, the IP is 0 (which is mapped to $\mu(x, y)$ for any other case. Because we are using the uniform distribution, $\mu(x, y) = 1/4$ when $x, y \in \{0, 1\}$, so this gives us the matrix form for M_{IP_1,μ_1}. Clearly the base case for induction holds, when $n = 1$. We claim that $M_{IP_N,\mu_N} = (M_{IP_1,\mu_1})^{\otimes N}$ and we consider what the form of $M_{IP_{N+1},\mu_{N+1}}$ is. We note that if two maps agree on the images of all of our basis vectors, then the two maps must be equivalent. We note that $M_{IP_{N+1},\mu_{N+1}}$ is a 2^{N+1} dimensional vector space, with a basis vector corresponding to each length $N + 1$ binary string. We define some bijection between $N + 1$ length strings and the integers in the set $[2^{N+1}]$. We note if we specify a length N string a, there are exactly four matrix entries whose rows/columns have a as a prefix:

$(a0, a0), (a0, a1), (a1, a0), (a1, a1)$. From the bilinear properties of the Inner product, $IP(a1, a1)$ must have the opposite sign as $IP(a, a)$, and the other three must have the same sign. Because of this bilinearity, we can express M as the tensor product of two maps. Then, we can write all of our basis vectors in the form $a \otimes x$, where $x \in \{0, 1\}$. The form of M on this vector space is $M_{IP_{N+1},\mu_{N+1}} \otimes F$ for some $F : \mathbb{R}^2 \to \mathbb{R}$. Moreover, this function must map to $\mu(x, y)$ if the Inner product is 0 and $-\mu(x, y)$ if the inner product is 1, because of what we just showed. Additionally, by the properties of the tensor product on linear maps and the induction hypothesis, $\mu_{N+1}(x, y) = \mu_N(x, y) \times \mu_1(x, y)$, for the uniform distribution over binary strings of length N. That means that F must equal $1/4$ when the inner product is zero and $-1/4$ otherwise. This clearly shows that F must just have the form M_{IP_1,μ_1} that we desire to show, and we are done.

Simple linear algebra shows us that $\lambda_{max}(M_1) = \pm 1/\sqrt{8}$. which gives us that $\lambda_{max}(M_N) = (1/8)^n/2 = \frac{1}{2^n}\frac{1}{2^{n/2}}$. Returning to Proposition 25, we have $Disc_\mu(f) \leq (1 - 2\varepsilon)/2^k \implies D_{\mu,\varepsilon} \geq k$ and our discrepancy is bound above by $2^n \frac{1}{2^n}\frac{1}{2^{n/2}} = 2^{-n/2}$ If we let $\varepsilon = 1/4$ and $k = n/2-1$ we have that $Disc_\mu f \leq (1-2\varepsilon)/2^k = \frac{1}{2^{n/2}}$ which implies that $D_{\mu,\varepsilon} \geq k = n/2 - 1$ Therefore, since a deterministic algorithm is $\Omega(n)$ for any μ, then $CC^{Pub}(IP) = \Omega(n)$ by Lemma 18.

$\square$