Randomized Algorithms 2022-3
Lecture 3

Existential Proofs via Probabilistic Constructions, Large Deviation Bounds and Communication Complexity.

Moni Naor

1 Large Deviations

We discussed some bounds on large deviations that are useful when employing randomized algorithms and specifically the probabilistic method. We based the discussion on Appendix A of Alon-Spencer [1].

For instance, Corollary A.1.14 there is often useful in our settings:

Theorem 1. *Let $X_1, X_2, \ldots, X_n$ be* **indicator random variables** *(i.e. there are either o or 1 indicating whether an events happened and suppose that they are mutually independent Let $Y = \sum_{i=1}^{n} X_i$ be the sum of the indicators and let $E[Y] = \mu$. For all $\varepsilon > 0$ there is a c_ε where*

$$\Pr[|Y - \mu| \geq \varepsilon\mu] < 2e^{c_\varepsilon \mu}.$$

Note that c_ε depends only on ε and the value of n does not appear in the bound, just the the value of the μ.

We used this for the 100 prisoners puzzle, to combat a malicious warden who is aware of the prisoners strategy and selects a bad permutation. We said that selecting a random permutation by the prisoners can overcome such a bad initial one. But can we select the permutation from a smaller set? So that its distribution and representation will be more tractable,

We can show the existence of a family of permutations Γ of size $O(n \log n)$ that is almost as a good as a truly random permutation. This is a case of a probabilistic construction. *It is an open problem to come up with an explicit construction of such a family.*

Question: consider the family of pair-wise independent permutations $F = \{f_{a,b} | f_{a,b}(x) = ax + b \bmod panda \neq 0\}$ (here we assume that $n = p$ is prime.) What can you say about it as a construction for the prisoners?

The same type of proof also shows that for any decision problem in the class BPP - where the probability of error is bounded by 1/3 - there exists a an equivalent deterministic algorithm with a fixed advice. Specifically, for any BPP algorithm A for inputs of size n there is a family of assignments

to the random string $R_{A,n}$ of polynomial size so that if one runs the original randomized algorithm with all strings in $R_{A,n}$ and takes the majority of the decision, then the result is guaranteed to be correct.

This puts the class BPP inside $P \setminus Poly$: the class of polynomial time algorithms that take advice that depends on the size of the problem and nothing else (sometimes called non-uniform P).

You can read about *Turing Machines that take advice* and the class $P \setminus Poly$ in Arora and Barak [2] Chapter 6.3 [1].

2 Communication Complexity

Let $f : 0, 1^n \times \{0,1\}^n \mapsto \{0,1\}$. Communication complexity studies the length of the messages that two parties A and B need to exchange in order to compute $f(x, y)$ where x is the input of A and y is the input of B. It is a very well studied topic with a rich structure. There are a couple textbooks such as Kushilevitz-Nisan [3] and the More recent Rao-Yehudayoff [5].

Understanding communication complexity protocols is key to understanding many models and issues such as VLSI and Streaming algorithms.

We saw an non-trivial example of a deterministic protocol. There is some underlying graph $G(V, E)$ on $n = |V|$ nodes (fixed and known to both sides). The inputs to the two parties are subsets of nodes. One party get a clique C and the other gets and independent I. Clearly the largest size of the intersection is is 1. They need to determine whether the two sets intersect or not. There is an $O(log^2 n)$ bits protocol for the problem, where the two parties send to each other low nad high degree nodes respectively. Each such rounds halves the number of potential nodes that could be in the intersection.

Regarding the equality function, where Alice gets $x \in \{0,1\}^n$ and Bob gets $y \in \{0,1\}^n$ and they need to decide whether $x = y$, we argued that any deterministic protocol requires n bits of communication. As for randomized protocols, here the question is whether there is **shared randomness** between the parties or **private randomness**. In both case the strategy is to consider some hash function $h \in_R H$ where the probability of collision between two different strings is roughly the size of the range of h.

In the shared randomness case the function h is defined by the shared random string. In the private case one of the parties needs to select it and send it to the other one, so the description of h, i.e. $\log |H|$ is important.

An example of a good function for the shared randomness case is the inner product function $h_r(x) = \sum_{i=1}^{n} x_i r_i \bmod 2$ where $r \in_R \in \{0,1\}^n$.

Claim 2. *For any $x, y \in \{0,1\}^n$ where $x \neq y$ we have that*

$$\Pr[\sum_{i=1}^{n} x_i r_i \bmod 2 = \sum_{i=1}^{n} y_i r_i \bmod 2] = 1/2$$

[1] See `https://books.google.co.il/books?id=8Wjqvsoo48MC`

where the probability is over the choice of r.

In the private randomness case we need a small family of functions. We can get such a family using small-bias probability spaces [4] as substitute for the above family or using a family based on polynomial evaluation, similar in nature to the one we saw for Multiset equality last time.

We considered the *simultaneous message model* for evaluating a function $f(x, y)$: Alice and Bob share a random string. They receive inputs x and y respectively and each should send a message to a referee, Charlie, who should evaluate the function $f(x, y)$. They may also have their own private source of randomness. The goal is for Alice and Bob to send short messages to Charlie.

We considered the equality function in this model.

1. When Alice and Bob share a random string. the protocol where Alice and Bob send to Charlie an inner product of their input with a common random string r and Charlie decides based on the equality of the messages he receives. What happens to this protocol if Eve, who selects the inputs and whose goal is to make Charlie compute the wrong value, knows the common string r when she selects x and y?

2. We suggested a non-trivial protocol for the case without shared randomness, one whose communication complexity is sublinear in the input length.

   The idea is to start with a good error correcting code $C(x)$ (one that expands the input by some constant and one where the relative distance is constant as well). Now Alice take $C(x)$ and arranges in a square matrix of size $O(\sqrt{n}) \times O(\sqrt{n})$ (and similarly Bob does for y). Now Alice selects a random index i and sends i and the corresponding row. Bob selects a random column j and sends it and the column. The intersection, entry (i, j) in the corresponding matrices, is a random letter of the code. Therefore, if $x \neq y$, there is some constant probability that the the values in location (i, j) will differ in the two matrices.

   The complexity is $\Theta(\sqrt{n})$ in the

Without shared randomness the complexity is $\Omega(\sqrt{n})$ in the simultaneous message model and we started discussign the Babai-Kimmel proof of that.

See the lectures by O'Donnel on the topic:

- Lectures 23a-d of CS Theory Toolkit `https://www.youtube.com/watch?v=mQQ36cDnmR8`

Proof by Compression: a quite general method to prove success of an algorithm by showing that failure allows us to compress the random bits used. We know that for any method the probability of compressing and chopping off w bits from a random string is 2^{-w}.

Example: the "birthday paradox". Suppose that you have k random elements $x_1, x_2, \ldots x_k$ from a domain of size n. When, can you expect collision, i.e. for what value of k as a function of n. We note that if $x_i = x_j$, then it is possible can encode x_j by pointing out to i (as is done in the Lempel-Ziv family of compression algorithms). In such a case, instead of using $\log n$ bits to encode

x_j we need only $\log k + \log k$ bits. This saves (i.e. compresses) when $k < \sqrt{n}$, so we conclude that it is not likely to happen before $k \geq \sqrt{n}$.

Homework. Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$ be a random function. Prove via compression that any algorithm that has **black-box access** to f and receives $y \in \{0,1\}^n$ and tries to find $x \in \{0,1\}^n$ s.t. $f(x) = y$ must access f close to 2^n times.

As we will see in future lectures, the method can be used to prove the run time of the algorithm in the "Algorithmic Lovasz Local Lemma" and the success probability of Cuckoo Hashing.

References

[1] Noga Alon and Joel Spencer, **The Probabilistic Method**, Wiley, 2008.

[2] Sanjeev Arora and Boaz Barak **Computation Complexity: A Modern Approach**

[3] Eyal Kushilevitz and Noam Nisan, **Communication Complexity**, Cambrdige, 1996.

[4] J. Naor and M. Naor, *Small-Bias Probability Spaces: Efficient Constructions and Applications*, SIAM J. on Computing, 1995.

[5] Anup Rao and Amir Yehudayoff, **Communication Complexity and Applications**, Cambridge 2020.

[6] Eugene Curtin and Max Warshauer, The Locker Puzzle, Mathematical Inteligenicer. `https://www.cl.cam.ac.uk/~gw104/Locker_Puzzle.pdf`