Randomized Algorithms 2024-5
Lecture 3

Large Deviation Bounds, Communication Complexity and Existential Proofs via Probabilistic Construction

Moni Naor

1 Large Deviations

The famous concentration bounds are Markov, Chebychev's and Hoefding. We discussed some bounds on large deviations that are useful when employing randomized algorithms and specifically the probabilistic method. One good source is Appendix A of Alon-Spencer [1].

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

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

One thing we mentioned in passing is via a probabilistic existential proof you can put 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 Multiset Equality and Memory Checking

We saw an application for the multiset equality problem to memory checking where a processor with a small memory tries to authenticate the actions of an untrusted memory. In this setting, a processor has small secret memory and access to a large memory of size n and the goal is to discover whether the memory malfunctions. The memory malfunctioning means that it does not return the correct value that was last stored at a given location (i.e/ the last value stored at that

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

location). There are no assumptions on the memory's behavior and it might answer in any way it wishes when executing a read operation. The processor's goal is to check that all read operations retrieve what was last written in the location read. It should not declare 'faulty' if the memory functions properly. There are several variants to this problem. We discussed what is known as the offline problem, where the identification that an error occurred happens after the computation has ended.

The transformation considered in class was to add a timestamp field to each address in memory; whenever writing a value to an address add the current-time. This creates READ and WRITE sets where the elements are of the form (v, i, t) where v is the content, i is the address and t is the time. One condition to add to the processor's functionality is that it reads an address i and receives the v and the content and t as the time (presumably when it was written) the processor compares t to the current time and makes sure it is smaller.

In more detail, the transformation involves:

- Scan all memory cells before the beginning of the sequence of operations and write in each cell a value '0' with time = 0.
- After each 'read' operation a 'write' operation is performed with *current_time* as the timestamp.
- Before each 'write' operation to location a, a 'read' operation to location a is performed.
- At the end of the sequence of operations, the memory is scanned again and all cells are read.

Whenever a 'read' operation is performed and a timestamp t is returned, verify that $t < \mathit{current\_time}$ (otherwise it is a clear malfunction of the memory). Call the property that this is always the case the **monotone property**:

Consider the following two sets:

$$R = \{(v, i, t) | \text{location } i \text{ was read with value } v \text{ and timestamp } t\}$$

and

$$W = \{(v, i, t) | \text{location } i \text{ was written with value } v \text{ and timestamp } t\}$$

The algorithm suggested in class tested whether these two sets are equal and that the monotone property holds. Prove the following: Question: prove the claim

Claim 2. *If the monotone property holds, then $W = R$ iff the memory functioned properly.*

Hint: the reduction works is that since time grows, if at any point the read is wrong, its timestamp is smaller than the current time and the tuple will never be inserted into the write set, since the time only grows.

3 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 of textbooks such as Kushilevitz-Nisan [6] and the More recent Rao-Yehudayoff [9].

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

We saw a 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 Alice gets a clique C (i.e. a set of nodes in V that induce a complete graph) and the other party Bob gets an independent set $I \subset V$. Clearly the largest size of the intersection between C and I is 1. Their goal is 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 and high-degree nodes respectively. Each such round halves the number of potential nodes that could be in the intersection. A major result is that this is the best possible: Goos, Pitassi and Watson [4]

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. This was via a covering rectangle argument.

As for randomized protocols, here the question is whether there is **shared randomness** between the parties or **private randomness**. In both cases, 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 randomness case, one of the parties needs to select the hash function and send it to the other one, so the length of 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 3. *For any $x, y \in \{0,1\}^n$ where $x \neq y$ we have that*

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

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 [7] as a 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. In the original paper that formally introduced communication complexity in 1979[2], Yao [10] mentioned that in the private coins case the parties must send $\Omega(\log n)$ bits. Different proofs for this statement can be found in either Hastad and Wigderson [5] or Ben-Sasson and Maor [3].

[2]The paper is four and a half pages long

We saw an existential result, that for any protocol in the shared randomness setting, there is a small ($O(n)$) set of random strings which the two parties can sample from. This implies that in the private randomness setting with additional round and $O(logn)$ bits of communication we can have the same power as the shared randomness model. See Newman [8].

The proof was based on a probabilistic construction of a collection of random strings, where one argues that for each pair $(x, y$ the probability that for at least 2/3 of the strings in the collection the protocol yields the correct result on (x, y) with probability larger than $1 - 1/2^{2n}$. This means, from a **union bound** over bad events, that there is some collection that is *simultaneously good for all input pairs.*

We discussed the disjointness problem, where the two parties each receive a subset $S_A \subseteq U$ and $S_B \subseteq U$ and the goal is to determine whether the two subsets intersect (whether $S_A \cap S_B = \phi$. This is an important problem in the area and, in general, the bound is $\Theta(n)$ where $n = |U|$ (we did not see the proof, you can find it in Chapter 6 of Rao and Yehudayoff). For the case where $|S_A| = |S_B| = k$, k-disjointness we saw an O(k) randomized algorithm due to Hastad and Wigderson [5].

Question: We said in class that using various data structures, in particular Bloom filters, it is possible to make this protocol also computationally efficient and consuming fewer bits. Try to formalize this.

Finally, we mentioned 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 and said that the original protocol works here. What happens when Alice and Bob do not share randomness? Can you think of a non-trivial protocol for this case?

Question (I don't know the full answer): is there a $poly(k)$ protocol for k-disjointness in this model (with shared randomness)? Yes, see homework.

4 Card Guessing

The scenario we are considering is where a deck of n distinct cards (for simplicity labeled $1, 2, \ldots, n$) is shuffled and the cards from the deck are drawn one by one. A player called 'guesser' tries to guess the next card, for n rounds and receives a point for each correct guess. We are interested in the expected number of points the guesser can have.

Suppose that the guesser has *perfect memory* and can recall all the cards that it has seen, then what is the expected number of correct guesses? At any point, the guesser picks one of the cards that have not appeared so far as a guess. If there are i cards left, the probability of guessing correctly is $1/i$ and the expected number of guesses is

$$1 + \frac{1}{2} + \frac{1}{3} \ldots + \frac{1}{n} = H_n \approx \ln n.$$

Note that any guess of an unseen card has the same probability of success, so there is really not much of a strategy here.

Question: What can you say about high concentration in this case?

Now consider the opposite scenario, where the guesser has no memory at all. I.e. before it turns over a card it has no idea what cards have already appeared. But we will give it for free the round number. So the best strategy it may have is represented by a fixed guess g_i for the round i. The probability that this is correct is $1/n$, so the expectation over all n rounds is 1.

See the lectures by O'Donnel on the topic:

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

References

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

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

[3] Eli Ben-Sasson and Gal Maor, *Lower bound for communication complexity with no public randomness*, ECCC TR15-139, 2015.

[4] Mika Goos, Toniann Pitassi, and Thomas Watson *Deterministic Communication vs. Partition Number*, Siam Journal on Computing 2018.

[5] Johan Hastad amd Avi Wigderson, The Randomized Communication Complexity of Set Disjointness, Theory of Computing 2007.

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

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

[8] Ilan Newman: Private vs. Common Random Bits in Communication Complexity. Inf. Process. Lett. 39(2): 67-71 (1991).

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

[10] Andrew Chi-Chih Yao, *Some Complexity Questions Related to Distributive Computing*, Proc. Of 11th STOC, pp. 209-–213, 1979,