CS 761: Randomized Algorithms Fall 2019

Problem Set 1

Prof. Gautam Kamath *Deadline: 11:59 PM on October 7, 2019*

You are allowed to discuss the problems in small groups (2-4 people). List your collaborators for each problem. Every person must write up and submit their own solutions.

1. **Boosting Success Probability.** Suppose that we have an algorithm A, which takes a dataset $X^{(1)} \sim D$ and outputs a real number with the following guarantee, with respect to some (unknown) value p:
$$\Pr[|A(X^{(1)}) - p| \leq \varepsilon] \geq 3/4.$$
That is, the algorithm is "accurate" with probability at least $3/4$, where the probability is over the sampling of $X^{(1)} \sim D$ and the randomness in the algorithm A. Using this algorithm as a black box, give an algorithm A' which boosts this success probability to $1 - \delta$ using $O(\log(1/\delta))$ independent repetitions of the algorithm.
$$\Pr[|A'(X^{(1)}, \ldots, X^{(O(\log(1/\delta)))}) - p| \leq \varepsilon] \geq 1 - \delta.$$
Assume that we can draw $O(\log(1/\delta))$ additional (independent) datasets from D.
This technique is useful when we may have a learning/estimation algorithm which is correct with probability strictly greater than $1/2$, and we wish to boost the success probability to be arbitrarily high.
Optional: Extend your to the vector-valued setting. Suppose that the output of A and p are d-dimensional vectors, and we have the following guarantee:
$$\Pr[\|A(X^{(1)}) - p\|_2 \leq \varepsilon] \geq 3/4.$$
Use this to design an algorithm A' which takes $O(\log(1/\delta))$ independent datasets from D and has the following guarantees:
$$\Pr[\|A'(X^{(1)}, \ldots, X^{(O(\log(1/\delta)))}) - p\|_2 \leq 2\varepsilon] \geq 1 - \delta.$$
Note that we allow an additional factor of 2 in the approximation guarantee.

2. **High-Probability Quicksort.** In class, we proved that the expected running time of randomized Quicksort is $O(n \log n)$. Prove that the running time of randomized Quicksort is $O(n \log n)$ with probability at least $1 - 1/n$.

3. **Sequential Selection.** You are on a game show with the following rules. There will be n time steps, and at the ith time step, the following occurs. You will be offered a dollar amount v_i. You can choose either to accept the prize v_i and the game ends, or irrevocably reject the prize and the game continues to time step $i + 1$. Assume that the values are all unique, and are presented in a uniformly random order.
One could play this game using the following strategy. Reject the first m dollar amounts $v_1, \ldots, v_m$. After the mth time step, accept the first value v_i which is greater than all the previously seen values.

Let E be the event that you accept the largest prize. Let E_i be the event that the ith prize is the largest one and that you accept it. Compute $\Pr(E_i)$ and show that $\Pr(E) = \frac{m}{n}\sum_{i=m+1}^{n}\frac{1}{i-1}$. Show that, with an appropriate choice of m, this probability can get arbitrarily close to $1/e$.

4. **Chernoff Bound.** Let X be a standard normal random variable, with probability density function $f(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right)$.

   (a) Compute the moment generating function of X^2, $M_{X^2}(t) = E\left[\exp\left(tX^2\right)\right]$. You may use the fact that $\int_{-\infty}^{\infty} f(x)dx = 1$.

   (b) Compute $E[X^4]$, potentially using your result from the previous part.

   (c) Let $X_1, \ldots, X_n$ be independent standard normal random variables, and $Z = \frac{1}{n}\sum_{i=1}^{n} X_i^2$. Use the Chernoff method to prove that $\Pr(Z \geq 1+\varepsilon) \leq \exp\left(-n\varepsilon^2/8\right)$ for $0 \leq \varepsilon \leq 1$. You may use the Taylor expansion $\ln(1-x) = -\sum_{i=1}^{\infty} x^i/i$ for $-1 \leq x \leq 1$.

5. ***s*-*t* min-cut.** Consider the s-t min-cut problem, in which we are given an undirected graph G, where two vertices s and t are specified. An s-t min-cut is a set of edges (of minimum cardinality) whose removal disconnects s from t. We try solving this problem using the contraction algorithm. As the algorithm proceeds, if s (respectively t) get merged with another node, the resulting merged node becomes s (respectively t). We make sure to never contract an edge between s and t.

   (a) Show that there are simple graphs (i.e., not multi-graphs) in which the probability that this algorithm finds an s-t min-cut is exponentially small.

   (b) Asymptotically, how many s-t min-cuts can an instance of the s-t min-cut problem have?

6. **Second min-cut.** Consider the problem of finding the second smallest cut in a graph. This may be equal to the min-cut, if there are two min cuts, or it might be much larger. Show that a modification to a single run of the contraction algorithm as an $\Omega(1/n^2)$ chance of finding the second smallest cut.

7. **Balls and bins in rounds.** Suppose we have n jobs and n machines. Each machine can process 1 job at each time step. At time 0, we assign the n jobs uniformly at random to the n machines. If this is all we do, we know from class that it will take $\Theta(\log n/\log\log n)$ time steps until all jobs are processed. Instead, at each time step, we will assign all incomplete jobs uniformly at random to the n machines.

   (a) Argue that if αn jobs begin a round, then with high probability only $(\alpha^2/1.9)n$ will not be processed. **Hint:** consider assigning the jobs one at a time, and upper bound the probability that a job is assigned to a machine that already has a job.

   (b) Conclude that choosing new machine for each round means that all jobs will be processed in $O(\log\log n)$ time with high probability.

8. **Set difference using Bloom Filter.** Suppose you have two sets, X and Y, such that $|X| = |Y| = m$ and $|X \cap Y| = r$. Create a Bloom filter using a table of size n for each of X and Y, both using the same set of k hash functions.

   (a) Determine the expected number of bits where the two Bloom filters differ, as a function of m, n, k, and r.

   (b) Show how this can be used as a method for estimating r.