600.464 Randomized and Big Data Algorithms
Homework #1 Answers

Ravindra Gaddipati

October 4, 2016

Problem 1 (7 points)

Let $[n] = 1, 2, ..., n$ and $g :\mapsto -1, 1$ be a function and let $\epsilon < 0.1$. We say that g is ϵ-good for $S \subset [n]$ if $|\sum_{i \in S} g(i)| \leq n^{1-\epsilon}$. Give a randomized algorithm that finds g such that for a set S the probability that g is not ϵ-good is at most $\frac{1}{n^(1-2\epsilon)}$. Give a randomized algorithm that for fixed sets $S_1, ..., S_t$ with $t = n^\epsilon$ is ϵ-good for all sets with probability $1 - o(1)$. Note that your algorithm does not know the sets $S, S_1, ..., S_t$ ahead of time.

Answer:

For each $i \in [n]$ we assign $g(i)$ 1 or -1. Let X be the sum of all $g(i)$ in S given S. We apply Chebyshev's inequality :

$$P(|X - E[X]| \geq k) \leq \frac{Var(X)}{k^2}$$

by first finding the expected value and variance X.

$$\begin{aligned}
X &:= \sum_{i \in S} g(i) \\
E[X] &= E\left[\sum_{i \in S} E[g(i)]\right] \\
&= E\left[\sum_{i \in S} 1 * P(g(i) = 1) - 1 * P(g(i) = -1)\right] \\
&= E\left[\sum_{i \in S} 0\right] = 0
\end{aligned}$$

By taking the variance and using the above result for all independent pairs i, j, the variance can be simplified to $E[|S|]$.

$$
\begin{aligned}
Var[X] &= E[X^2] - (E[X])^2 \\
&= E\left[\left(\sum_{i\in S} g(i)\right)^2\right] = E\left[\sum_{i\in S}\sum_{j\in S} g(i)g(j)\right] \\
&= \sum_{i\in S}\sum_{j\in S} E[g(i)g(j)] \\
&= \sum_{i\in S}\sum_{j\in S, j\neq i} E[g(i)]E[g(j)] + \sum_{i\in S} E[(g(i))^2] \\
&= \sum_{i\in S} (-1)^2 * P(g(i) = -1) + 1^2 * P(g(i) = 1) = \sum_{i\in S} 1 = |S|
\end{aligned}
$$

Since we do not know the distribution of cardinalities of S, we use the maximum value of $|S|$, n. If the distribution is uniform random, the expected value of $|S|$ can be used, which evaluates to $n/2$. In both cases the bound holds. With the variance and mean we can compute the Chebyshev inequality to bound P(g is not ϵ-good).

$$
\begin{aligned}
P(|X - E[X]| \geq k) &\leq \frac{Var(X)}{k^2} \\
P(|X| \geq n^{1-\epsilon}) &\leq \frac{n}{(n^{1-\epsilon})^2} = \frac{n}{n^{n-2\epsilon}} = \frac{1}{n^{1-2\epsilon}}
\end{aligned}
$$

Since $1/n^{1-2\epsilon} \leq 1/n^{1-2\epsilon}$ the algorithm satisfies the constraints. We can now show that this algorithm can be used to show that fixed sets $S_1, ..., S_t$ with $t = n^\epsilon$ is ϵ-good with probability $1 - o(1)$. This is the same as the same as showing at least 1 of the sets is not ϵ-good with probability $o(1)$. Let X_j now be the sum of $g(i)$ for all $i \in S_j$ for $j = 1, ..., t$.

$$
\begin{aligned}
X_j &\coloneqq \sum_{i\in S_j} g(i), j = 1, .., t \\
P\left(\bigcap_{j=1}^{t} |X_j| \leq n^{1-\epsilon}\right) &= P\left(\bigcup_{j=1}^{t} |X_j| > n^{1-\epsilon}\right)
\end{aligned}
$$

Applying the union bound and our result from part 1,

$$
\begin{aligned}
P\left(\bigcup_{j=1}^{t} |X_j| > n^{1-\epsilon}\right) &\leq \sum_{j=1}^{t} P(|X_j| > n^{1-\epsilon}) \\
&\leq \sum_{j=1}^{t} \frac{1}{n^{1-2\epsilon}} \\
&\leq \frac{t}{n^{1-2\epsilon}} = \frac{n^{\epsilon}}{n^{1-2\epsilon}} = n^{3\epsilon-1}
\end{aligned}
$$

The problem states that $\epsilon < 0.1$, so $n^{3\epsilon-1} > n^{3(0.1)-1} = n^{0.7}$. As $n \to \infty$ this goes to 0, so it is $o(1)$. Since the probability that at least one of the sets is not ϵ-good is $o(1)$, than all sets $S_1, ..., S_t$ are ϵ-good with probability $1 - o(1)$.

Problem 2 (1 point)

Let G be an undirected weighted graph with non-negative weights. In class we have learned the randomized algorithm for the MAX-CUT problem such that the expected value of the resulting cut is at least $0.5OPT$ where OPT is the value of a maximum cut in G. Design a randomized algorithm that runs in polynomial time and finds, with probability at least 0.99, a cut in G of size X such that $X \geq 0.49OPT$.

Answer:

Our algorithm will run the given max cut algorithm k times. After the end of the runs, the max cut is returned. To find how many times we need to run the algorithm, we let $Y = OPT - X$ where X is the weight of the cut. The failure event is when $Y > 0.51OPT$ We use the Markov inequality:

$$P(Y > a) \leq \frac{E[Y]}{a} = \frac{E[OPT] - E[X]}{a} = \frac{0.5OPT}{a}$$

$$P(Y > 0.51OPT) \leq \frac{0.5OPT}{0.51OPT}$$

$$P(X < 0.49OPT) \leq \frac{0.5}{0.51}$$

We want to fail less than or equal 0.01, so $\left(\frac{0.5}{0.51}\right)^k \leq 0.01$, giving us $k \geq 233$ repetitions. Our algorithm runs a polynomial time algorithm a constant number of times, so the runtime remains polynomial.

Problem 3 (2 point)

Explain why $ZPP \subseteq RP \cap coRP$. An informal (but correct) argument will be sufficient.

Answer:

First we informally define each class:

- An algorithm in class RP is an algorithm that runs in polynomial time, and is guaranteed to return NO if the correct answer is NO, and will return YES with $P > 1/2$ if the correct answer is YES.
- An algorithm in class $coRP$ is guaranteed to return YES if the correct answer is YES, and returns NO with $P > 1/2$ if the correct answer is NO.
- An algorithm is class ZPP always returns the correct YES or NO answer.

To show that $ZPP \subseteq RP \cap coRP$ we want to show that there is an intersection between the algorithms in RP and algorithms in $coRP$ that satisfy the the conditions for ZPP. Since ZPP algorithms always give the correct answer the intersection of the two classes would comprise the set of algorithms that report YES with $P = 1$ if the correct answer is YES and NO with $P = 1$ if the correct answer is NO.

The RP class of algorithms always returns NO if the correct answer is NO. Since $P = 1 > P1/2$ falls under the probability of and RP algorithm reporting YES if the correct answer is YES, we can say ZPP is a proper subset of RP. A similar argument follows for class $coRP$ reporting NO if the correct answer is NO.

Since ZPP is a proper subset of RP and $coRP$, the intersection of RP and $coRP$ is exactly equal to ZPP. Thus we have shown $ZPP \subseteq RP \cap coRP$. Since RP and $coRP$ are both of polynomial complexity, any subset of each class must also be polynomial. Thus, ZPP is also of polynomial complexity.

Problem 4 (1 point)

In class we have learned the randomized algorithm for the MIN-CUT problem that finds a minimum cut with probability at least $\frac{2}{n(n-1)}$ where n is the number of nodes in the graph, $n = |V|$. The algorithm can be implemented in time $O(n^2)$. Using the aforementioned algorithm design another (simple) algorithm that finds a minimum cut with probability at least $1 - \frac{1}{n^{10}}$. What is the running time of your algorithm?

Answer:

We know that the probability of finding a minimum cut $P(X) \geq \frac{2}{n(n-1)}$. We run this algorithm k times. For each i, the probability of the event of finding the min cut X_i is $P(X_i) \geq \frac{2}{n(n-1)}$ and equivalently the probability of failing to find a min cut is $P(\bar{X}_i) \leq 1 - \frac{2}{n(n-1)}$ for $i = 1, ..., k$. To find a minimum cut after k runs, we must not find the min cut previously. Since each run is independent, the probability of failing to find a minimum cut after k runs is:

$$P\left(\bigcap_{i=1}^{k} \bar{X}_i\right) \leq \left(1 - \frac{2}{n(n-1)}\right)^k$$

We want to bound this with $1 - \frac{1}{n^{10}}$. For large n, $1/n^{10} = 1/e^{ln(n^{10})} \leq 1 - 1/n^{10}$. Using the limit $\lim_{x\to\infty}\left(1 - \frac{1}{x}\right)^{ax} = \frac{1}{e^a}$ for large n:

$$P\left(\bigcap_{i=1}^{k} \bar{X}_i\right) \leq \left(1 - \frac{2}{n(n-1)}\right)^k \leq \frac{1}{e^{ln(n^{10})}}$$

where given $n > 0$:

$$k = \frac{n(n-1)}{2} ln\left(n^{10}\right) = 5n(n-1)ln(n)$$

The algorithm is able to find a minimum cut with $P > 1 - \frac{1}{n^{10}}$ after $5n(n-1)ln(n)$ repetitions. A single iteration of the algorithm is $O(n^2)$, yielding a total runtime of $O(n^2 * 5n(n-1)ln(n)) = O(n^4 ln(n))$. This satisfies the polynomial time requirement.

Problem 5 (1 point)

Let H be a class of all functions from M to N. Is it true that H is a 2-universal family? Prove your answer. Is it a good idea to use H for applications? Please explain your answer.

Answer:

H is a 2-universal family if for all pairs $(x, y) \in M, x \neq y$, $P(h(x) = h(y)) \leq 1/n$ for all $h \in H$ chosen uniformly at random. Let $m = |M|$ and $n = |N|$. There must be a function in H that maps every value of M to any value of N. As a result, we have n^m total functions. The number of functions that collide for a given (x, y) is then $n * n^{m-2}$. Given that the hash function h is chosen randomly and uniformly, the probability becomes

$$P(collision) = \frac{n^{m-1}}{n^m} = 1/n$$

Where n^{m-1} represents the number of functions that collide. This satisfies the definition of a 2-universal family.

Though this class of functions is 2-universal, it is not useful for applications. Since the number of functions is n^m the set of functions is exponential. To represent this family of functions, we would need $m * log(n)$ bits, unfeasible for large input sizes.

Problem 6 (2 point)

In class we have learned how to build a family H of 2-universal hash functions $h : M \mapsto N$ where $|M| = m$, $|N| = n$. Let $n > 10^{10}$ be a sufficiently large parameter. Can you construct a family of 2-universal hash functions H such that $|H| \leq 10$? Give an example of such family and prove its 2-universality. Otherwise prove that no such family exists.

Answer:

Let $\delta(x, y, H)$ be the number of collisions for pair $(x, y), x \neq y$ across all hash functions $h \in H$. For a class of functions to be 2-universal $\delta(x, y, H) \leq |H| < n$. We want to build a family H with $|H| \leq 10$ for $n > 10^{10}$. Since $|H|/n = 10/10^{10} \approx 0$, we desire a class of functions H that maps $M \mapsto N$ with no collisions. By definition $|M| \geq |N|$.

In the case $|M| = |N|$, we are able to design a hash function that maps each element M_i directly to some element N_i. This guarantees no collisions.

In the more likely case $|M| > |N|$, we are able to map M_i to N for $M_1, ..., M_{|N|}$ with no collisions by the same logic as for $|M| = |N|$ However for $M_i > M_{|N|}$, there are no free destination slots in N, thus a collision must occur. There cannot exist a bijection for $|M| > |N|$. We are unable to construct a H for $|H| \leq 10$ to guarantee no collisions for $|M| > |N|$.

Problem 7 (1 point)

Design a randomized algorithm that finds, in polynomial time, **all** min-cuts with probability at least 0.99. Can you derive an upper bound on the number of different min-cuts?

Answer:

Let C be the set of min cuts in the graph. Karger's algorithm gives us the probability of finding a min cut $c_i \in C$ with probability $\frac{n(n-1)}{2}$. When constructing a min cut with Karger's algorithm, nodes are collapsed into supernodes until two supernodes remain. With n initial nodes, we can collapse the graph into $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs of supernodes, giving us an upper bound for number of min cuts $|C|$. Applying the union bound to the probability of missing the min cut after running Karger's algorithm r times:

$$\bigcup_{c_i \in C} \left(1 - \frac{2}{n(n-1)}\right)^r \leq \sum_{i=0}^{|C|} \left(1 - \frac{2}{n(n-1)}\right)^r \leq 0.01$$

$$|C| \left(1 - \frac{2}{n(n-1)}\right)^r \leq 0.01$$

$$\left(1 - \frac{2}{n(n-1)}\right)^r \leq \frac{0.02}{n(n-1)}$$

$$rln(1 - \frac{2}{n(n-1)} \leq ln(0.02) - ln(n(n-1))$$

$$r \geq \frac{ln(0.02) - ln(n(n-1))}{ln\left(1 - \frac{1}{n(n-1)}\right)}$$

This is polynomial in the input size. Since Karger's algorithm is polynomial, and we run it r times for $k = \frac{n(n-1)}{2}$ minimum cuts, the resulting algorithm is polynomial time.

Citations

Various problems were discussed with Matthew Ige, Philip Piantone, Emily Wagner, and Lin Yang. Solutions were written up alone.
Lecture notes from Dr. Sanjeev Arora's Advanced Algorithm Design course was also helpful, in addition to the course notes provided on Blackboard.
http://www.cs.princeton.edu/courses/archive/fall15/cos521/