6.856 — Randomized Algorithms

David Karger

Handout #24, December 5th, 2002 — Homework 11 Solutions

Problem 1

(a) Let D be the disjoint union, and $N := |D|$. We will denote by $(a, x) \in D$ a particular assignment/clause pair in the disjoint union. That is, assignment a satisfies clause x and is the "clause x" copy of assignment a in the disjoint union. Note that our random sample chooses any pair $(a, x) \in D$ with probability $1/N$. It follows that

$$\begin{aligned} N \cdot E[X_t] &= N \cdot \sum_{(a,c)\in D} (1/N) \cdot (1/c_a) \\ &= \sum_a c_a \cdot 1/c_a \\ &= \sum_a 1 \end{aligned}$$

which is just the number of satisfying assignments, as claimed.

(b) We need to argue that we can estimate $E[X_t]$ to within $(1 \pm \varepsilon)$ using the desired number of trials. But observe that X_t is a random variable whose value is bounded in the range $[0, 1]$. Thus the generalized Chernoff bound of Homework 2, Problem 6 applies to sums of independent samples of X_t. Note also that $E[X_t] \geq 1/m$ since $c_a \leq m$ in all cases. It follows that the sum of $m\mu_{\varepsilon\delta}$ samples has expected value at least $\mu_{\varepsilon\delta}$, which means (by the definition of $\mu_{\varepsilon\delta}$) that the probability that this sum deviates by more than ε from its mean is at most δ.

(c) To estimate c_a, we will choose random clauses and check if a satisfies them. Note that the probability that a satisfies a randomly chosen clause is just c_a/m. So we can estimate c_a by estimating the probability a random clause satisfies a. We will refer to these random choices of clauses as "sub-trials" and reserve the word "trial" to refer to choosing a random assignment and then carrying out a series of sub-trials to estimate its coverage.

We'll use a δ' that we set later. We will keep choosing clauses until we see $\mu_{\varepsilon\delta'}$ clauses that a satisfies. We expect to have to choose $(m/c_a)\mu_{\varepsilon\delta'}$ clauses to do this. Once we see this many clauses, we get an estimate c for c_a that is accurate to within $(1 \pm \varepsilon)$ with probability $1 - \delta'$. It follows that $1/c = (1 \pm 2\varepsilon)/c_a$, so that we have an $O(\varepsilon)$-accurate approximation to $1/c_a$ with probability $1 - \delta'$. If each of our estimates of $1/c_a$ is accurate to within $1 + O(\varepsilon)$, then our sum of estimates approximates the sum of $1/c_a$ values to within $O(\varepsilon)$, which in turn approximate the mean $E[X_t]$ to within $O(\varepsilon)$. The errors in these approximations multiply, so the accuracy of our approximation is $(1+O(\varepsilon))^{O(1)}$, which is just $1+O(\varepsilon)$. Of course, we asked for an ε approximation. To get an ε-accurate estimate instead of an $O(\varepsilon)$-accurate one we just use a constant-factor smaller initial ε. Now we can set δ'. We want our estimates for $1/c_a$ to be accurate in *all* the trials we perform. Since we carry out $O(m\mu_{\varepsilon\delta})$ trials, to make the union bound work we will set $\delta' = \delta/(m\mu_\varepsilon\delta)$. This means the probability *any* trial hits a bad estimate from its sub-trials is at most $1/\delta$.

One subtlety in this problem was realizing that if one approximated $c \approx c_a$ to within an ε bound, then $1/c$ was also a legitimate 2ε approximation for $1/c_a$. This is not equivalent to asserting (incorrectly) that the inverse of the expectation of a variable is the same as the expectation of its inverse, because we are not in fact dealing with an expectation. We are simply approximating the value of a quantity, and then inverting that value to obtain an approximation for its inverse.

(d) Let's compute the expected running time of a trial, measured in terms of *the number of clauses against which we test an assignment.* As just discussed, if the result returned by the trial is $1/c_a$ then the number of clauses we need to sample is $O(m/c_a)\mu_{\varepsilon\delta}$. It follows that the expected number of clauses we need to sample in one trial is

$$\begin{aligned} O(E[(m/c_a)\mu_{\varepsilon\delta}]) &= O(m\mu_{\varepsilon\delta}) \cdot E[1/c_a] \\ &= O(m\mu_{\varepsilon\delta}E[X_t]) \end{aligned}$$

On the other hand, recall that the goal of these trials is to estimate $E[X_t]$, which means that we need to carry out about $\mu_{\varepsilon\delta}/E[X_t]$ trials to do so. It follows that the total work done is the product of these two quantities. This product cancels the quantity $E[X_t]$, leaving us with a total time of

$$O(m\mu_{\varepsilon\delta}^2)$$

which is essentially linear in the formula size.

We can improve the μ^2 term to μ with a longer analysis. Rather than waiting for $\mu_{\varepsilon\delta'}$ sub-trials to yield satisfied clauses for a sub-trial, we stop as soon as we see one satisfied clause and, if we made s samples, we take $1/s$ as our estimate. The number of sub-trials we perform is then negative binomial distribution with mean m/c_a, the right value. So we can use a Chernoff bound on sums of negative binomial distributions (with different means) to argue that we get an accurate estimate.

Recall that this bound is the number of clause vs. assignment tests we carry out in the algorithm. The actual running time needs to take into account the number of variables we need to test in a clause, which depends on the formula size.

(e) (optional) If some clauses in a DNF formula are much larger than others, then the probability that they become true is so small compared to the probability other clauses become true that we can discard those clauses from the formula without affecting the truth probability significantly. Once all clauses are the same size, our argument that we test roughly m clauses means that we test a number of variables roughly equal to the total size of the formula.

Problem 2

1. Let c_A be the coverage of assignment A, h the number of satisfying assignments in the disjoint union, and k the number of satisfying assignments. The probability that the algorithm outputs a value in a given attempt is

$$\sum_A (1/c_A)(c_A/h) = k/h.$$

Notice that the number of iterations before an assignment is output is geometrically distributed with parameter $p = k/h$. The expected time before a "success" is $1/p$. But we argued previously that $k/h > 1/m$. Thus the expected time to output a value is $O(m)$.

2. The probability an assignment A is output in a given trial is

$$\begin{aligned} \Pr[A \text{ output} \mid A \text{ picked}] \cdot \Pr[A \text{ picked}] &= (1/c)(c/h) \\ &= 1/h \end{aligned}$$

It follows that the probability assignment A is output in a trial, *given that there was an output in that trial*, is (by the definition of conditional probabilities)

$$\frac{\Pr[\text{stop and output } A]}{\Pr[\text{stop}]} = (1/h)/(k/h) = 1/k.$$

3. When we choose a random assignment A (which takes $O(m)$ time) we can estimate c_A in $O(m\mu_{\varepsilon,\delta}/c_A)$ time as shown in problem 4(c). With probability $1-\delta$ that estimate is within $(1 \pm \varepsilon)$ of the correct value. Conditioning on that fact, we obtain from (a) and (b) that its output probability is within $(1 \pm O(\varepsilon))$ of $1/k$. By starting with a constant factor smaller ε, we can get to a $(1 \pm \varepsilon)$ accurate probability (and the constant disappears in the O-bound), giving us a (ε, δ) approximation scheme.

Let us now estimate the running time. Consider running the experiment to output an assignment. In the process we consider some sequence of assignments $A_1, A_2, \ldots, A_\infty$ (for simplicity assume that the experiment runs forever, even though we stop at the first output). The "expected number of outputs" that we produce from among $A_i, \ldots, A_j$ is $1/c_{A_i} + \cdots + 1/c_{A_j}$. Of course, as soon as this exceeds 1, the Chernoff bound tells us that we produce at least one output with probability $e^{-1/2}$, i.e. a constant. So let's break the infinite sequence of trials (assignments) up into "epochs": the first epoch starts at A_1 and continues until the inverse coverages sum to 1; then the next epoch starts and continues until the inverse coverages sum to 1 again, and so on. In each epoch, we have a constant probability of outputting an assignment, so the expected number of epochs until we output an assignment is a constant.

How long do we spend computing coverages in an epoch? $O(m\mu_{\varepsilon,\delta}(\sum 1/c_A))$ time, where the sum is taken over assignments A in the epoch; this is just $O(m\mu_{\varepsilon,\delta})$. Since we expect to evaluate only a constant number of epochs, this is also the expected computation time for our whole algorithm.