10 | 10 | 10 | 10 | 10

**6.856** **Dan Ports**

**Randomized Algorithms** drkp@mit.edu

2005/04/01 Collaborators: {sarahl, gpickard}

# Problem Set 7

**Problem 1:**

Suppose we take $n$ samples from a probability distribution with mean $\mu$ and standard deviation $\sigma < \mu$. Let $X_n$ be the random variable giving the sum of the samples, and $\mu_n$ and $\sigma_n$ its mean and standard deviation respectively. Certainly $\mu_n = n\mu$, and $\sigma_n^2 = n\sigma^2$, so $\sigma_n = \sqrt{n}\sigma < \sqrt{n}\mu$. Then we can use Chebyshev's bound to bound the probability that $X_n$ exceeds its expected value by more than some constant $k$ times its standard deviation:

$$\mathbf{Pr}\left[|X_n - n\mu| \geq k\sqrt{n}\mu\right] \leq \frac{1}{k^2}$$

If we set $k = \sqrt{2}$, then

$$\mathbf{Pr}\left[|X_n - n\mu| \geq \sqrt{2n}\mu\right] \leq \frac{1}{2}$$

Then we can estimate $\mu$ as $\frac{X_n}{n}$, which gives an error within a multiplicative factor of $\frac{\sqrt{2n}}{n} = \sqrt{\frac{2}{n}}$ with probability at least $1/2$. If we choose $n \geq \frac{2}{\epsilon^2}$, then we achieve a $(1 \pm \epsilon)$ approximation with probability at least $1/2$.

We apply the technique of the previous problem set to reduce the probability of error from $1/2$ to $\delta$: we repeat the previous process $m$ times and compute the median of the results. The result from the previous problem set tells us that $m = O\left(\log \frac{1/2}{\delta}\right) = O\left(\log \frac{1}{\delta}\right)$ repetitions suffice, since the probability improvement is exponential in $m$.

So this algorithm gives a $(1 \pm \epsilon)$-approximation with probability $1 - \delta$. The total number of samples required is $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$.

**Problem 2:** *

Let $G$ be a graph with $n$ vertices and $m$ unit-length edges. We give an algorithm based on the transitive closure algorithm for identifying for all vertices the number of other vertices within distance $d$.

We perform the following sampling step repeatedly to estimate the size of the $d$-neighborhood of each vertex: we begin by permuting the vertices randomly and call the resulting order $v_1, \ldots, v_n$. To estimate the size of the $d$-neighborhood of a vertex $x$, we will find the smallest index $l_x$ of a vertex within distance $d$ of $x$, and then estimate the size from this as in the transitive closure algorithm. Let $i$ be the smallest index such that $v_i$ is still in the graph. Then perform a breadth-first search from $v_i$ to identify all vertices $x$ such that the distance from $v_i$ to $x$ is at most $d$; set $l_x = i$ for these. This gives a sample for every vertex within distance $d$ of $v_i$.

We want to achieve $\mu_{\epsilon\delta}$ samples for each vertex in order to obtain a good estimation (the error bound analysis is the same as in the transitive closure case). To accomplish this, rather than keep a single counter $f_i$ for vertex $i$ that is incremented every time the vertex is traversed in the BFS and delete $i$ once $f_i$ reaches $\mu_{\epsilon\delta}$, we maintain $d$ counters $f_{i,1}, \ldots, f_{i,d}$ for each vertex. We increment counter $f_{i,j}$ each time vertex $i$ is reached in a BFS from a source at distance $j$ from $i$, and we remove vertex $i$ when all of its counters equal $\mu_{\epsilon\delta}$. This ensures that vertex $i$ is only deleted once every vertex reachable from it within distance $d$ has received $\mu_{\epsilon\delta}$ samples, which gives the desired error

* Consulted Edith Cohen's lecture notes on the transitive closure algorithm since our lecture notes were sparse.
http://www.cs.berkeley.edu/~edith/cs270/Lectures/lecture28.ps

bound. Since the algorithm now requires $d$ times as many samples before deleting a vertex, the expected runtime of the algorithm is increased by a constant factor of $d$ over that of the transitive closure algorithm, still $O\left((m+n)\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$ within a logarithmic factor.

**Problem 3:**

Consider the problem of estimating the probability of a DNF formula being satisfied with a random assignment of variables where each variable is true with probability $p \leq 1/2$. We modify the sampling algorithm over the disjoint union to account for the fact that the probabilities are no longer uniform.

Following the terminology in M&R, let $U$ be the universe of all truth assignments, $G$ the set of satisfying assignments, and $H_i$ the set of assignments that satisfy clause $i$. As before, $G = \bigcup_{i=1}^{m} H_i$. Now, however, rather than estimating $|G|$, we want to estimate $\mathbf{Pr}[G]$, which we define to be $\mathbf{Pr}[a]$ for every assignment $a \in G$. As before, the canonical element for $v$ in the disjoint union $\biguplus_i H_i$ is the element corresponding to $v$ in the least $i$ such that $v \in H_i$. We sample elements in the disjoint union to determine the probability that a randomly-chosen assignment is satisfying.

For each set $H_i$, let $\mathbf{Pr}[H_i]$ be the probability that a randomly selected assignment is in $H_i$. Note that we can easily determine this probability: if clause $i$ requires $x$ variables to be set true and $y$ variables to be set false (assuming there are no contradictions), then $\mathbf{Pr}[H_i] = p^x(1-p)^y$. So rather than sample from set $H_i$ with probability $\frac{|H_i|}{\sum_j |H_j|}$, we sample from set $H_i$ with probability $\frac{\mathbf{Pr}[H_i]}{\sum_j \mathbf{Pr}[H_j]}$.

Once we have chosen a $H_i$ using the probability described above, we choose a random satisfying assignment $a$ by setting the variables determined by clause $i$ appropriately, then setting each of the remaining variables true with probability $p$ and false with probability $1-p$. Testing this assignment to see whether $(a, i)$ is the canonical assignment for $a$ (by checking whether there is some clause $j < i$ such that $a$ satisfies clause $j$) estimates the probability $\mathbf{Pr}[a \in G \mid a \in H_i]$. Since we chose set $H_i$ with probability $\mathbf{Pr}[H_i]$, repeatedly performing this sampling estimates $\mathbf{Pr}[a \in G] = \mathbf{Pr}[G]$. which is the desired probability.

**Problem 4:**

**a)** Let $G$ be the set of satisfying assignments and $\mathscr{G}$ be the disjoint union of the satisfying assignments. Define $c_a$ to be the number of clauses satisfied by $a$ and $X_t = \frac{1}{c_a}$. Then by definition $\mathbf{E}[X_t] = \sum_{a \in \mathscr{G}} \frac{1}{N}\frac{1}{c_a}$. So $N\mathbf{E}[X_t] = \sum_{a \in \mathscr{G}} \frac{1}{c_a}$. But this is precisely the number of satisfying assignments, since each satisfying assignment appears in the disjoint union $\mathscr{G}$ exactly $c_a$ times, and each time contributes $\frac{1}{c_a}$ to the sum.

**b)** We can estimate the number of satisfying assignments to the DNF formula by sampling $n$ random assignments from the disjoint union. The quantity $X = \frac{\sum_{t=1}^{n} X_t}{n}$ is an estimator for $\mathbf{E}[X_t]$, so $NX$ is an estimator for the number of satisfying assignments.

Since $X_t$ is a random variable assuming only values in the $[0,1]$ interval, we can apply the generalized Chernoff bound. Since $X_t = \frac{1}{c_a}$ and $c_a$ is at most $m$, $X_t \geq \frac{1}{m}$, and so $\mathbf{E}[X_t] \geq \frac{1}{m}$. Thus,

$$\mathbf{Pr}\left[X \geq (1+\epsilon)\mathbf{E}[X_t]\right] \leq e^{-\frac{\epsilon^2 n \frac{1}{m}}{3}}$$

If $n = \Theta(m\mu_{\epsilon\delta})$,

$$\mathbf{Pr}\left[X \geq (1+\epsilon)\mathbf{E}[X_t]\right] \leq e^{-\frac{\epsilon^2 \Theta(\mu_{\epsilon\delta})}{3}}$$

which by the definition of $\mu_{\epsilon\delta}$ is less than $\delta$. Essentially the same argument holds for $\mathbf{Pr}[X \geq (1+\epsilon)\mathbf{E}[X_t]]$, so $X$ estimates $\mathbf{E}[X_t]$ and $NX$ estimates the number of satisfying assignments with the correct probability and error bound.

**c)** Now fix some assignment $a$. We can estimate $c_a$ by randomly sampling clauses and checking how many of them $a$ satisfies. We sample until we have found $\mu_{\varepsilon\gamma}$ satisfied clauses; this requires $\frac{m}{c_a}\mu_{\varepsilon\gamma}$ clauses in expectation since $\frac{c_a}{m}$ clauses are satisfied. We scale up the result and let $Y$ be the estimator for $c_a$. Then $Y$ is within a factor of $(1\pm\varepsilon)$ of $c_a$ with probability $1-\gamma$. So with probability $1-\gamma$,

$$\frac{1}{(1+\varepsilon)c_a} \leq \frac{1}{Y} \leq \frac{1}{(1-\varepsilon)c_a}$$
$$\frac{1-\varepsilon}{(1+\varepsilon^2)c_a} \leq \frac{1}{Y} \leq \frac{1+\varepsilon}{(1-\varepsilon^2)c_a}$$
$$(1-O(\varepsilon))\frac{1}{c_a} \leq \frac{1}{Y} \leq (1+O(\varepsilon))\frac{1}{c_a}$$

We choose $\varepsilon$ to be a constant factor of $\epsilon$, so that $\frac{1}{Y}$ estimates $\frac{1}{c_a} = X_t$ within a factor of $(1\pm\epsilon)$ with probability $1-\gamma$. Since we sum $O(m\mu_{\epsilon\delta})$ samples of $X_t$, we obtain a overall multiplicative error of $(1\pm\epsilon)$ if none of the $X_t$ samples differ by more than $(1\pm\epsilon)$. Using the union bound, we can achieve this with probability $\delta$ if we choose $\gamma = O\left(\frac{\delta}{m\mu_{\epsilon\delta}}\right)$.

**d)** Testing an assignment with $c_a$ satisfied clauses requires evaluating $\mathbf{E}\left[\frac{m}{c_a}\right]\mu_{\varepsilon\gamma}$ clauses in expectation. Note that $\mu_{\varepsilon\gamma} = O(\mu_{\epsilon\delta})$ since $\varepsilon$ and $\gamma$ are within constant factors of $\epsilon$ and $\delta$ respectively. So it requires evaluating $O\left(\mathbf{E}\left[\frac{m}{c_a}\right]\mu_{\epsilon\delta}\right)$ clauses in expectation.

We need to sample until $\mu_{\epsilon\delta}$ satisfying assignments are found. The probability of a sampled assignment being a satisfying one is $X_t$, so in expectation we test $O\left(\frac{\mu_{\epsilon\delta}}{\mathbf{E}[X_t]}\right)$ assignments. So the total number of clauses evaluated, and thus the total work if the length of each clause is constant, is expected to be

$$O\left(\frac{\mu_{\epsilon\delta}}{\mathbf{E}[X_t]}\mathbf{E}\left[\frac{m}{c_a}\right]\mu_{\epsilon\delta}\right) = O\left(\frac{\mu_{\epsilon\delta}}{\mathbf{E}[X_t]}m\mathbf{E}[X_t]\mu_{\epsilon\delta}\right) = O\left(m\mu_{\epsilon\delta}^2\right)$$

**Problem 5:**

**a)** Let $G$ be a graph where each edge has failure probability $p$, and $e$ an edge in the graph. Let $x_e$ be the event that edge $e$ fails, and $F$ the event that the graph fails. Then by Bayes' Rule,

$$\mathbf{Pr}[x_e \mid F] = \mathbf{Pr}[F \mid x_e]\frac{\mathbf{Pr}[x_e]}{\mathbf{Pr}[F]} \tag{1}$$

This reduces the problem of estimating $\mathbf{Pr}[x_e \mid F]$ to the problem of estimating three other quantities. $\mathbf{Pr}[x_e]$ is known; it is simply $p$. $\mathbf{Pr}[F]$ can be estimated to within a factor of $(1\pm\epsilon)$ in the appropriate time bound via a direct application of the FPRAS. $\mathbf{Pr}[F \mid x_e]$ can be estimated to within the same factor by removing edge $e$ – as though it had failed – and then applying the FPRAS to the resulting graph. Applying Equation 1 to these values, we obtain a value for $\mathbf{Pr}[x_e \mid F]$. This can be made accurate to within a factor of $(1\pm\epsilon)$ by choosing the $\epsilon$ factor for the FPRAS subproblems to be a constant factor smaller. This gives a FPRAS for $\mathbf{Pr}[x_e \mid F]$.

**b)** If $x_e$ occurs, then $e$ fails and can be removed. Computing the probability of failure on the resulting graph gives $\mathbf{Pr}[F \mid x_e]$.

If $x_e$ does not occur, then we know that $e$ does not fail, and will always be present in the network. So we contract the endpoints of $e$. Computing the probability of failure on the resulting graph gives $\mathbf{Pr}[F \mid \overline{x_e}]$.

c) We now give an algorithm for producing a random disconnected version of $G$, conditioned on $F$. We do so by reducing the problem to a smaller instance of itself. We begin by randomly selecting an edge $e$, and using the FPRAS presented above to estimate the failure probability of $e$ given $F$. With the resulting probability, we decide whether $e$ will be up or down in the disconnected graph. If it is up, we contract $e$'s endpoints; if it is down, we remove $e$ from the graph. This reduces the problem to a smaller problem with at least one less edge. We repeat this process until the graph is disconnected.