CMPUT 675: Randomized Algorithms	**Fall 2005**

Lecture 5: Sept 22

Lecturer: Mohammad R. Salavatipour *Scribe: Bruce Fraser*

5.1 Chernoff Bound

Today, we are going to find a bound on the tails probabilities that is considerably sharper than Markov's and Chebyshev's. We focus on the sum of independent Bernoulli trials. If $X_1, \ldots, X_n$ are independent Bernoulli trials with $\Pr[X_i = 1] = p$ then $X = \sum_{i=1}^n X_i$ has a binomial distribution. In more general cases, if $\Pr[X_i = 1] = p_i$ then $\mathrm{E}[X] = \sum_{i=1}^n p_i$ and these are referred to as Poisson trials.

Theorem 5.1 (Chernoff Bound) *Assume $X_1, X_2, \ldots, X_n$ are independent Poisson trials with $\Pr[X_i = 1] = p_i, for 0 < p_i < 1$. Then for $X = \sum_{i=1}^n X_i$, $\mu = E[X]$:*

1. *for any $0 < \delta$: $\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu$.*
2. *for any $0 < \delta \leq 1$: $\Pr[X \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3}$.*
3. *for any $R \geq 6\mu$: $\Pr[X \geq R] \leq 2^{-R}$.*

Note that these bounds are independent of μ.

Proof: We prove the first bound. For all $t > 0$:

$$\Pr[X \geq (1+\delta)\mu] = \Pr[tX \geq t(1+\delta)\mu] = \Pr[e^{tX} \geq e^{t(1+\delta)\mu}].$$

By Markov's inequality,

$$\Pr[e^{tX} \geq e^{t(1+\delta)\mu}] \leq \frac{\mathrm{E}[e^{tX}]}{e^{t(1+\delta)\mu}} = \frac{\mathrm{E}[\prod_i e^{tX_i}]}{e^{t(1+\delta)\mu}} = \frac{\prod_i \mathrm{E}[e^{tX_i}]}{e^{t(1+\delta)\mu}}.$$

Now $e^{tX_i} = \begin{cases} 1, & 1-p_i \\ e^t, & p_i \end{cases}$. So $\mathrm{E}[e^{tX_i}] = e^t p_i + (1-p_i) = 1 + p_i(e^t - 1)$.. Thus

$$\Pr[X \geq (1+\delta)\mu] \leq \frac{\prod_i [p_i(e^t+1)]}{e^{t(1+\delta)\mu}} \leq \frac{\prod_i e^{p_i(e^t-1)}}{e^{t(1+\delta)\mu}},$$

(since always $1 + x \leq e^x$). Continuing,

$$\Pr[X \geq (1+\delta)\mu] \leq \frac{e^{(e^t-1)\sum_i p_i}}{e^{t(1+\delta)\mu}} = \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}}.$$

If we let $t = \ln(1+\delta)$, then

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu.$$

■

Similarly, we can show:

Theorem 5.2 *Under the same assumptions as in Theorem 5.1:*

1. *for any* $0 < \delta < 1$*:* $\Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^{\mu}$.
2. *for any* $0 < \delta < 1$*:* $\Pr[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}$.

Corollary 5.3 *For* $0 < \delta < 1$*:*

$$\Pr[|X-\mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}.$$

5.1.1 Example 1: Flipping n unbiased coins

Suppose we flip n unbiased coins uniformly randomly and independently. Let $X_i = 1$ if coin i is heads and $X = \sum X_i$ is the number of heads, $E[X] = \frac{n}{2}$. Using the Chernoff bound,

$$\Pr\left[X \geq \frac{n}{2} + \lambda\right] = \Pr\left[X \geq \mu(1+\frac{\lambda}{\mu})\right] \leq e^{-\left(\frac{\lambda}{\mu}\right)^2 \frac{\mu}{3}} = e^{\frac{-\lambda}{3\mu}}.$$

Now try $\lambda = \sqrt{3n\ln n}$. Then $\frac{\lambda^2}{3\mu} = \frac{3n\ln n}{3\frac{n}{2}} = 2\ln n$. So we have

$$\Pr[X \geq \mu + \sqrt{3n\ln n}] \leq e^{-2\ln n} = n^{-2}.$$

This is a better bound than given by Chebyshev's inequality: $\sigma^2[X] = \sum \sigma^2[X_i] = \frac{n}{4}$ and

$$\Pr[X \geq \mu + \lambda] \leq \frac{\sigma^2}{\lambda^2} \in \frac{O(n)}{O(n\ln n)} = O(\frac{1}{\ln n}).$$

5.1.2 Example 2: Minimum Discrepancy Coloring of Hypergraphs

Consider the problem of 2-coloring hypergraphs.

Definition 5.4 *A hypergraph is a set of vertices,* $V = \{v_1, v_2, ..., v_n\}$ *and a set of edges* $E = \{e_1, e_2, ..., e_m\}$*, where* $e_i \subseteq V$*. (Graphs are the special case when* $|e_i| = 2$*.)*

Our goal is to color vertices red and blue such that we have a balanced number of colors in each edge. That is, we want to minimize the *discrepancy* where $\mathrm{Disc(e_i)} = \max_{\mathrm{e_i}} |\mathrm{number\ of\ reds\ in\ e_i} - \mathrm{number\ of\ blues\ in\ e_i}|$.

Lemma 5.5 *A random 2-coloring gives* $Disc \leq O(\sqrt{n\ln n})$*.*

Proof: Consider random variable X_v:

$$X_v = \begin{cases} 1 & \text{if v is red.} \\ 0 & \text{otherwise.} \end{cases}$$

Let $X_{e_i} = \sum_{v \in e_i} X_v$. Then $E[X_{e_i}] = \frac{|e_i|}{2}$ and $\mathrm{Disc(e_i)} = 2\left|\frac{|e_i|}{2} - X_{e_i}\right|$. Using Chernoff bound with $\lambda = \sqrt{3n \log m}$:

$$\Pr\left[\left|\frac{|e_i|}{2} - X_{e_i}\right| > \lambda\right] \leq 2e^{\frac{2\lambda^2}{3|e_i|}} < e^{-O\left(\frac{n \ln m}{n}\right)} \leq \frac{1}{m^2}.$$

Since there are m edges, using union bound:

$$\Pr[\text{any edge has disc.} > \sqrt{3n \ln m}] \leq m\left(\frac{1}{m^2}\right) = \frac{1}{m}.$$

■

5.1.3 Oblivious Routing in Hypercubes

Consider a network with N nodes, with directed links between nodes. Furthermore, every node i has a packet v_i which has a destination $d(i)$. We assume that these destinations are all distinct, i.e. $d(i)$'s form a permutation of vertices. The system is synchronous and works in rounds. In every round vertices may send some packets to some of their neighbors. Every edge can transmit only one packet in every time step and it takes exactly one time step for a packet to travel an edge. If there are several packets at a node that want to go through an edge we buffer them (queue) and send them one by one. We usually use FIFO queuing at buffers.

We are looking for a routing scheme for these packets with minimum number of rounds. The scheme must be *oblivious*; i.e. every packet is routed independently, without regard to where other packets are routed. At every node we decide where to send each packet based only on its destination $d(i)$. Typically the network is sparse. The model of network we consider is a k-dimensional hypercube.

Definition 5.6 *A hypercube or k-cube has 2^k nodes, with k-bit binary strings as labels for each node. Two nodes are connected with directed edges if and only if their labels differ in exactly one bit. Thus in a k-cube, each node is of degree k.*

The following theorem (which we don't prove) gives a lower on the number rounds in the worst case for any deterministic algorithm.

Theorem 5.7 *For any deterministic algorithm, there is a permutation that requires $\Omega\left(\frac{\sqrt{N}}{d}\right)$ rounds, where d is the degree of the nodes.*

In the case of the hypercube, the bound is $\Omega\left(\frac{\sqrt{2^k}}{k}\right)$. Our goal is to route all the packets in $O(\log N)$ rounds, where $N = 2^k$ (i.e. $O(k)$ rounds).

A typical oblivious and simple routing algorithm is call *bit-fixing*: given v_i to be sent to $d(i)$ at current node $\sigma(i)$, compare $d(i)$ with $\sigma(i)$, and find the leftmost bit at which the two strings differ, say bit j. Then send it to the neighbour of $\sigma(i)$ that differs from it in bit j. So in every step, we "fix" one of the bits by making it closer and closer to the destination. Obviously, if there are no other packets in the network, a packet is delivered in k steps.

Let us first examine the effect of bit-fixing when the input is a random permutation. Define $T(e_i)$ to be the number of routes that use edge e_i. By symmetry, $\mathrm{E}[T(e_i)]$ is the same for all the edges. The expected length of a route is $\frac{k}{2}$ (since the source and destination can differ in at most k bits). There are N nodes (and so N packets), so the expected number of edge uses is $\frac{Nk}{2}$. There are Nk edges in the cube, so expected load per edge is $\frac{1}{2}$. Hence $\mathrm{E}(T(e_i)) = \frac{1}{2}$. Assume that the $T(e_i)$'s are independent. We can use Chernoff bound to say:

$$\Pr[|T(e_i) - \mathrm{E}(T(e_i))| > 6k] \leq 2^{-6k} = \frac{1}{N^6}.$$

So each edge causes a delay of at most $O(k)$ w.h.p. Since every route has to go through $O(k)$ edges, it will have a delay of less than $O(k^2)$. Of course this argument is loose (because the variables $T(e_i)$'s are *not* independent. But the idea works.

The key idea for our randomized improvement: first try to route each v_i to a random location, then route back to original destination. This way we confound an adversary choosing a worst-case permutation for us.

Two Phase Algorithm
Phase 1: Route packets to a random location (not a permutation!) using bit-fixing.
Phase 2: Route packets from the random intermediate locations to their destination using bit-fixing.

Analysis of this algorithm will follow in the next lecture.