CS 388R: Randomized Algorithms Fall 2017

Lecture 16 — October 31, 2017

Prof. Eric Price *Scribes: Andrew Russell, Aditya Gupta*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

n Coin Flips

Question: We have n coins, the probability of heads on each coin is $p(\ll 1)$. Find $Pr(\#heads \geq \frac{n}{2})$

Approach 1) We begin by finding lower and upper bounds.
For lower bound, when all tosses are heads,

$$Pr(\#heads \geq \frac{n}{2}) \geq p^n$$

For upper bound, using union bound,

$$\begin{aligned} Pr(\#heads \geq \frac{n}{2}) &\leq \binom{n}{\frac{n}{2}} p^{\frac{n}{2}} \\ &\leq 2^n p^{\frac{n}{2}} \\ &\leq (4p)^{\frac{n}{2}} = e^{-nlog(\frac{1}{4p})\theta(1)} \end{aligned}$$

Approach 2) Let $X_i \in [0,1], Var(X_i) = p(1-p)$,

$$\begin{aligned} X_i &\in subgamma(2\sqrt{p}, 2) \\ \sum_{i=1}^{n} X_i &\in subgamma(2\sqrt{np}, 2) \\ Pr[\sum_{i=1}^{n} X_i \geq np + t] &\leq e^{-min\{\frac{t^2}{8np}, \frac{t}{4}\}} \end{aligned}$$

Set $t = (\frac{1}{2} - p)n \geq \frac{n}{4}$

$$\begin{aligned} &\leq e^{-min\{\frac{n}{128p}, \frac{n}{16}\}} \\ &\leq e^{-\frac{n}{16}} \end{aligned}$$

Thus, using the first approach gives us a tighter bound.

1 Overview

In this lecture we discuss streaming algorithms. Our primary motivation here is to be able to compute interesting functions on a stream $v_1, v_2, \ldots, v_m$ of data and corresponding counts $x = (x_1, x_2, \ldots, x_n)$ where $x_i = (\#j \in [m] \mid v_j)$ where n and m are much larger than our capacity to store.

There are two models of streams that we will consider.

- Insertion-only: items are only added to the stream.
- "Turnstile": items can be inserted *and* deleted from the stream.

In this lecture, we will primarily focus on the insertion only model.

2 Streaming

Suppose we want to answer the following questions about the stream's count vector.

1. We wish to estimate the zero norm $||x||_0 := (\#$ nonzero $x_i)$ within a multiplicative factor of $(1 \pm \epsilon)$ with probability $(1 - \delta)$.
2. Sample a random $i \in \mathsf{Supp}(x)$ where $\mathsf{Supp}(x)$ is the nonzero coordinates of x.
3. Estimate $||x||_1$.
4. Estimate $||x||_2$.

We will make frequent use of hash functions to conserve space.

2.1 Estimating the 2-norm

We start with estimating the 2-norm. First we note that if we pick an $m \times n$ matrix A with entries $a_{ij} \in \{-1, 1\}$ then if $m = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta}$ we will have $\frac{1}{m}||Ax||_2^2 = (1 \pm \epsilon)||x||_2^2$. So if we can compute Ax progressively, we can estimate $||x||_2$. We can do this by applying A progressively to the stream values. If e_i is the ith standard basis vector for $\mathbb{R}^n$, then we have:

$$Ax = A\left(\prod_{i=1}^{m} e_{v_j}\right) = \prod_{i=1}^{m} Ae_{v_j}$$

We now have one last thing to consider, which is the storage of matrix A. We can generate A progressively by using 4-wise independent hash functions h_i for each row i of A. Specifically,

$A_{ij} = h_i(j)$. This is known as the "AMS Sketch." Note that this gives us a result with constant probability: for $m = O(\frac{1}{\epsilon^2}$ rows, we get an estimation $(1 \pm \epsilon)||x||_2^2$ with $\frac{9}{10}$ probability. To get a higher probability $1 - \delta$, we can use the median of matrices $A_1, A_2, \ldots, A_{\log \delta}$.

2.2 Sampling from the support

First, consider a random hash function $h : [n] \to [0, 1]$. As we receive stream data v_j, keep track of the v_j that has minimum $h(v_j)$. Since h is a random function, this minimum value will be random also, and independent of the order of the stream data. This approach is known as the "min-hash." However, this is problematic for our space-constrained setting, since storing a truly random function is $O(n)$. So instead we will use a $O(\log(\frac{1}{\epsilon})$-wise independent hash function, which gives us

$$\mathbb{P}[min_{x \in S}(h(x)) = h(\tilde{x})] = \frac{1 \pm \epsilon}{|S|} \quad \forall S \subseteq [n], \tilde{x} \in S$$

2.3 Estimating the zero norm

We will use the min-hash technique for this problem as well. Let $k = ||x||_0$ be the number of nonzero entries of x. Intuitively, if we use a random hash function, its minimum value across nonzero values of x will get smaller as the number of nonzero values of x increases. More precisely, we have $E[min(h(x))] = \frac{1}{k+1}$ since the range of the hash function is the unit interval. Thus, $k \approx \frac{1}{min(h(x))} - 1$ and so we want to estimate $E[min(h(x))]$, which we can do more precisely with multiple hash functions: $\frac{1}{S} \sum_{i=1}^{|S|} min_{x \in S}(h_i(x))$. This requires $\frac{1}{\epsilon^2}$ words ($O(\frac{1}{\epsilon^2} \log n)$ bits). Alternatively, we can store the log of the minimum (i.e., the number of leading zeros), which would give us a $\log \log n$ factor rather than $\log n$.