CS 388R: Randomized Algorithms	Fall 2015

Lecture 17 — Nov 4, 2015

Prof. Eric Price — *Scribe: Enxu Yan, Fu Li*

1 Sampling

Example: estimating π:

- Choose $x, y \in [-1, 1]$ at random. Check if $x^2 + y^2 \le 1$.
- The fraction of samples satisfying $x^2 + y^2 \le 1$ is an estimate of the true $prob = \frac{\pi}{4}$.

Question: how many samples do we need? In general, suppose we are getting samples form an unknown set with p fraction of elements having some property. How many samples is needed to estimate p with estimator satisfying $\tilde{p} = (1 \pm \epsilon)p$ with probability $1 - \delta$, that is, an (ϵ, δ) approximation.

Let's say we draw

- n samples and let Z be the number of samples with the property.
- We have $E[Z] = pn$.
- By Chenoff bound,

$$\begin{aligned} P[Z \ge pn + t] &\le e^{-2t^2/n} \\ &\le e^{-2\epsilon^2 p^2 n} && (with\ \ t = \epsilon pn) \\ &\Rightarrow\ \ needs\ n \ge \frac{1}{\epsilon^2 p^2} \log(\frac{1}{\delta})\ \ to\ obtain\ (\epsilon, \delta)\ \ approx. \end{aligned}$$

- When p is small, Chenoff gets a loose bound. Let's try Bernstein-style bound. By Theorem 6 of Lecture 6's note, since $Z = \sum_i Z_i$ with $Z_i \in [0, 1]$ and variance $p(1-p) \le p$, Z_i is subgamma with $(\sigma^2 = 2p, B = 1/2)$ and Z is subgamma with $(\sigma^2 = 2np, B = 1/2)$. Therefore, we have

$$P[Z \ge pn + t] \le \max\left\{ e^{-\frac{t^2}{4pn}},\ e^{-t/4} \right\}.$$

  Let $t \ge \sqrt{pn \log \frac{1}{\delta}} + \log \frac{1}{\delta}$, we have $1 - \delta$ probability that

$$Z \le pn \left(1 + \sqrt{\frac{\log \frac{1}{\delta}}{pn}} + \frac{\log \frac{1}{\delta}}{pn} \right) \ \Rightarrow\ Z \le pn\left(1 + \mathcal{O}(\epsilon)\right)\ \ if\ pn \ge \frac{1}{\epsilon^2} \log \frac{1}{\delta},$$

  which means we only need

$$n \ge \frac{1}{p\epsilon^2} \log \frac{1}{\delta}$$

  to get (ϵ, δ) approximation, a tighter result than that from Chenoff.

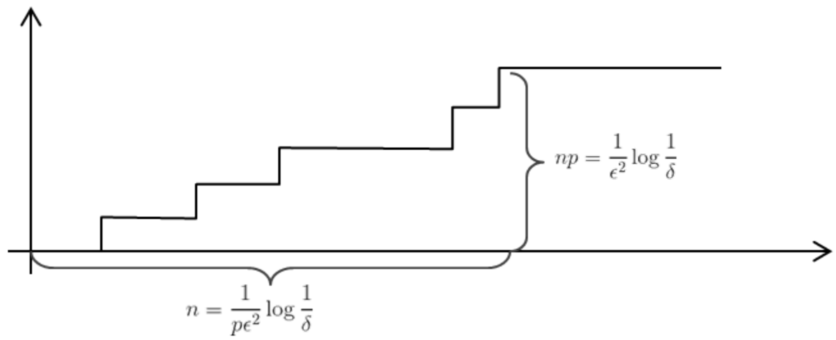

Figure 1: Dynamics of Z as n keeps increasing.

Question: What if we don't know p?

- After $n \geq \frac{1}{p\epsilon^2}\log\frac{1}{\delta}$ steps, we have $Z = pn(1 \pm \epsilon)$ true with prob. $1-\delta$.
- Now suppose we run the experiment until getting $\mu \approx \frac{1}{\epsilon^2}\log\frac{1}{\delta}$ number of hits, and we output

$$\tilde{p} = \frac{\mu}{\tilde{n}},$$

  where random variable $\tilde{n}$ is the number of samples we draw. We have $\tilde{n} = \frac{\mu}{\tilde{p}} \in \frac{\mu}{p}[\frac{1}{1+\epsilon}, \frac{1}{1-\epsilon}]$ with high probability based on previous analysis.

- On the other hand, from previous results (from Bernstein), for any n', we have

$$Z = pn'\left(1 \pm O(\sqrt{\frac{\log\frac{1}{\delta}}{n'p}})\right)$$

  with probability $1-\delta$. Now if $n' \in \frac{\mu}{p}[\frac{1}{1+\epsilon}, \frac{1}{1-\epsilon}]$, we have

$$Z = n'p\left(1 \pm \epsilon\frac{1}{\sqrt{1-\epsilon}}\right) = n'p(1 \pm \mathcal{O}(\epsilon))$$

  as desired.

2 Median finding

Given the item set $\{x_1, \ldots, x_n\}$, we want to find the median value as soon as possible. First of all, we can come up with the following methods:

1. Quick sorting : $O(n \log n)$.

   That is, choose random x_i and then sort $\{x_j | x_j \leq x_i\}$ and $\{x_k | x_k > x_i\}$.

2. Randomized select: $O(n)$

   Modify the Quicksort to find the i^{th} biggest item. That is, select x_i randomly and recursive on one side. This will run for $\log n$ rounds and the expectation of time $E[time] = n + 3/4n + (3/4)^2 + \ldots = O(n)$. Note that, for getting a concentration result, we need a bit more work.

3. Deterministic select: $O(n)$

   If we first partition the items into groups of 5 and then apply the same divide and conquer trick, we can get a deterministic algorithm with running time $O(n)$.

In the following, we will show

Theorem 1. *There is a random algorithm that can find the median in* $3/2n + o(n)$ *time with high probability.*

Proof. We design the claimed algorithm in the following three steps:

1. Sample s items from the n items where $s = \sqrt{n}$.

   Let S denote the set of these s sampled items. Then, we sort S and could find the $median(S) = S_m$. However, we cannot guarantee the S_m is the median of the original set with high probability.

2. Find $L, H \in S$ such that median $\in [L, H]$ with probability $1 - \delta$ and the number of elements in $[L, H]$ is $o(n)$.

   Instead of directly using the median of S_m, we consider about the $\sharp$elements in S that rank $\leq (1/2 - \alpha)n$ in the original set. Note that, by the discussion in the first section, we know

   $$Pr[Z \geq (1/2 - \alpha)s + t] \leq exp(-2t^2/s)$$

   Let $t = \alpha s$, then

   $$Pr[Z \geq 1/2s] \leq exp(-2\alpha^2 s).$$

   Namely, the point in sample with rank βs has rank within $(\beta \pm \sqrt{\frac{\log 1/\delta}{s}})n$ with probability $(1 - \delta)$. Thus, let $L = (1/2 - \sqrt{\frac{\log 1/\delta}{s}})s$ rank sample in S and $H = (1/2 + \sqrt{\frac{\log 1/\delta}{s}})s$ rank sample in S, which satisfies our requirement for the Step 2.

3. With the L, H, we can find the median by scanning the original set, which is stated as the Algorithm **??**.

As last, we consider about the running time. We first spend $s \log s = \sqrt{n} \log \sqrt{n} = o(n)$ on sorting the items in S. Then notice that with high probability $1 - \delta$, $|\{x_i | x_i < L\}|, |\{x_i | x_i > H\}| \leq n/2$ and $|mid| \leq 4\sqrt{s \log 1/\delta}$. For simplicity, we can let $\delta = 2^{-n^{2/3}}$, then $|mid| = o(n)$ with high probability. So know the running time of the Algorithm **??** is at most $3n/2 + |mid| \log |mid| = 3n/2 + o(n)$. Thus, in total, the running time of the whole random algorithm will be $3n/2 + o(n)$ with high probability $1 - 2^{-n^{2/3}}$.

$\square$

Algorithm 1: Scan the item set with L, H

Input: L, H and n items $x_1, \dots, x_n$
Output: The median of $\{x_1, \dots, x_n\}$

Initially, let $count_L = count_H = 0$ and $mid = []$;
for *each element* x_i **do**
  if $x_i \leq L$ **then**
    $count_L{+}{+}$;
    else if $x_i \geq H$ **then**
      $count_H{+}{+}$;
      else if $x_i \in [L, H]$ **then**
        Insert x_i into mid;
      end
    end
  end
end
Sort mid;
return the $(n/2 - count_L)^{th}$ smallest item in the mid.

3 Streaming Algorithm

- See v_1, v_2, ..., v_m.
- Let x_u = number of times i that $v_i = u$.
- n = number of different u.
- Both m, n are very large, and the goal is to get some statistics from v_1,...,v_m with $o(m)$, $o(n)$ space.

Example: find heavy hitters

For some α with $1/\alpha \ll m, n$, find S s.t.

$$\{u | x_u \geq \alpha m\} \subseteq S \subseteq \{u | x_u \geq \frac{\alpha}{2} m\}$$

with output space $|S| \leq \frac{1}{\alpha}$.

Suppose we know m, we can

1 Sample randomly, keep counters w/in samples.

2 Hash and discard when counters small. There is a well-known deterministic algorithm for this, called *Misra-Gries*, which is a generalization of the linear-time *Majority Algorithm* to find all items of frequency larger than m/k.

Misra-Gries **Algorithm** works as follows:

- At each stage, we maintain a map of at most $k-1$ pairs of (item,counter) as the $k-1$ candidates of frequent items. (Note we cannot have more than $k-1$ frequent items of frequency $> m/k$.).
- In the beginning, the map is initialized as empty.
- For each new incoming v_i,

  i if item v_i is in the map, increment its counter.

  ii Otherwise, if map size $< k-1$, add v_i into map with counter$=1$.

  iii Otherwise, decrement all the $k-1$ counters. Remove any pair with counter$=0$.
- Output the $k-1$ items in the map.

Properties of Misra-Gries

- It has false positives, but no false negative.
- Proof for no false negative: An item's counter gets decremented only when the map is full (with $k-1$ items in it), and then there are k items get decremented together (including the new incoming one decremented from 1 to 0). However, we have only a total counts of m, so there can be at most m/k such decrements. Therefore, an item of frequency $> m/k$ cannot be decremented to 0.
- We can remove false positive if having a second pass on v_1,...,v_m.

References