CS 591, Lecture 8
Data Analytics: Theory and Applications
Boston University

Babis Tsourakakis

February 15th, 2017

Five Puzzles

- Today we will go through five puzzle in a hands-on session (coding in python)
- These seemingly innocent puzzles have deep extensions
- In the next lecture we will see such extensions

Puzzle 1: Sample Binomial in Sublinear Expected Time

- You are given an array with $n = 10^9$ integers.
- You want to sample each entry with probability $p = \frac{3}{10^9}$.
- The straightforward approach is to toss a coin for each entry. We need n toin cosses
- To sample in expectation only 3 items, we toss 10^9 coins (i.e., run time $O(n)$)

Can we do better? We can choose 3 indices uar, but unfortunately this is wrong (why?)

Puzzle 1: Sample Binomial in Sublinear Expected Time

Yes! Idea:

- Let X be the number of coin tosses between two succesive successes.
- Question 1: What distribution does X follow?
- Question 2: How do we generate samples from that distribution, assuming access to $U[0,1]$?

Puzzle 1: Sample Binomial in Sublinear Expected Time

Yes! Implementation:

- Let X be the number of unsuccessful coin tosses until a successful toss.
- Answer 1: $\mathbf{Pr}\left[X = x\right] = (1-p)^{x-1}p$
- Answer 2: Sample $U \sim [0,1]$, and set $X \leftarrow \lceil \frac{\log U}{1-p} \rceil$
- Correctness:

$$(1-p)^{x-1} > U \geq (1-p)^{x} = (1-p)^{x-1} - (1-p)^{x} =$$
$$= (1-p)^{x-1}p.$$

- Expected run time now is $O(pn) \ll O(n)$, i.e., sublinear.

Puzzle 2: Size Estimation

- Universe of elements $U = \{1, \ldots, N\}$
- Problem: Select $X \subseteq U$ such that $|X| \approx n$ such that we can estimate from X the cardinality of any $S \subseteq U$.
- Any ideas for X?

Puzzle 2: Size Estimation

- Pick each element from U with probability $\frac{n}{N}$. (if $n \ll N$, then use Puzzle 1!)
- Define $X_i = 1$ if element $i \in U$ is chosen, o/w 0.
- $X = \sum_{i=1}^{N} X_i$
- $\mathbb{E}[X] = n$
- Any ideas for an unbiased estimator?

Puzzle 2: Size Estimation

- For a set S, define $\bar{S} = X \cap S$.
- Let $s_i = 1$ if $i \in S$

- RV: $Z = \frac{N}{n}|\bar{S}|$

- Z is an unbiased estimator

$$\mathbb{E}[Z] = \frac{N}{n}\sum_{i=1}^{N}\mathbb{E}[X_i s_i] = \frac{N}{n}\sum_{i=1}^{|S|}\mathbb{E}[X_i] = |S|.$$

- How well concentrated is Z to $|S|$?

Reminder

Multiplicative Chernoff bounds ($0 < \beta \leq 1$):

$$\boxed{\mathbf{Pr}\left[Z \leq (1-\beta)\mathbb{E}\left[Z\right]\right] \leq e^{-\frac{\beta^2}{2}\mathbb{E}[Z]}}$$

$$\boxed{\mathbf{Pr}\left[Z \geq (1+\beta)\mathbb{E}\left[Z\right]\right] \leq e^{-\frac{\beta^2}{3}\mathbb{E}[Z]}}$$

Puzzle 2: Size Estimation

- We apply the Chernoff on $\bar{S}$

$$\mathbf{Pr}\left[|\bar{S} - \mathbb{E}\left[barS\right]| \geq \epsilon \mathbb{E}\left[barS\right]\right] \leq 2\exp\left(-\frac{\epsilon^2}{3}\frac{n}{N}|S|\right)$$

- Since $Z = \frac{N}{n}\bar{S}$, we obtain

$$\mathbf{Pr}\left[|Z - \mathbb{E}\left[Z\right]| \geq \epsilon \mathbb{E}\left[Z\right]\right] \leq 2e^{-\frac{\epsilon^2}{3}|S|\frac{n}{N}} \leq \delta \Rightarrow$$
$$n \geq \frac{3}{\epsilon^2}\frac{N}{|S|}\log\left(\frac{1}{\delta}\right).$$

Data Streams: Algorithms and Applications *

S. Muthukrishnan†

Contents

Monograph by Muthu Muthukrishnan.

Puzzle 3: Finding One Missing Number

- Let $\pi \in S_n$, i.e., π is a permutation of $\{1, \ldots, n\}$
- Further, let π_{-1} be π with one element missing
- We cannot memoize the stream, we actually have only $O(\log n)$ bits available.
- How do find the missing number?

Puzzle 3: Finding Two Missing Number

- Let $\pi \in S_n$, i.e., π is a permutation of $\{1, \ldots, n\}$
- Further, let π_{-1} be π with one element missing
- We cannot memoize the stream, we actually have only $O(\log n)$ bits available.
- How do find the two missing number?

Puzzle 3: Finding Two Missing Number

- One missing number: We keep track of

$$\sum_{j \le i} \pi_{-1}[j].$$

- We output

$$\frac{n(n+1)}{2} - \sum_{j \le i} \pi_{-1}[j].$$

- Two missing numbers: We keep track of

$$s = \sum_{j \le i} \pi_{-2}[j], ss = \sum_{j \le i} \pi_{-2}[j]^2.$$

- We solve a system with two unknowns and two equations (which ones?).

Puzzle 4: Majority element

- Stream of n items
- Assume there exists a **majority element** x that appears $> \frac{n}{2}$ times
- How do we find x?
- Demo
- Majority element does not get "cancelled" out
- In our next lecture we will see how this simple idea generalizes (Misra-Gries algorithm [Misra and Gries, 1982])

Moore-Boyer algorithm

```
def find_majority(stream):
 majority = None
 counter = 0
 for item in stream:
   if item == majority:
      counter +=1
   else:
      if counter == 0 :
           majority = item
            counter = 1
      else:
            counter -=1
return majority
```

Puzzle 5: Reservoir Sampling

- Input:Stream of n items that does not fit in memory
- Goal: Keep a uniform random sample
- Simple: Draw a random integer $\{1, \ldots, n\}$, and select that item.
- Problem: We don't know n a priori
- Ideas?

Puzzle 5: Reservoir Sampling

- Keep first item in memory
- When i-th element arrives
  - with probability $\frac{1}{i}$ keep the new item
  - with probability $1 - \frac{1}{i}$ keep the old item

What if we want to keep $k \geq 2$ random elements?

references I

Misra, J. and Gries, D. (1982).
Finding repeated elements.
Science of computer programming, 2(2):143–152.