CMPSCI 611: Advanced Algorithms

Lecture 18: Data Streams and Count-Min Sketch

Andrew McGregor

Last Compiled: December 14, 2017

Missing Lemma from Last Time

Lemma

Let X be the number of heads observed when we toss m coins each with probability of heads equal to p. Then $\mathbb{P}[X \geq k] \leq \binom{m}{k} p^k$.

- Let $S_1, S_2, \ldots S_{\binom{m}{k}}$ be all subsets of $[m]$ with exactly k elements.

$$P(A_{S_j}) = p^k$$

- Then $A_{S_1} \cup A_{S_2} \cup \ldots \cup A_{S_{\binom{m}{k}}}$ is the event you get k or more heads.
- Hence,

$$P(k \text{ or more heads}) = P(A_{S_1} \cup A_{S_2} \cup \ldots \cup A_{S_{\binom{m}{k}}}) \leq \sum_{j=1}^{\binom{m}{k}} P(A_{S_j}) = \binom{m}{k} p^k$$

Outline

Data Stream Puzzles

- Suppose that a stream of numbers consists of all values between 1 and 1000000 except one value that is missing. Can you find the missing value without remembering the entire stream?
  - Keep track of the sum S of the values you observe.
  - The missing value is $\left(\sum_{i=1}^{1000000} i\right) - S$

Another Data Stream Puzzle

- Suppose you want to pick an entry in the stream uniformly at random but don't know the length of the stream in advance. Can you do this without remembering the entire stream?
  - Store the first value in the stream.
  - When you read the ith element of the stream, store this value with probability $1/i$ and throw away the value you were currently storing.
  - If the final length of the stream is m, then the probability you are storing the ith element is

$$\begin{aligned} & 1/i \times \left(1 - \frac{1}{i+1}\right) \times \left(1 - \frac{1}{i+2}\right) \times \ldots \left(1 - \frac{1}{m}\right) \\ = \;& 1/i \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \ldots \frac{m-1}{m} = 1/m \end{aligned}$$

A Final Data Stream Puzzle

- Suppose you observe a sequence of numbered balls being added or removed to a bucket that is initially empty. For example,

$$\text{add } 1, \quad \text{remove } 1, \quad \text{add } 6, \quad \text{add } 7, \quad \text{add } 6, \quad \text{remove } 7, \ldots$$

  and note that multiple balls can have the same number.
- When the sequence stops, can you determine whether all the balls in the bucket have the same number while only using a little memory?

The Solution

- As the stream is observed, maintain three counters m, t_1 and t_2.
- When we observe "add i":

$$m \leftarrow m+1 \qquad t_1 \leftarrow t_1 + i \qquad t_2 \leftarrow t_2 + i^2$$

- When we observe "remove i":

$$m \leftarrow m-1 \qquad t_1 \leftarrow t_1 - i \qquad t_2 \leftarrow t_2 - i^2$$

- If f_i is the number of times i is in the bucket then

$$m = \sum f_i \qquad t_1 = \sum i \times f_i \qquad t_2 = \sum i^2 \times f_i$$

- Theorem: All the balls have the same number if $(t_1/m)^2 = t_2/m$.

Proof of Theorem

- Consider the random variable X defined by picking a random ball in the bucket and letting X be the number of this ball. Note that

$$P(X = i) = f_i/m$$

- The expectation of X is:

$$\mathbb{E}\left[X\right] = \sum i \times \frac{f_i}{m} = t_1/m$$

and

$$\mathbb{E}\left[X^2\right] = \sum i^2 \times \frac{f_i}{m} = t_2/m$$

- X always takes the same value if $var(X) = 0$. Since

$$var(X) = E(X^2) - E(X)^2$$

X always takes the same value if

$$t_2/m = E(X^2) = E(X)^2 = (t_1/m)^2$$

Estimating Frequencies without Counting

Problem

You observe a long stream m of integers,

$$3, 5, 2, 9, 10, 101, 17, \ldots,$$

Because the stream is long and there are many integers, you can't remember all the values. At the end of the stream, there's a quiz with a single question:

How many times did the value "x" appear?

How well can you estimate the number of occurrences of x? The catch is that you don't know x in advance.

Denote the number of occurrences, or *frequency*, of x by f_x.

Count-Min Sketch

- Extension of Balls and Bins: For every element in the stream, we throw a ball into one of n bins but we insist that all balls corresponding to the same value land in the same bin, i.e.,

  for each element j in the stream, a ball lands in bin $h(j)$

  where $h(1), h(2), \ldots$ are random values in $\{1, 2, \ldots, n\}$
- The algorithm maintains a counter for each bin:

$$c_j = \text{number of balls in } j\text{th bin} = \sum_{y:h(y)=j} f_y$$

- Let c_i be our estimate for f_x where $i = h(x)$. Note

$$f_x \leq c_i \ .$$

- If $n = 2/\epsilon$, we shall show that with probability at least $1/2$

$$c_i \leq f_x + \epsilon m$$

Count-Min Analysis: Expected Error is Small

- Define random variable Z such that $c_i = f_x + Z$, i.e.,

$$Z = \sum_{y \neq x} f_y C_y$$

where $C_y = 1$ if $h(y) = i$ and 0 otherwise.

- Since h is random

$$\mathbb{E}[Z] = \sum_{y \neq x} f_y \mathbb{E}[C_y] = \sum_{y \neq x} f_y \mathbb{P}[h(y) = i] \leq m/n = \epsilon m/2$$

- By Markov inequality,

$$\mathbb{P}[Z \geq \epsilon m] \leq \frac{\epsilon m/2}{\epsilon m} = 1/2$$

and so $c_i \leq f_x + \epsilon m$ with probability at least $1/2$ as required.

Count-Min Analysis: Small Error With High Probability

- Suppose we want an estimate $\tilde{f}_x$ such that with probability 0.999,

$$f_x \leq \tilde{f}_x \leq f_x + m\epsilon \ .$$

- Repeat process $t = 14$ times in parallel and return smallest estimate.
- Say an estimate is "bad" if is exceeds $f_x + m\epsilon$.
- Probability all estimates are bad is $(1/2)^t \leq 0.0001$.
- Hence, smallest estimate is $\leq f_x + m\epsilon$ with probability at least 0.999.

Count-Min Sketch Applications

- **Computing popular products.** For example, A could be all of the page views of products on amazon.com yesterday. Using CM, we can are find the most frequently viewed products.
- **Computing frequent search queries.** For example, A could be all of the searches on Google yesterday. Using CM, one can find the searches made most often.
- **Identifying heavy TCP flows.** Here, A is a list of data packets passing through a network switch, each annotated with a source-destination pair of IP addresses. Using CM, one can find the flows that are sending the most traffic. This is useful for, among other things, identifying denial-of-service attacks.

k-Universal Hash Functions

Definition

Let $U = \{0, 1, \ldots, m-1\}$ and $V = \{0, 1, \ldots, n-1\}$ where $m \geq n$. A family of hash functions $\mathcal{H}$ from U to V is said to be k-universal if, for any distinct elements $x_1, \ldots, x_k \in U$ and for a hash function h chosen uniformly at random from $\mathcal{H}$, we have

$$\mathbb{P}\left[h(x_1) = h(x_2) = \ldots = h(x_k)\right] \leq \frac{1}{n^{k-1}} \ .$$

Example

Fix some prime $p \geq m$ and define

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod n \ .$$

The family $\mathcal{H} = \{h_{a,b} | 1 \leq a \leq p-1, 0 \leq b \leq p-1\}$ can be shown to be 2-Universal.

Proof of 2-Universality (Not Covered in Class)

- Fix $x_1 \neq x_2 \in U$. Since $|\mathcal{H}| = p(p-1)$, if we can show that at most $p(p-1)/n$ pairs (a, b) that satisfy $h_{a,b}(x_1) = h_{a,b}(x_2)$ then,

$$\mathbb{P}\left[h_{a,b}(x_1) = h_{a,b}(x_2)\right] \leq \frac{p(p-1)/n}{p(p-1)} = \frac{1}{n} \ .$$

- Let $u = (ax_1 + b \bmod p)$ and $v = (ax_2 + b \bmod p)$. Then (u, v) with $u \neq v$ uniquely specifies a and b:

$$a = \left(\frac{v-u}{x_2 - x_1} \bmod p\right) \quad \text{and} \quad b = (u - ax_1 \bmod p)$$

- Hence, we need to count how many pairs $0 \leq u, v \leq p-1$ satisfy $u \neq v$ and $u = v \bmod n$:

$$p((p-1)/n + 1 - 1) = p(p-1)/n \ .$$