29.1.2025 ***Adaptive data analysis***

Lecture 12: "Sampling"

Source: Lecture notes by Aaron Roth and Adam Smith

Lecturer: Uri Stemmer

Recall the following algorithm and theorem (from Lecture 10), that operate in the oblivious streaming model.

Reservoir Sampling [Vitter, 1985]:

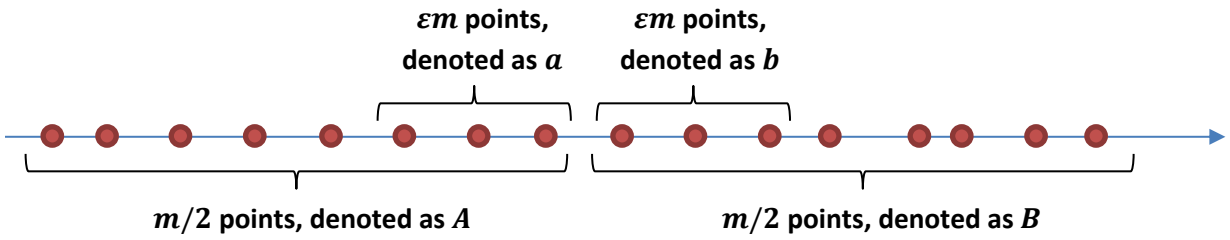
1. Init $\boldsymbol{s} = \perp$
2. When the next item u_i is read, toss a biased coin and with probability $\boldsymbol{1/i}$ let $\boldsymbol{s = u_i}$ (note that we need to maintain both the element $\boldsymbol{s}$ and a counter for $\boldsymbol{i}$)
3. Output $\boldsymbol{s}$

Theorem: The output of the above algorithm is an element uniformly sampled from the stream. Specifically, if the stream is of length $\boldsymbol{m}$, then for every $\boldsymbol{i} \in [\boldsymbol{m}]$ we have $\mathbf{Pr}[\boldsymbol{s = u_i}] = \frac{\boldsymbol{1}}{\boldsymbol{m}}$.

Recall that we can maintain a sample of size $\boldsymbol{k}$ by simply running $\boldsymbol{k}$ copies of this algorithm in parallel.

Observation: In the oblivious model, if we maintain a sample of size $\boldsymbol{k = O\left(\frac{1}{\varepsilon^2}\right)}$ then w.h.p. the median of the sample will be an $\boldsymbol{\varepsilon}$-approximation to the median of the entire stream.[1]

Proof sketch: Fix an input stream and consider the following plot where the red point correspond to the (sorted) elements in the stream:



Note that for a random sample from the stream, the probability of it falling in $\boldsymbol{A \cup b}$ or in $\boldsymbol{a \cup B}$ is exactly $\frac{\boldsymbol{1}}{\boldsymbol{2}} + \boldsymbol{\varepsilon}$. Thus, by Chernoff/Hoeffding, w.h.p., a sample of size $\boldsymbol{k \gtrsim \frac{1}{\varepsilon^2}}$ contains more than $\frac{\boldsymbol{k}}{\boldsymbol{2}}$ points from $\boldsymbol{A \cup b}$ and more than $\frac{\boldsymbol{k}}{\boldsymbol{2}}$ points from $\boldsymbol{a \cup B}$. Thus, the sample median cannot be outside of $\boldsymbol{a \cup b}$. To see this, note that the sample median cannot be to the left of $\boldsymbol{a}$ because then there are more than $\frac{\boldsymbol{k}}{\boldsymbol{2}}$ sample points to the right of it.

[1] The term "approximate" here means that the median of the sampled set will be an element whose order among the elements of the full stream, when the elements are sorted by value from smallest to largest, is within the range $\boldsymbol{(1 \pm \varepsilon)m/2}$ with high probability.

What happens in the adaptive/adversarial setting?

AdaptiveSamplingGame(A, M)

1. Instantiate the sampling algorithm $\boldsymbol{M}$ and let $\boldsymbol{\sigma_0}$ denote its initial internal state
2. The adversary $\boldsymbol{A}$ gets $\boldsymbol{\sigma_0}$
3. For $\boldsymbol{i = 1, 2, \dots, m}$:
   - The adversary $\boldsymbol{A}$ chooses the next item in the stream $\boldsymbol{u_i}$
   - The mechanism $\boldsymbol{M}$ gets $\boldsymbol{u_i}$ and updates its internal state to $\boldsymbol{\sigma_i}$
   - The adversary $\boldsymbol{A}$ gets $\boldsymbol{\sigma_i}$
4. At the end of the stream, the sampling algorithm $\boldsymbol{M}$ outputs a set of $\boldsymbol{k}$ elements

Will Reservoir Sampling "work" in the adaptive case in the sense that the sample median approximates the stream median?

Let $\boldsymbol{k}$ denote the sample size and let $\boldsymbol{k'}$ be the total number of elements that were sampled during the reservoir sampling process. That is, $\boldsymbol{k'}$ counts sampled elements that were possibly evicted at a future iteration. We bound $\boldsymbol{k'}$ as follows:

$$\mathbb{E}[\boldsymbol{k'}] = \sum_{i=1}^{m} \frac{\boldsymbol{k}}{\boldsymbol{i}} = \boldsymbol{k} \sum_{i=1}^{m} \frac{\boldsymbol{1}}{\boldsymbol{i}} \underbrace{\leq}_{\textbf{harmonic sum}} \boldsymbol{k}(\ln(\boldsymbol{m}) + \boldsymbol{1}) \leq \boldsymbol{2k} \ln \boldsymbol{m}$$

Thus, by Markov's inequality, with probability at least **0.9**, we will have $\boldsymbol{k' \leq 20k \ln m}$. We next present an attack in which these $\boldsymbol{k'}$ elements are the smallest elements in the stream. The final sample set consists of some $\boldsymbol{k}$ elements among these $\boldsymbol{k'}$ elements. Hence, when the stream is of length $\boldsymbol{m \gg 2k'}$, the sample median does not approximate the stream median.

The attack: Consider a setting where the stream consists of $\boldsymbol{k}$ points $\boldsymbol{x_1, \dots, x_n}$ in the one-dimensional range $[\boldsymbol{0, 1}]$. The adversary keeps a "working range" at any point during the game, starting with the full range $[\boldsymbol{0, 1}]$. In the first round, the adversary chooses the number $\boldsymbol{x_1 = 0.5}$ as the first element in the stream. If $\boldsymbol{x_1}$ is sampled, then the adversary moves to the range $[\boldsymbol{0.5, 1}]$, and otherwise, to the range $[\boldsymbol{0, 0.5}]$. Next, the adversary submits $\boldsymbol{x_2}$ as the middle of the current range. This continues for $\boldsymbol{m}$ steps.
Note that at any point throughout the process, the adversary always submits an element that is larger than all elements in the current sampled set, and also smaller than all the non-sampled elements of the stream. Therefore, the end result is that after this process is over, the $\boldsymbol{k'}$ sampled elements are precisely the smallest $\boldsymbol{k'}$ elements in the stream.

An adaptive "sampling" algorithm for the median

Define a "merge" operation on two sets $\boldsymbol{A, B}$ of size $\boldsymbol{k}$ each: Sort $\boldsymbol{A \cup B}$ and return the set $\boldsymbol{C}$ obtained by deleting every second item in the ordering. We think of $\boldsymbol{A \cup B}$ as a *multiset* containing $\boldsymbol{2k}$ items. Note that $\boldsymbol{|C| = k}$.

Also note that for every number $\boldsymbol{t}$ it holds that

$$\boldsymbol{2\sum_{x\in C} 1_{\{x\le t\}} \in \sum_{x\in A\cup B} 1_{\{x\le t\}} \pm 1 \in \sum_{x\in A\cup B} 1_{\{x\le t\}} \pm \varepsilon|A\cup B|} \qquad \boldsymbol{((1))}$$

Where the last transition follows by requiring $\boldsymbol{k \ge \frac{2}{\varepsilon}}$, where $\boldsymbol{\varepsilon}$ is the approximation parameter.

Consider a complete binary tree whose nodes correspond to buckets of size $\boldsymbol{k}$ that are initially empty. Let $\mathbf{p}$ be a pointer to the "current leaf" of the tree, initiated as the left-most leaf. Throughout the execution, the elements in the stream are added to $\mathbf{p}$'s bucket until it is full (containing $\boldsymbol{k}$ elements), at which point we move to the next leaf (meaning that $\mathbf{p}$ now points to the next leaf). Whenever two sibling nodes in the tree are full, we "merge" them and place the resulting set of size $\boldsymbol{k}$ in their parent node.

Let $\boldsymbol{S}$ denote the input stream and let $\boldsymbol{m = |S|}$ be the length of the stream. Note that in this case our tree has $\boldsymbol{m/k}$ leaves and its height is $\boldsymbol{\log\frac{m}{k}}$.

Claim: Let $\boldsymbol{\vec{B} = (B_1, \dots, B_j)}$ denote the buckets in level $\boldsymbol{\ell}$ of the tree (where the leaves are at level $\mathbf{0}$). Then for every number $\boldsymbol{t}$ we have

$$\boldsymbol{2^{\ell}\cdot\sum_{x\in\vec{B}} 1_{\{x\le t\}} \in \sum_{x\in S} 1_{\{x\le t\}} \pm \varepsilon m\ell}$$

Proof: The proof is by induction on $\boldsymbol{\ell}$. The base-case for $\boldsymbol{\ell = 0}$ is trivial. Suppose that the claim holds for $\boldsymbol{\ell - 1}$. For every bucket $\boldsymbol{B_i}$ in level $\boldsymbol{\ell}$, let $\boldsymbol{B_i^0, B_i^1}$ denote its two children (which are buckets at level $\boldsymbol{\ell - 1}$). Then,

$$\boldsymbol{2^{\ell}\cdot\sum_{x\in\vec{B}} 1_{\{x\le t\}} = 2^{\ell-1}\left(2\sum_{x\in B_1} 1_{\{x\le t\}} + \cdots + 2\sum_{x\in B_j} 1_{\{x\le t\}}\right)}$$

$$\boldsymbol{\underbrace{\le}_{\text{by }((1))} 2^{\ell-1}\left(\left(\sum_{x\in B_1^0\cup B_1^1} 1_{\{x\le t\}} + \varepsilon\left|B_1^0\cup B_1^1\right|\right) + \cdots + \left(\sum_{x\in B_j^0\cup B_j^1} 1_{\{x\le t\}} + \varepsilon\left|B_j^0\cup B_j^1\right|\right)\right)}$$

$$\boldsymbol{= 2^{\ell-1}\left(\sum_{x\in B_1^0\cup B_1^1} 1_{\{x\le t\}} + \cdots + \sum_{x\in B_j^0\cup B_j^1} 1_{\{x\le t\}}\right) + 2^{\ell-1}\varepsilon\cdot\left(\left|B_1^0\cup B_1^1\right| + \cdots + \left|B_j^0\cup B_j^1\right|\right)}$$

$$\underbrace{\leq}_{* \text{ see below}} \quad 2^{\ell-1}\left(\sum_{x\in B_1^0\cup B_1^1} 1_{\{x\le t\}} + \cdots + \sum_{x\in B_j^0\cup B_j^1} 1_{\{x\le t\}}\right) + 2^{\ell-1}\varepsilon\cdot\frac{m}{2^{\ell-1}}$$

$$= \quad 2^{\ell-1}\left(\sum_{x\in B_1^0\cup B_1^1} 1_{\{x\le t\}} + \cdots + \sum_{x\in B_j^0\cup B_j^1} 1_{\{x\le t\}}\right) + \varepsilon\cdot m$$

$$\underbrace{\leq}_{\text{induction}} \quad \sum_{x\in S} 1_{\{x\le t\}} + \varepsilon m(\ell-1) + \varepsilon\cdot m$$

$$= \quad \sum_{x\in S} 1_{\{x\le t\}} + \varepsilon m\ell$$

The transition marked with $*$ follows from the fact that $|B_1^0\cup B_1^1| + \cdots + |B_j^0\cup B_j^1|$ is the width of level $(\ell-1)$. There are $\frac{m/k}{2^{\ell-1}}$ buckets in this level, each of size k, so overall $|B_1^0\cup B_1^1| + \cdots + |B_j^0\cup B_j^1| = \frac{m}{2^{\ell-1}}$.

This completes the proof of the claim.

Now let R denote the bucket in the root of the tree (the level of the root is $\ell = \log(m/k)$). By the above claim, for every number t it holds that

$$2^{\log(m/k)}\cdot\sum_{x\in R} 1_{\{x\le t\}} \in \sum_{x\in S} 1_{\{x\le t\}} \pm \varepsilon m\cdot\log(m/k)$$

That is,

$$\frac{1}{k}\sum_{x\in R} 1_{\{x\le t\}} \in \frac{1}{m}\sum_{x\in S} 1_{\{x\le t\}} \pm \varepsilon\cdot\log(m/k)$$

Therefore, by setting $k \gtrsim \frac{1}{\varepsilon}$ we ensure that the root provides a good approximation to the median of the stream (we need to set k to be slightly bigger than $\frac{1}{\varepsilon}$ in order to cancel out this $\log(m/k)$ factor).

Question: What is the space complexity of this algorithm?

Balls into bins

Suppose we throw $\boldsymbol{n}$ balls into $\boldsymbol{n}$ bins uniformly at random.
The expected load of every bin is 1. A simple application of the Chernoff bound shows that w.h.p. the maximal load would be $\boldsymbol{O(\log n)}$. In fact, classical results w.r.t. this problem show that w.h.p. the maximal load is bounded by $\boldsymbol{O\left(\frac{\log n}{\log\log n}\right)}$.

Proof (even for 2n balls and n bins):

The probability that bin 1 receives at least M balls is at most

$$\binom{2n}{M}\left(\frac{1}{n}\right)^M < \left(\frac{e2n}{M}\right)^M \cdot \left(\frac{1}{n}\right)^M = \left(\frac{2e}{M}\right)^M$$

This follows from a union bound; there are $\binom{2n}{M}$ distinct sets of M balls, and for any set of M balls the probability that all land in bin 1 is $\left(\frac{1}{n}\right)^M$.

Applying a union bound again allows us to find that, for $M \geq \frac{6\ln n}{\ln\ln n}$, the probability that any bin receives at least M balls is bounded above by

$$2n\left(\frac{2e}{M}\right)^M \leq 2n\left(\frac{2e\ \ln\ln n}{6\ln n}\right)^{\frac{6\ln n}{\ln\ln n}} \leq 2n\left(\frac{\ln\ln n}{\ln n}\right)^{\frac{6\ln n}{\ln\ln n}} = e^{\ln(2n)} \cdot \left(e^{\ln\ln\ln n - \ln\ln n}\right)^{\frac{6\ln n}{\ln\ln n}}$$

$$= e^{\ln(2n)} \cdot e^{6\ln n \cdot \ln\ln\ln n\ /\ \ln\ln n} \cdot e^{-6\ln n} \leq \frac{1}{n}$$

for n sufficiently large.

Now suppose we play the following adaptive game with an adversary:

$\mathbf{AdaptiveBallsBins}(\boldsymbol{n})$

1. Randomly throw $\boldsymbol{n}$ balls into $\boldsymbol{n}$ bins
2. The adversary $\boldsymbol{A}$ gets to see the resulting allocation
3. For $\boldsymbol{i = 1, 2, \ldots, n/2}$:
   - The adversary $\boldsymbol{A}$ chooses a bin $\boldsymbol{B_i}$, and gains the reward $\boldsymbol{r_i = \mathbf{Load}(B_i)}$
   - The $\boldsymbol{B_i}$ is now excluded from the rest of the game and its balls are thrown randomly among the remaining bins
   - The adversary $\boldsymbol{A}$ gets to see the resulting allocation
4. The total reward of the adversary is $\boldsymbol{r_1 + r_2 + \cdots + r_{n/2}}$

In the oblivious setting, the sequence of bin deletions is fixed before the game begins. In this case, for every step $\boldsymbol{i} \in [\boldsymbol{n}]$ of the game, we have that the balls are distributed uniformly across the remaining bins, and thus the maximal load is bounded by $\boldsymbol{O}\left(\frac{\log n}{\log\log n}\right)$ w.h.p., meaning that the total reward of the adversary is bounded by $\boldsymbol{O}\left(\boldsymbol{n}\cdot\frac{\log n}{\log\log n}\right)$. Actually, the total reward would be even smaller (since when the adversary eliminates the maximum-load bin then the second-highest bin would have significantly lower load).

What happens in the adaptive setting? Can the adversary somehow effect the distribution of the balls in a way that allows him to significantly increase its total reward?

Let us reconsider the oblivious case when the sequence of bin deletions is fixed before the game begins. To simplify the notations, let us assume that the order of bin deletions is simply Bin 1, Bin 2, Bin 3, …, Bin n/2.

Let us focus on a single ball $\boldsymbol{b}$ and bound the reward it generates during its movements. As we have fixed the order of bin-deletions, the reward generated by ball $\boldsymbol{b}$ is exactly the number of times it is thrown into one of the bins 1,2,…,n/2. This is a Geometric RV with success probability 0.5.

Thus, the total reward across all $\boldsymbol{n}$ balls throughout the execution is distributed as the sum of $\boldsymbol{n}$ Geometric RV's with parameter 0.5 (this is known as the "Negative binomial" distribution). We will use the following concertation bound:

Theorem: Let $\boldsymbol{GS(n,p)}$ be a negative binomially distributed random variable which is the sum of $\boldsymbol{n}$ i.i.d. geometrically distributed random variables, each with success probability $\boldsymbol{p}$. Then, for $\boldsymbol{k>1}$

$$\mathbf{Pr}\left[\boldsymbol{GS(n,p)} > \frac{\boldsymbol{k\cdot n}}{\boldsymbol{p}}\right] \leq \mathbf{exp}\left(-\frac{\boldsymbol{k(1-1/k)^2}}{\boldsymbol{2}}\cdot\boldsymbol{n}\right)$$

Using this theorem, for our fixed ordering, we get that

$$\Pr\left[\begin{matrix}\text{total}\\ \text{reward}\end{matrix} > 16n\log n\right] = \Pr\left[GS(n,0.5) > \frac{8\cdot\ln(n)\cdot n}{0.5}\right] \leq \exp\left(-\frac{8\cdot\ln(n)\left(1-\frac{1}{4\ln(n)}\right)^2}{2}\cdot n\right)$$

$$\leq \exp(-2\cdot n\cdot\ln(n)) \qquad ((*))$$

That is, for every fixed ordering of the deletions of the bins, with very high probability it holds that the total reward is bounded by $O(n\cdot\log n)$.

How many orders could there be? At most $n! \leq n^n$.

Note that that failure probability we derived in $((*))$ is so low that it allows us to "union bound" over all possible orderings. But what does this mean exactly? What is the probability space that ties all of these orderings together?

We consider a simultaneous game against all possible choices of the adaptive adversary using a rooted game tree $\boldsymbol{T}$. Each node in $\boldsymbol{T}$ represents an assignment of the balls into the bins which have not yet been deleted. The root represents the initial assignment of the balls into the $\boldsymbol{n}$ bins. A node $\boldsymbol{v}$ which has $\boldsymbol{\ell}$ remaining bins (at depth $\boldsymbol{n-\ell}$ of $\boldsymbol{T}$) has $\boldsymbol{\ell}$ children, each corresponds to a possible decision of the adversary (delete the first among remaining bins, the second, etc). Each edge from $\boldsymbol{v}$ to a child $\boldsymbol{u}$ is associated with sufficiently many random bits to redistribute the balls in the bin which the adversary deletes when it faces the configuration associated with $\boldsymbol{v}$. Each leaf corresponds to a configuration with $\boldsymbol{n/2}$ bins containing all the balls. The tree has at most $\boldsymbol{n}!$ distinct leaves. The path from the root to a leaf corresponds to an ordering of the $\boldsymbol{n/2}$ deleted bins (representing a complete deletion order of the bins).

The probability space of this simultaneous game corresponds to sampling all of the random bits on all the edges of the tree $\boldsymbol{T}$. Once we flip all these bits then the simultaneous game is completely determined and in particular the reward associated with each leaf (deletion ordering) is determined.

The bound we derived in $((*))$ bounds the probability that the recourse is high at a specific leaf. By a union bound,

$$\Pr\left[\begin{array}{c}\text{There is an order}\\ \text{of deletion with}\\ \text{reward greater}\\ \text{than } 16n\,log\,n\end{array}\right] \leq n! \cdot \exp(-2 \cdot n \cdot \ln(n)) \leq \exp(n\,ln(n) - 2n\,ln(n)) = \exp(-n\ln(n))$$

Now, if the reward is small in every leaf then the adversary cannot gain a high reward.