CS 591, Lecture 9
Data Analytics: Theory and Applications
Boston University

Babis Tsourakakis

February 22nd, 2017

Announcement

- We will cover the Monday's 2/20 lecture (President's day) by appending half an hour to next Monday's and Wednesday's lectures.
- Next week's office hours for next week: **4.15-5.00pm**

Reminder: Reservoir Sampling

- **Input:** Stream of n items that does not fit in memory $\langle x_1, \ldots, x_n \rangle$

- Goal: Keep a uniform random sample

- Simple: Draw a random integer $\{1, \ldots, n\}$, and select that item.

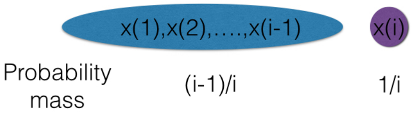
- Issue: We don't know the length n of the stream a priori

- Problem: Find a uniform sample s from a stream of unknown length

Algorithm: Reservoir Sampling

1. Initially we set $s \leftarrow x_1$
2. When i-th element arrives, we set $s \leftarrow x_i$ with probability $\frac{1}{i}$

Intuition:



Correctness: What is the probability that $s = x_i$ at some time $t \geq i$?

Algorithm: Reservoir Sampling

Correctness: What is the probability that $s = x_i$ at some time $t \geq i$?

$$\mathbf{Pr}\left[s = x_i\right] = \frac{1}{i} \times \left(1 - \frac{1}{i+1}\right) \times \ldots \times \left(1 - \frac{1}{t}\right) = \frac{1}{t}.$$

Question: What about the space complexity?

Extension: Weighted Reservoir Sampling

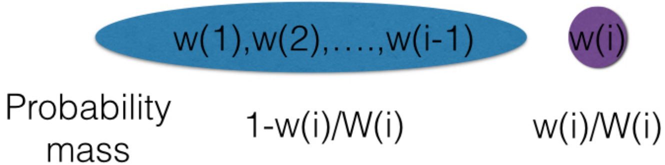
- **Input:** Stream of $\langle w_1, w_2, \ldots \rangle$
- Goal: Select index i with probability proportional to w_i
- Again, we don't know the length n of the stream a priori
- Define for each index i

$$W_i = w_1 + \ldots w_i.$$

- Any ideas?

Extension: Weighted Reservoir Sampling

Intuition:



Algorithm: Weighted Reservoir Sampling

1. Initially we set $s \leftarrow x_1$
2. When i-th element arrives, we set $s \leftarrow x_i$ with probability $\frac{w_i}{W_{i-1}+w_i}$

Correctness:

On seeing w_{i+1}:

- We switch to w_{i+1} with probability $\frac{w_{i+1}}{W_{i+1}}$
- If we don't switch the winner $j^* \leq i$ remains with probability

$$\boxed{\mathbf{Pr}\left[s = w_{j^*}\right] = \left(1 - \frac{w_{i+1}}{W_{i+1}}\right)\frac{w_{j^*}}{W_i} = \frac{w_{j^*}}{W_{i+1}}.}$$

Question: What about the space complexity?

Reservoir Sampling, k items

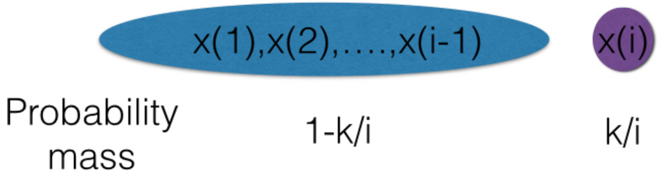
- **Input:**
  Stream of items $\langle x_1, x_2, \ldots, \rangle$
  Parameter $k \geq 1$

- Goal: Keep k uniform random samples

- Ideas?

Extension: Weighted Reservoir Sampling

Intuition:



$$\# \text{ k-sets that do not contain } i = \frac{\binom{i-1}{k}}{\binom{i}{k}} = 1 - \frac{k}{i}.$$

Python implementation

```python
import random

def reservoir_sampling(stream,k):
  S =[]
  counter = 1
  for x in stream:
        if(counter<=k):
            S.append(x)
        else:
            c = random.randint(0,counter-1);
            if( c<k):
                S[c] = x
        counter +=1
  return S
```

Probabilistic Counting

- Researchers at Bell Labs needed to count overlapping substrings of three letters.
- Why? To develop statistic-based spell-checker. This spellchecker (TYPO) was included in early Unix distributions
- Robert Morris came up with an elegant way of counting each item approximately [Morris, 1978]
- Why approximately? 8-bit counters can hold counts up to 255 (too small range)...

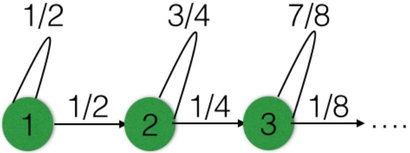
Probabilistic Counting

- Increments performed in a probabilistic manner
- Suppose we use constant probabilities, i.e., we increase the counter from i to $i+1$ with probability p
- Why is this problematic?
- Morris' idea: probability $\mathbf{Pr}[i \to i+1] = f(i)$ depends on the actual counter value
- Outline: Morris $\to$ Morris+ $\to$ Morris++[1].

[1] We will follow Jelani Nelson's exposition (M,M+,M++).

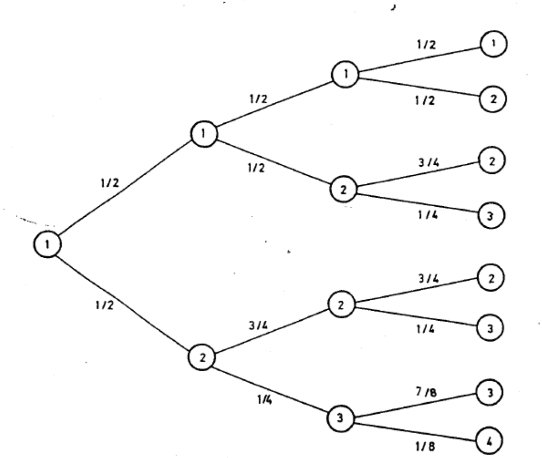
Morris' algorithm

1. Initialize $X \leftarrow 0$.
2. For each event, increment X with probability $\frac{1}{2^X}$.
3. Output $\tilde{n} = 2^X - 1$.



Notice that our data structure just maintains an integer! We will see that this integer grows as $\sim \log n$, hence we need $O(\log \log n)$ bits.

Example: Morris' algorithm



Source: [Flajolet, 1985]

$$\mathbf{Pr}\left[\tilde{n}=1\right]=\frac{8}{64},\ \mathbf{Pr}\left[\tilde{n}=2\right]=\frac{38}{64},$$
$$\mathbf{Pr}\left[\tilde{n}=3\right]=\frac{17}{64},\ \mathbf{Pr}\left[\tilde{n}=4\right]=\frac{1}{64}.$$

Morris' algorithm: Analysis

- $X_n :=$ Morris' counter after n updates
- Let's compute $\mathbb{E}\left[2^{X_n}\right]$.
- Idea 1: Let's apply the formula

$$\boxed{\mathbb{E}\left[2^{X_n}\right] = \sum_{l=0}^{n} 2^l p_{n,l}.}$$

- We have to compute $\mathbf{Pr}\left[X_n = l\right] = p_{n,l}$

Morris' algorithm: Analysis

- Suppose we reached state l via n_1 transitions from state 1 to state 1, n_2 from state 2 to state 2, ..., n_l from state l to l

$$(1-2^{-1})^{n_1}2^{-1}(1-2^{-2})^{n_2}2^{-2}\ldots(1-2^{-(l-1)})^{n_{l-1}}2^{-(l-1)}(1-2^{-l})^{n_l}.$$

with the condition that $n_1+\ldots+n_l+l-1=n$.

- **Tedious approach...**

- Let's try an **inductive** approach.

Morris' algorithm : Analysis

1. Base case. It's obviously true for $n = 0$.
2. Induction step.

$$
\begin{aligned}
\mathbb{E}\left[2^{X_{n+1}}\right] &= \sum_{j=0}^{\infty} \mathbf{Pr}\left[X_n = j\right] \cdot \mathbb{E}\left[2^{X_{n+1}} | X_n = j\right] \\
&= \sum_{j=0}^{\infty} \mathbf{Pr}\left[X_n = j\right] \cdot (2^j(1 - \frac{1}{2^j}) + \frac{1}{2^j} \cdot 2^{j+1}) \\
&= \sum_{j=0}^{\infty} \mathbf{Pr}\left[X_n = j\right] \cdot 2^j + \sum_{j} \mathbf{Pr}\left[X_n = j\right] \\
&= \mathbb{E}\left[2^{X_n}\right] + 1 \\
&= (n+1) + 1
\end{aligned}
$$

Morris' algorithm : Analysis

- In a similar way, we can show that

$$\mathbb{E}\left[2^{2X_n}\right] = \frac{3}{2}n^2 + \frac{3}{2}n + 1.$$

- Therefore

$$\mathbb{V}ar\left[2^{X_n}\right] < \frac{1}{2}n^2.$$

- **Chebyshev's inequality**:

$$\mathbf{Pr}\left[|\tilde{n} - n| \geq \epsilon n\right] < \frac{1}{2\epsilon^2}.$$

Details

For the second moment, again we use induction:

$$\begin{aligned}
\mathbb{E}\left[2^{2X_n}\right] &= \sum_{j\geq 0} 2^{2j}\mathbf{Pr}\left[X_n = j\right] \\
&= \sum_{j\geq 0} 2^{2j}\Big(\mathbf{Pr}\left[X_{n-1} = j\right]\left(1-2^{-j}\right) \\
&+ \mathbf{Pr}\left[X_{n-1} = j-1\right]2^{-(j-1)}\Big) = \dots \\
&= \mathbb{E}\left[2^{2X_{n-1}}\right] + 3\mathbb{E}\left[2^{X_{n-1}}\right] \\
&= \frac{3}{2}(n-1)^2 + \frac{3}{2}(n-1) + 1 + 3n \\
&= \frac{3}{2}n^2 + \frac{3}{2}n + 1
\end{aligned}$$

Morris' algorithm

counter value	7	8	9	10	11	12	13
probability	0.0011	0.0602	0.3424	0.4218	0.1538	0.0195	0.0001

Source: [Flajolet, 1985]

- Exact probability distribution for Morris' counter for $n = 1024$
- Variance is not bad, but we want to do better! **Morris+**

Morris+ algorithm

- We instantiate s independent copies of Morris and average their outputs.

- $s = \frac{1}{2\epsilon^2\delta}$
- Our estimate becomes

$$\tilde{n} = \frac{1}{s}\sum_{i=1}^{s} \tilde{n}_i.$$

- Now **Chebyshev's inequality** gives

$$\mathbf{Pr}\left[|\tilde{n} - n| \geq \epsilon n\right] < \frac{1}{2s\epsilon^2} < \delta.$$

Next step is Morris++!

Morris++ algorithm

- Run $t = 36 \lg \frac{1}{\delta}$ copies of **Morris+** with failure probability $\delta = \frac{1}{3}$
- Output the median estimate from the t copies of **Morris+**
- In other words, we have t coin tosses, each results in 1 with probability $\frac{1}{3}$, and in 0 with probability $\frac{2}{3}$.
- Failure if median is 0, otherwise success $\rightarrow$ **Chernoff**!

Morris++ algorithm analysis

$$Y_i = \begin{cases} 1, & \text{if } i\text{-th Morris+ fails.} \\ 0, & \text{otherwise.} \end{cases}$$

By Chernoff bound,

$$\boxed{\mathbf{Pr}\left[\sum_i Y_i > \frac{t}{2}\right] \leq \mathbf{Pr}\left[\sum_i Y_i - \mathbb{E}\left[\sum_i Y_i\right] > \frac{t}{6}\right] < \delta.}$$

Morris++ estimate $\tilde{n}$ $(1 \pm \epsilon)$-approximates n with probability at least $(1 - \delta)$.

Question: What is the space complexity that we achieved?

references I

Flajolet, P. (1985).
Approximate counting: a detailed analysis.
BIT Numerical Mathematics, 25(1):113–134.

Morris, R. (1978).
Counting large numbers of events in small registers.
Communications of the ACM, 21(10):840–842.