COMS E6998-9: Algorithmic Techniques for Massive Data — Sep 22, 2015

Lecture 5 – Precision Sampling (cont), Streaming for Graphs

Instructor: *Alex Andoni* — Scribe: *Nikhil Mitra*

Plan

- Precision Sampling (continuation)
- Streaming for Graphs

1 Precision Sampling (continuation)

1.1 Recap

Last time we proved the prescision sampling lemma that we can (with 90% success) get O(1) additive error and 1.5 multiplicative error :

$$\frac{s}{1.5} - O(1) < \tilde{s} < 1.5s + O(1) \tag{1}$$

with average cost equal to O($\log n$)
The algorithm:

- Draw each u_i randomly from Exp(1), a probability distribution described roughly by e^{-x}
- Return $\widetilde{S} = \max_i \widetilde{a}_i / u_i$

1.2 Proof of correctness

$$max(\frac{a_i}{u_i}) \sim \sum \frac{a_i}{Exp(1)} \tag{2}$$

Therefore,

$$max(\frac{\tilde{a}_i}{u_i}) \sim \frac{\sum a_i}{Exp(1)} \pm 1 \tag{3}$$

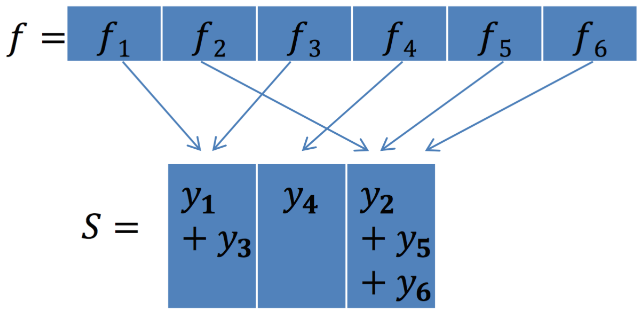
2 p-moment via Prescision Sampling

2.1 Theorem

Linear Sketch for the p-moment with O(1) approximation and $O(n^{1-2/p})\log^{O(1)} n$ space (with 90% success probability) can be found

2.2 Sketch

- Pick random $r_i \in \{\pm 1\}$ and $u_i \sim Exp(1)$
- Let $y_i = f_i \cdot r_i / u_i^{1/p}$
- Hash into Hash Table S



The width of table S is w, where

$$w = O(n^{1-2/p}) \log^{O(1)} n \tag{4}$$

2.3 Estimator

Use the estimator $\max_j |S[j]|^p$

2.4 Algorithm

`Initialize(w):`
    `array S[w]`
    `hash functions` h`, into` $[w]$
    `hash functions` R`, into` $[\pm 1]$
    `reals` u_i`, from Exp distribution`

`Process(vector` $f \in \Re^n$`):`
    `for (i = 0; i < n; ++j)`
        `S[`h_i`] +=` $\frac{f_i r_i}{{u_i}^{1/p}}$

`Estimator:`
    $\max_j |S[j]|^p$

2.5 Correctness of Estimator

Theorem 1. $\max_j |S[j]|^p$ *is O(1) approximation with 90% probability, with* $w = O(n^{1-2/p}) \log^{O(1)} n$ *cells*

Proof. To prove the theorem, we use precision sampling lemma.

$$a_i = |f_i|^p \tag{5}$$

$$\sum a_i = \sum |f_i|^p \tilde{a}_i/u_i = |S(H(i))|^p \tag{6}$$

Now we need to show that $|a_i - \tilde{a}_i|$ is small. More prescisely speaking, we need to prove

$$\left| \frac{\tilde{a}_i}{u_i} - \frac{a_i}{u_i} \right| \leq \epsilon F_p \tag{7}$$

Claim 2. $|S(H(i))^p - f_i^p/u_i| < O(\epsilon F_p)$

Proof. Consider cell $z = h(i)$

$$s(z) = \frac{f_i r_i}{{u_i}^{1/p}} + C \tag{8}$$

Where $r_i = \pm 1$ and u_i is an exponential random variable. Now we need to estimate the value of C i.e how much chaff is there.

Let $y(i) = \frac{f_i r_i}{{u_i}^{1/p}}$

$$y(i) = \frac{f_i r_i}{{u_i}^{1/p}} \tag{9}$$

$$C = \sum_{j \neq i} y_i \cdot \chi[h(j) = z] \tag{10}$$

Now E[C] = 0, and hence does not serve as a good indicator of value. $E[C] = 0$ because $r_i = \pm 1$. Therefore, we find out $E[C^2]$

$$E(C^2) \quad = \quad E\left(\sum_{j1,j2 \neq i} Y_{j1} Y_{j2} \cdot \chi\left[h(j1) = z\right] \cdot \chi\left[h(j2) = z\right] \right) \tag{11}$$

Now all elements where $j1 \neq j2$ disappear.

$$E(C^2) = E\left(\sum_{j1,j2 \neq i} Y_j^2 \cdot \chi^2 \left[h(j) = z\right] \right) \tag{12}$$

$$= E\left(\sum_{j1,j2 \neq i} Y_j^2 \frac{1}{w} \right) \tag{13}$$

$$\leq \frac{\|y\|^2}{w} \tag{14}$$

Now,

$$E\left[\sum y_j^2\right] = E\left[\sum_j \frac{{f_j}^2}{{u_j}^{1/p}}\right] \tag{15}$$

$$= \sum_j {f_j}^2 E\left[\frac{1}{{u_j}^{1/p}}\right] \tag{16}$$

Now, using the concavity property of the function,

$$\leq \sum_j {f_j}^2 (\log n)^{1/p} \tag{17}$$

Now, Using Holder's inequality, we can write

$$\|f\|^2 \leq n^{1-2/p} \|f\|_p^2 \tag{18}$$

Using Markov's, we can write

$$C^2 \leq \|f\|_p^2 \cdot n^{1-2/p} \cdot O(\log n)/w \tag{19}$$

Setting $w = \frac{1}{\epsilon^{2/p}} n^{1-2/p} \cdot O(\log n)$

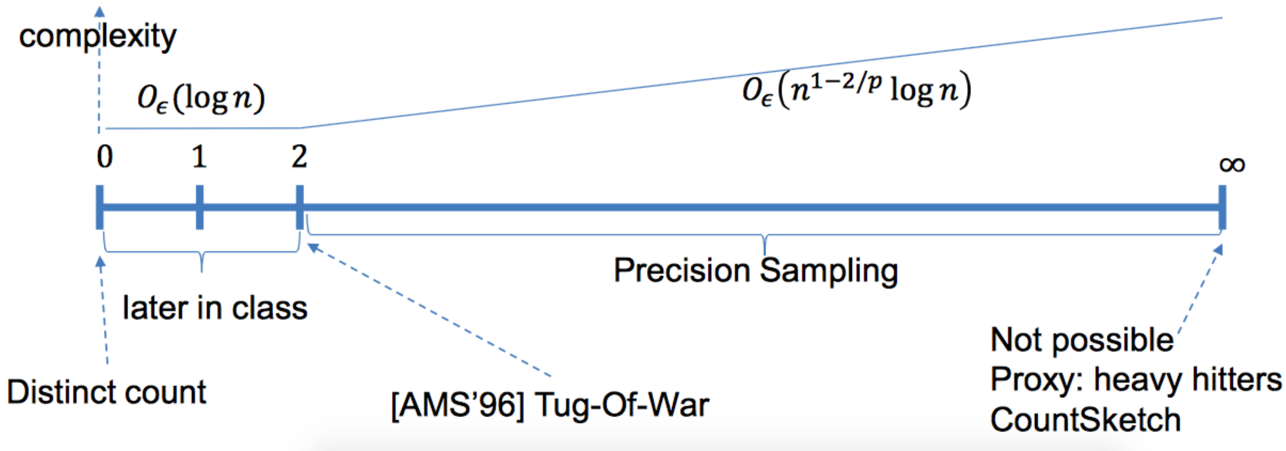
$$|C|^p < \|f\|_p^p = \epsilon F_p \tag{20}$$

□

□

2.6 Recap

- We claimed $|S(H(i))^p - f_i^p/u_i| < O(\epsilon F_p)$
- $S(H(i))^p = \left(\frac{f_i}{{u_i}^{1/p}} + C\right)^p$, where $C = \sum_{j \neq i} y_i \cdot \chi[h(i) = h(j)]$
- We proved $E(C^2) \leq \frac{\|y\|^2}{w}$. This implies that $|C|^p < \epsilon F_p$ with 90% probability for a fixed i. But we need it for all i.
- What we want is a $|C|^2 < \beta \|y\|^2 / w$ with high probability for a smallish w. We can indeed prove that $\beta = O(\log^2 n)$ using a strong concentration inequality (Bernstein)



3 Streaming for Graphs

3.1 Graphs

Suppose we have a graph G with n vertices and m edges. Suppose the data stream is of the list of edges. There are multiple such examples where such graphs are used to model the data. For example,

- Web
- Socialgraphs
- Phonecalls
- Maps
- Geographical data etc.

3.2 Why Streaming for Graphs

Suppose we have a graph G which consists of a large number of vertices and even larger number of edges. It is typical to store such graphs on the hard drive. The usual algorithms for graphs (e.g. breadth first search) are typically random access.If we used a streaming algorithm, we would need to do a linear scan through all the edges. For typical hard drives, linear scan is much more efficient than random access. Further, most of the usual graph algorithms use random access. Some problems associated with graphs are

- Connectivity
- Distances (similarities) between nodes
- PageRank (stationary distribution of random walk)
- Counting # of triangles (measure of clusterability) Various other statistics
- Matchings
- Graph partitioning

3.3 Parameters for graph algorithms

The usual aim of streaming graph algorithms is to use $O(n)$ space or $O(n \log n)$. An $O(n \log n)$ algorithm would still take much less space than an $O(m)$ algorithm, because m can be of the order of n^2.

The usual space bound for streaming algorithms, which is sublinear, is generally not possible.

3.4 Problem 1: Connectivity

The problem is stated as, Given a graph G, check wether the graph G is connected. Using a streaming approach, we can do it on $O(n)$ space.

The basic idea is to use a minimum spanning tree of the graph. The algorithm can be stated as

- Keep a subgraph H (starts empty)
- When we see an edge (i,j), if this edge does not create a cycle in H, then add it to H

Space: $\leq n-1$ edges only.

This subgraph H can be used to

- find connectivity between 2 nodes
- number of connected components

3.5 Problem 2: Distance

The problem can be stated as, given a graph G and 2 nodes s,t in it, find the distance between them upto and approximation of α, where α is an odd integer.

We can do this with slight modification to the previous algorithm.

- Keep a subgraph H
- When we encounter edge (i,j), if $d_h(i,j) > \alpha$, then add it to H.

With regards to the space complexity, we can see that all cycles in H have length $\alpha + 2$, then according to Bollobas' theorem, $|H| \leq O(n^{1+\frac{2}{\alpha+1}})$

Theorem 3. *(Bollobas) If all cycles in a graph are of length $\alpha + 2$, then $|H| \leq O(n^{1+\frac{2}{\alpha+1}})$*

Proof. For a simplified case, let us assume all nodes have degree d.

- Suppose $\alpha = 2k - 1$
- Explore a vertex v.

- At depth k, all nodes differ. Therefore,

$$d^k \leq n \tag{21}$$

$$d = n^(1/k) \tag{22}$$

$$m \leq n^{1+1/k} \tag{23}$$

$$= n^{1+\frac{2}{1+\alpha}} \tag{24}$$

$\square$