Streaming & sketching Algorithms

- large growth in data gathering & processing
- Often the setting allows limited access to data or one has limited computation power (remote sensing)
- we see some tools / techniques to extract information from data and/or compress data without much loss

Topics

Streaming: There are situations that the data set is too large to be stored or it may be impossible to process all the data at the same time.
We might have access to one (or a few) pass over data (like on a conveyor belt that passes the data)
We have to process the data as it arives & make decisions

Sketching / Sampling: The goal is to have a compressed form of data which we can still use to answer queries and extract useful information.

Dimensionality reduction: In many applications data comes with very high dimensions think thousands of features (e.g. medical data, spam filtering). Designing algorithms to manage high dimensions is difficult.
One goal is to reduce dimension (e.g. projection) while preserving (approximately) relevant structure / geometry.

Property testing: Checking quickly w.h.p. whether a given

Property testing: Checking quickly w.h.p. whether a given object has certain properties (e.g. if a matrix multiplication is correct, if a large graph has certain properties, etc).

Sparse Fourier Transform: There are old algorithms to compute discrete Fourier transform of a seq. of length n in time $O(n \log n)$. Various applications (signal & image processing, multiplications of large integers, etc). Sparse Fourier transform can compute DFT when the output is sparse faster.

Approximate Counting

Let's start with the simple example of of counting (approximately) the number of events.

Suppose we have a long seq. of events and we want to have a counter to simply count the # of them.

- Clearly if we have n events we need $O(\log n)$ bits

Easy to see any deterministic algorithm needs this much.

- Goal: use much less than $O(\log n)$.

Given ε & δ, if the actual answer is n & $\tilde{n}$ is the estimate we return we want:

$$\Pr\left[|\tilde{n} - n| > \varepsilon n\right] < \delta$$

We call this an (ε, δ)-estimator. Morris '78 came up with a simple (ε, δ)-estimator usin $\tilde{O}(\log\log n)$ bits

we can turn an (ε, δ)-estimator. Point is to come up with a simple (ε, δ)-estimator using $\tilde{O}(\log\log n)$ bits

Morris Approximate Counting

$X \leftarrow 0$

For each new event increment X with prob. $\frac{1}{2^X}$

return $\tilde{n} = 2^X - 1$

Analysis: let X_n be the rand. var. representing X after n steps and let $Y_n = 2^{X_n}$

Lemma: $E[Y_n] = n+1$

proof: Induction on n. $n=0$ easy

$$E[Y_{n+1}] = \sum_{i=0}^{\infty} \Pr[X_n = i] \cdot E[2^{X_{n+1}} \mid X_n = i]$$

$$= \sum_{i=0}^{\infty} \Pr[X_n = i] \left(2^i\left(1 - \frac{1}{2^i}\right) + \frac{1}{2^i} \cdot 2^{i+1}\right)$$

$$= \sum_{i=0}^{\infty} \Pr[X_n = i] 2^i + \sum_{i=0}^{\infty} \Pr[X_n = i]$$

$$= E[Y_n] + 1 \quad = (n+1) + 1$$

So the output $\tilde{n} = 2^X - 1$ is an estimate of n in expectation

Also, since $E[Y_n] = n+1$, $E[X_n] = \log_2(n+1)$ and so # of bits used after n steps is $O(\log\log n)$.

Lemma: $E[Y_n^2] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$ and $\mathrm{Var}[Y_n] = \frac{n(n-1)}{2}$

proof: Again we use induction. $n=0$ ✓

$$E[Y_{n+1}^2] = \sum_{i=0}^{\infty} 2^{2i} \Pr[X_{n+1} = i]$$

$$E[Y_{n+1}^2] = \sum_{i=0}^{\infty} 2^{2i} \Pr[X_{n+1} = i]$$

$$= \sum_{i=0}^{\infty} 2^{2i} \left(\Pr[X_n = i](1 - \frac{1}{2^i}) + \Pr[X_n = i-1] \frac{1}{2^{i-1}} \right)$$

$$= \sum_{i=0}^{\infty} 2^{2i} \Pr[X_n = i] + \sum_{i=0}^{\infty} (-2^i \Pr[X_n = i] + 4 \times 2^{i-1} \Pr[X_n = i-1])$$

$$= E[Y_n^2] + 3E[Y_n]$$

$$= \frac{3}{2}n^2 + \frac{3}{2}n + 1 + 3(n+1)$$

$$= \frac{3}{2}(n+1)^2 + \frac{3}{2}(n+1) + 1$$

Also, $\mathrm{Var}[Y_n] = E[Y_n^2] - E[Y_n]^2 = \frac{n(n-1)}{2}$

Success probability:

using Chebyshev's inequality

$$\Pr[|\tilde{n} - n| > \varepsilon n] < \frac{1}{(\varepsilon n)^2} \cdot \frac{n(n-1)}{2} \approx \frac{1}{2\varepsilon^2}$$

but this is useless for small ε.

Morris+: Using average to boost probability

Suppose we run r parallel copies of Morris algorithm and find values $\tilde{n}_i$ for $1 \le i \le r$ and let $\tilde{n} = \frac{1}{r} \sum_{i=1}^{r} \tilde{n}_i$

Note that $\tilde{n}$ is an estimator for n.

$$\Pr[|\tilde{n} - n| > \varepsilon n] \le \frac{1}{2r\varepsilon^2} < \delta \quad \text{if we choose } r > \frac{1}{2\varepsilon^2\delta}$$

The amount of space used is $O(\log\log n / \varepsilon^2\delta)$ bits.

Note that with $r > \frac{2}{\varepsilon^2}$ we have $\Pr[|\tilde{n} - n| > \varepsilon n] < \frac{1}{4}$

Morris++: Using median to boost probability:

We can do even better: instead of using average we use the median.

Run $\ell = c \log \frac{1}{\delta}$ parallel copies of Morris+ for some constant c. Suppose we get estimates $Z_1 \ldots Z_\ell$ and let $\tilde{n}$ be the median of them.

Note that by arguments for Morris+, for each i:

$$\Pr[|Z_i - n| > \varepsilon n] < \frac{1}{4}$$

So if we have random var $Y_i = \begin{cases} 1 & \text{if } |Z_i - n| > \varepsilon n \\ 0 & \text{o.w.} \end{cases}$

then Y_i's are independent & $\Pr[Y_i] < 1/4$ and

$E[\sum Y_i] < \ell/4$.

Note: We have $|\tilde{n} - n| > \varepsilon n$ only if at least $\frac{\ell}{2}$ of Z_i's are larger than n by $> \varepsilon n$, i.e. $\sum Y_i \geq \frac{\ell}{2}$

Using Chernoff bound:

$$\Pr[|\tilde{n} - n| > \varepsilon n] \leq \Pr\left[\sum Y_i \geq \frac{\ell}{2}\right] \leq \Pr\left[\left|\sum Y_i - E[\sum Y_i]\right| > \frac{\ell}{4}\right]$$

($E[\sum Y_i] < \frac{\ell}{4}$)

$$\leq \Pr\left[\sum Y_i \geq 2\frac{\ell}{4}\right] \leq \left(\frac{e}{2^2}\right)^{\ell/4}$$

with $\ell = c \log \frac{1}{\delta}$ this is $< \delta$ for large enough c.

Total space is $O\left(\frac{\log \frac{1}{\delta}}{\varepsilon^2} \log \log \left(\frac{n}{\varepsilon\delta}\right)\right)$

Estimating Frequency Moments

Suppose we have a stream of data $\sigma = a_1, a_2, \ldots a_m$ where $a_i \in \{1, \ldots, n\}$.

We define frequency vector $f = (f_1, \ldots, f_n)$ for σ where f_i is the # of times i appears in σ

e.g. if $\sigma = 3, 2, 1, 2, 3, 4, 5, 3, 5, 3, 2, 4, 5, 3, 2, 2$

then $f = (1, 5, 5, 2, 3)$

Definition: Given a sequence σ, we define the k'th moment of σ, $F_k(\sigma)$, to be: $F_k(\sigma) = \sum_{i=1}^{n} f_i^k$

We also define $0^0 = 0$. Then:

$k=0$, $F_k(\sigma) = \sum_{i=0}^{n} f_i^0 = \sum_{i: f_i > 0} 1$ = # of distinct elements in σ

$k=1$, $F_k(\sigma) = \sum_{i=1}^{n} f_i = m$ = # of elements

$k=\infty$, define $F_\infty(\sigma) = \max$ frequency $= \max_i f_i$

We have already seen Morris++ for $k=1$.

Next we see algorithms for $k=0$; # of distinct elements in σ. if $d = |\{i : f_i > 0\}|$ it can be no deterministic alg can solve this in sublinear space.

Recall Hashing:

k-universal: A family $\mathcal{H}$ of hash functions $h: U \to T$ is (strongly) k-universal if for any k distinct elements

$x_1, \ldots, x_k \in U$ and k values $a_1, \ldots, a_k \in T$

$$\Pr\left[\bigwedge_{i=1}^{k} h(x_i) = a_i\right] = \frac{1}{n^k} \quad \text{where } |T| = n$$

(these are also call k-wise independent)

Def: A collection of rand. var's $X_1 \ldots X_n$ are k-wise independent if for any subset of k i

$$\Pr[X_{i_1} = a_1 \wedge X_{i_2} = a_2 \wedge \cdots X_{i_k} = a_k] = \prod_{j=1}^{k} \Pr[X_{i_j} = a_j]$$

Note: if $X_1 \ldots X_n$ are pair-wise independent and $Y = \sum_{i}^{n} X_i$ then

1) $\mathrm{Var}[Y] = \sum_{i=1}^{n} \mathrm{Var}[X_i]$

2) if $X_1 \ldots X_n$ are Bernoulli (i.e. 0/1) then

$$\mathrm{Var}[Y] \leq \sum_{i=1}^{n} E[X_i^2] = \sum_{i=1}^{n} E[X_i] = E[Y]$$

Proof: left as excercise!

AMS Algorithm for $F_0(\sigma)$

We describe an algorithm for estimating # of distinct elements. This is due to Alon/Matias/Szegedy and base on earlier work by Flajolet/Martin.

big picture idea: put each distinct item randomly between $0 \ldots 1$; on expectation if there are t distinct elements then interval $[0, 1]$ is divided into parts of size roughly $\frac{1}{t+1}$. We use a 2-universal hash func. to place items randomly.

to place items randomly.

To make this work formally we need a few def.

Def: For any $t \in \{1 \ldots n\}$ let Zero(t) be the largest power of 2 it is divisible by (i.e. # of zero's in the right)

(e.g. Zero(1024) = 10, Zero(23) = 0, Zero(48) = 4, etc)

First assume we have sufficiently many numbers and they are uniformly distributed. if we have d distinct elements if we look at Zero(x) for each x in the stream we expect one from d to have Zero(x) $\geq \log d$

In order to ensure we have (almost) uniform dist. we use hashing.

AMS - distinct elements estimator

1) let H be a 2-universal hash family $[n] \to [n]$ and pick $h \in H$
2) $z \leftarrow 0$
3) while there are elements in σ do
   pick next element a_i
   $z \leftarrow \max\{z, \text{zero}(h(a_i))\}$
4) return $2^{z+\frac{1}{2}}$

let S be the set of distinct elements in σ

Define $X_{r,i} = \begin{cases} 1 & \text{if } \text{zero}(h(i)) \geq r \\ 0 & \text{o.w.} \end{cases}$ $\forall\, i \in \{1 \cdots n\}$

let $Y_r = \sum_{i \in S} X_{r,i} = |\{x \in \sigma : \text{zero}(h(x)) \geq r\}|$

So Y_r is the # of distinct elements with at least r zero's (from right) in the binary rep. of their $h(x)$.

$$E[X_{r,i}] = Pr[\text{zero}(h(i)) \geq r] = Pr[2^r \text{ divides } h(i)]$$

since $h(i)$ is u.r. in $[n]$ $= \frac{n/2^r}{n} = \frac{1}{2^r}$

Therefore $E[Y_r] = \sum_{i \in S} E[X_{r,i}] = \frac{d}{2^r}$

Assuming (d is a power of 2): $E[Y_{\log d}] = \frac{d}{2^{\log d}} = 1$

Now we compute variance of Y_r. Since h is 2-universal we have 2-wise ind. Thus

$$Var[Y_r] = \sum_{i \in S} Var[X_{r,i}] \leq \sum_{i \in S} E[X_{r,i}^2] = \sum_{i \in S} E[X_{r,i}] \leq \frac{d}{2^r}$$

Using Chebyshev's inequality:

$$Pr[Y_r > 0] = Pr[Y_r \geq 1] \leq E[Y_r] \leq \frac{d}{2^r}$$

$$Pr[Y_r = 0] = Pr\left[|Y_r - E[Y_r]| > \frac{d}{2^r}\right] \leq \frac{Var[Y_r]}{(d/2^r)^2} \leq \frac{2^r}{d}$$

Suppose z' is the last value of z in the alg.

So output is $2^{z'+\frac{1}{2}} = \tilde{d}$. We claim $\tilde{d}$ is close to d.

$a+\frac{1}{2}$

So output is $2^{\frac{1}{2}+z} = \tilde{d}$. We claim $\tilde{d}$ is close to d.

let a be the smallest integer s.t. $2^{a+\frac{1}{2}} \geq 3d$

$$\Pr[\tilde{d} \geq 3d] \leq \Pr[z' \geq a] = \Pr[Y_a > 0] \leq \frac{d}{2^a} \leq \frac{\sqrt{2}}{3}$$

Similarly let b be the largest int. s.t. $2^{b+\frac{1}{2}} \leq \frac{d}{3}$. Then

$$\Pr[\tilde{d} \leq \frac{d}{3}] \leq \Pr[z' \leq b] = \Pr[Y_{b+1} = 0] \leq \frac{2^{b+1}}{d} \leq \frac{\sqrt{2}}{3}$$

We claim the alg returns a value $\frac{d}{3} \leq \tilde{d} \leq 3d$ w.p. $1 - \frac{2\sqrt{2}}{3}$ ≈ 0.057

So we have a $(3, 0.057)$-estimator

We can repeat this $O(\log \frac{1}{\delta})$ times and pick the median of average and obtain $(3, 1-\delta)$-estimator

This takes $O(\log n \log \frac{1}{\delta})$ space.