Topics in Streaming

Lecture 18 and 19
October 27 and 29, 2020

Topics in Streaming

- F_p estimation for $p \in (0, 2]$ via p-stable distributions and pseudorandom generators
- Priority Sampling
- Precision Sampling and Applications to ℓ_2 sampling in streams
- ℓ_0 Sampling

Part III

Sampling according to frequency moments

Sampling

Sampling problem: given $x \in \mathbb{R}^n$ in ~~(strict)~~ turnstile setting, at the end output random (I, R) where $I \in [n]$ and $R \in \mathbb{R}$ such that $\Pr[I = i] \simeq \frac{|x_i|^p}{\sum_j |x_j|^p}$ and $R = x_i$ if $I = i$.

$\frac{|x_i|^2}{\|x\|_2^2}$

$x = (0, 0, \ldots, 0)$

+2

−3

−10

+100

ℓ_2

ℓ_p $p \in (0, 2)$

ℓ_0

??

$x = (-1, 0, 10, 0.1, \ldots)$

Sampling

Sampling problem: given $x \in \mathbb{R}^n$ in (strict) turnstile setting, at the end output random (I, R) where $I \in [n]$ and $R \in \mathbb{R}$ such that $\Pr[I = i] \simeq \frac{|x_i|^p}{\sum_j |x_j|^p}$ and $R = x_i$ if $I = i$.

Sampling is generally a more challenging problem than estimation

Sampling

Sampling problem: given $x \in \mathbb{R}^n$ in (strict) turnstile setting, at the end output random (I, R) where $I \in [n]$ and $R \in \mathbb{R}$ such that $\Pr[I = i] \simeq \frac{|x_i|^p}{\sum_j |x_j|^p}$ and $R = x_i$ if $I = i$.

Sampling is generally a more challenging problem than estimation

Approximation: $\Pr[I = i] = (1 \pm \epsilon)\frac{|x_i|^p}{\sum_j |x_j|^p} + 1/\text{poly}(n)$ for some small ϵ and $R = (1 \pm \epsilon)x_i$.

Sampling

Sampling problem: given $x \in \mathbb{R}^n$ in (strict) turnstile setting, at the end output random (I, R) where $I \in [n]$ and $R \in \mathbb{R}$ such that $\Pr[I = i] \simeq \frac{|x_i|^p}{\sum_j |x_j|^p}$ and $R = x_i$ if $I = i$.

Sampling is generally a more challenging problem than estimation

Approximation: $\Pr[I = i] = (1 \pm \epsilon)\frac{|x_i|^p}{\sum_j |x_j|^p} + 1/\text{poly}(n)$ for some small ϵ and $R = (1 \pm \epsilon)x_i$.

Can do ℓ_0, ℓ_2 and ℓ_p for $0 < p < 2$ in polylog space using ideas from sketching. Works in (strict) turnstile models.

Several important applications

Part IV

ℓ_2 Sampling

$x = (1, -3, 10, 5, 0, 3)$

$w_1 \quad w_2 \quad \cdots \quad w_n$

$(1, 9, 100, 25, 0, 9).$

$\|x\|_2^2 = \sum (1 + 9 + 100 \cdots).$ $\quad$ x_c^2

$2 \text{ with } \dfrac{9}{\|x\|_2^2}$ $\qquad$ $i \quad \dfrac{w_i}{W}$ $\quad \|x\|_2^2$

ℓ_2 Sampling

Based on precision sampling which has similarities to priority sampling.

High-level Algorithm:

- $x = (x_1, x_2, \ldots, x_n)$ is the vector being updated
- Can estimate $\|x\|_2$ using F_2 estimation. Assume $\|x\|_2 = 1$ for normalization purposes/simplicity
- Consider $y = (y_1, y_2, \ldots, y_n)$ where $y_i = x_i/\sqrt{u_i}$ where $u_1, u_2, \ldots, u_n$ are independent random variables from $[0, 1]$.
- For some threshold t to be chosen, return (i, x_i^2) if i is the *unique* index such that $y_i^2 \geq t$.

Questions:

- How should we choose t? Why does it work?
- How do we implement in streaming setting?

Choosing threshold

Let $w_i = x_i^2$ and hence we have $w_1, w_2, \ldots, w_n$ and $W = \sum_i w_i = \|x\|_2^2$. Normalize such that $W = 1$

Recall priority sampling where we pick $u_1, \ldots, u_n \in [0, 1]$ independently and store the largest k amongst w_i/u_i values. Here we think of storing only largest. Also $y_i^2 = x_i^2/u_i = w_i/u_i$

Choosing threshold

Let $w_i = x_i^2$ and hence we have $w_1, w_2, \ldots, w_n$ and $W = \sum_i w_i = \|x\|_2^2$. Normalize such that $W = 1$

Recall priority sampling where we pick $u_1, \ldots, u_n \in [0, 1]$ independently and store the largest k amongst w_i/u_i values. Here we think of storing only largest. Also $y_i^2 = x_i^2/u_i = w_i/u_i$

Fix threshold t. What is probability that i is returned?

$$\Pr\left[y_i^2 \geq t\right] \prod_{j \neq i} \Pr\left[y_j^2 < t\right] = \frac{x_i^2}{t} \prod_{j \neq i} \left(1 - \frac{x_j^2}{t}\right).$$

If t large then above is $\simeq \frac{x_i^2}{t}$

Probability some item is output is $\simeq \frac{1}{t}$. Hence repeat $\Omega(t \log(1/\delta))$ times to ensure output with prob at least $(1 - \delta)$.

$$\prod_{j \neq i} \left(1 - \frac{x_j^2}{t}\right) \approx e^{-\frac{x_j^2}{t}}.$$

$$\sum e^{-\frac{\sum x_j^2}{t}} \qquad t \geq 100 w.$$

$$e^{-\frac{(w - w_i)}{t}}$$

$x = (\_,\ \ldots\ \_)$

$y = (\ \ , \frac{x_i}{\sqrt{u_i}}, \ldots)$

$x_i \leftarrow x_i + \Delta_i$

$y_i \leftarrow y_i + \frac{\Delta_i}{\sqrt{u_i}}$

Choosing threshold and identifying i

t should be large compared to $\sum_i x_i^2 = \|x\|_2^2$. Probability of output is $1/t$ so need t attempts. Thus choose $t = O(\log n)\|x\|_2^2$.

Choosing threshold and identifying i

t should be large compared to $\sum_i x_i^2 = \|x\|_2^2$. Probability of output is $1/t$ so need t attempts. Thus choose $t = O(\log n)\|x\|_2^2$.

Need to store $y_1^2, y_2^2, \ldots, y_n^2$?

Choosing threshold and identifying i

t should be large compared to $\sum_i x_i^2 = \|x\|_2^2$. Probability of output is $1/t$ so need t attempts. Thus choose $t = O(\log n)\|x\|_2^2$.

Need to store $y_1^2, y_2^2, \ldots, y_n^2$? But we only need the two largest to decide if largest is above threshold. Hence can use Count Sketch on y to store only heavy hitters.

Choosing threshold and identifying i

t should be large compared to $\sum_i x_i^2 = \|x\|_2^2$. Probability of output is $1/t$ so need t attempts. Thus choose $t = O(\log n)\|x\|_2^2$.

Need to store $y_1^2, y_2^2, \ldots, y_n^2$? But we only need the two largest to decide if largest is above threshold. Hence can use Count Sketch on y to store only heavy hitters.

Issues:

- Count Sketch gives heavy hitters with additive error that depends on $\|y\|_2$.
- Threshold t is with respect to $\|x\|_2^2$.
- How do we store independent $u_1, \ldots, u_n$ to sketch y?

Resolving issues

Note that $y_i^2 \geq x_i^2$ for all i, hence $\|y\|_2^2 \geq \|x\|_2^2$.

Lemma

With probability $\geq (1 - \delta)$ *we have* $\|y\|_2^2 \leq \frac{1}{\delta} c \ln n \|x\|_2^2$ *for some fixed* c.

Prove above as exercise. Thus $\|y\|_2$ is not much larger than $\|x\|_2$.

$y = \left(\frac{x_1}{\sqrt{u_1}}, \frac{x_2}{\sqrt{u_2}}, \ldots, \frac{x_n}{\sqrt{u_n}} \right).$

$y_i^2 = \frac{x_i^2}{u_i} \geq x_i^2 \qquad u_i \in (0,1).$

$\|y\|_2 \geq \|x\|_2$

Resolving issues

Note that $y_i^2 \geq x_i^2$ for all i, hence $\|y\|_2^2 \geq \|x\|_2^2$.

Lemma

With probability $\geq (1-\delta)$ *we have* $\|y\|_2^2 \leq \frac{1}{\delta} c \ln n \|x\|_2^2$ *for some fixed* c.

Prove above as exercise. Thus $\|y\|_2$ is not much larger than $\|x\|_2$.

Recall Count Sketch for y gives estimate $\tilde{y}_i$ for each i such that $|\tilde{y}_i - y_i|^2 \leq \epsilon^2 \|y\|_2^2$ and space is $O(\frac{1}{\epsilon^2} \log n)$. Choose $\epsilon = \epsilon' / \log n$ and hence we have $|\tilde{y}_i - y_i|^2 \leq \frac{\epsilon'^2}{\log n} \|x\|_2^2$

$$|\tilde{y}_i - y_i|^2 \leq \frac{(\epsilon')^2}{(\log n)^2} \|y\|_2^2 \leq \frac{(\epsilon')^2}{(\log n)^2} \cdot \|x\|_2^2$$

Resolving issues

Note that $y_i^2 \geq x_i^2$ for all i, hence $\|y\|_2^2 \geq \|x\|_2^2$.

Lemma

With probability $\geq (1 - \delta)$ *we have* $\|y\|_2^2 \leq \frac{1}{\delta} c \ln n \|x\|_2^2$ *for some fixed* c.

Prove above as exercise. Thus $\|y\|_2$ is not much larger than $\|x\|_2$.

Recall Count Sketch for y gives estimate $\tilde{y}_i$ for each i such that $|\tilde{y}_i - y_i|^2 \leq \epsilon^2 \|y\|_2^2$ and space is $O(\frac{1}{\epsilon^2} \log n)$. Choose $\epsilon = \epsilon' / \log n$ and hence we have $|\tilde{y}_i - y_i|^2 \leq \frac{\epsilon'}{\log n} \|x\|_2^2$

Above implies that $\tilde{y}_i$ is a close mutiplicative approximation of y_i if y_i is sufficiently large compared to $\|x\|_2^2$

Resolving issues

Recall threshold $t = c \log n \|x\|_2^2$. Implies that

- Sufficient to keep track of small number of heavy hitters in y hence Count Sketch for y needs only $\text{poly}(\log n/\epsilon^2)$ space.
- Can keep track of $\|x\|_2$ and $\|y\|_2$ to check if heavy hitters are sufficiently large and hence estimates are accurate even if additive error
- Output i if $\tilde{y}_i^2 \geq t$ and is unique.

Resolving issues

Recall threshold $t = c \log n \|x\|_2^2$. Implies that

- Sufficient to keep track of small number of heavy hitters in y hence Count Sketch for y needs only $\text{poly}(\log n/\epsilon^2)$ space.
- Can keep track of $\|x\|_2$ and $\|y\|_2$ to check if heavy hitters are sufficiently large and hence estimates are accurate even if additive error
- Output i if $\tilde{y}_i^2 \geq t$ and is unique.

Since we use $\tilde{y}_i$ which is an estimate of y_i, the probability of i being output is proportional to $\frac{(1\pm\epsilon)x_i^2}{\|x\|_2^2}$.

Resolving issues

How do we sketch y without storing $u_1, \ldots, u_n$? Recall analysis crucially relied on independence.

Resolving issues

How do we sketch y without storing $u_1, \ldots, u_n$? Recall analysis crucially relied on independence.

- Use k-wise independence for sufficiently large k and redo analysis
- Use hammer of pseudorandom generators

Algorithm again

- x is vector being updated. Keep track of $\|x\|_2$
- Use Count Sketch to sketch y where $y_i = x_i/\sqrt{u_i}$ with u_i drawn independently from $[0, 1]$. Use sketch to obtain estimates $\tilde{y}_i$ for heavy hitters in y
- Output i if $\tilde{y}_i^2$ is the unique heavy hitter that is above threshold t where $t = c \log n \|x\|_2^2$. If no such i then declare FAIL.

Repeat above in parallel $O(\log^2 n)$ times to guarantee high probability of obtaining a good sample.

Algorithm again

- x is vector being updated. Keep track of $\|x\|_2$
- Use Count Sketch to sketch y where $y_i = x_i/\sqrt{u_i}$ with u_i drawn independently from $[0, 1]$. Use sketch to obtain estimates $\tilde{y}_i$ for heavy hitters in y
- Output i if $\tilde{y}_i^2$ is the unique heavy hitter that is above threshold t where $t = c \log n \|x\|_2^2$. If no such i then declare FAIL.

Repeat above in parallel $O(\log^2 n)$ times to guarantee high probability of obtaining a good sample.

Space is for Count Sketch and to store generate u_i values pseudorandomly.

Algorithm again

- x is vector being updated. Keep track of $\|x\|_2$
- Use Count Sketch to sketch y where $y_i = x_i/\sqrt{u_i}$ with u_i drawn independently from $[0, 1]$. Use sketch to obtain estimates $\tilde{y}_i$ for heavy hitters in y
- Output i if $\tilde{y}_i^2$ is the unique heavy hitter that is above threshold t where $t = c \log n \|x\|_2^2$. If no such i then declare FAIL.

Repeat above in parallel $O(\log^2 n)$ times to guarantee high probability of obtaining a good sample.

Space is for Count Sketch and to store generate u_i values pseudorandomly.

Algorithm uses $\text{poly}(\log n/\epsilon))$ space and with high probability outputs $i \in [n]$ such that
$\Pr[\text{i is output}] = (1 \pm \epsilon)x_i^2/\|x\|_2^2 + 1/n^c$.

Application of ℓ_2 sampling to F_p estimation

For $p > 2$ AMS-Sampling gives algorithm to estimate F_p using $\tilde{O}(n^{1-1/p})$ space. Optimal space is $\tilde{O}(n^{1-2/p})$.

Can estimate F_p or ℓ_p for $p = 0$ and $p \in [0,2]$ in polylog space

$p \geq 2$ $\Omega(n^{1-\frac{2}{p}})$

Application of ℓ_2 sampling to F_p estimation

For $p > 2$ AMS-Sampling gives algorithm to estimate F_p using $\tilde{O}(n^{1-1/p})$ space. Optimal space is $\tilde{O}(n^{1-2/p})$.

- Use ℓ_2 sampling algorithm to generate $(i, |\tilde{x}_i|)$
- Estimate $\|x\|_2^2$
- Output $T = \|x_2\|^2 |\tilde{x}_i|^{p-2}$ as estimate

To simplify analysis/notation assume sampling is exact.

$\mathbf{E}[T] = \|x\|_2^2 \sum_i \frac{x_i^2}{\|x\|_2^2} |x_i|^{p-2} = \sum_i |x_i|^p$

Application of ℓ_2 sampling to F_p estimation

For $p > 2$ AMS-Sampling gives algorithm to estimate F_p using $\tilde{O}(n^{1-1/p})$ space. Optimal space is $\tilde{O}(n^{1-2/p})$.

- Use ℓ_2 sampling algorithm to generate $(i, |\tilde{x}_i|)$
- Estimate $\|x\|_2^2$
- Output $T = \|x_2\|^2 |\tilde{x}_i|^{p-2}$ as estimate

poly log space.

To simplify analysis/notation assume sampling is exact.

$$\mathbf{E}[T] = \|x\|_2^2 \sum_i \frac{x_i^2}{\|x\|_2^2} |x_i|^{p-2} = \sum_i |x_i|^p$$

$$Var[T] \leq \|x\|_2^4 \sum_i \frac{x_i^2}{\|x\|_2^2} x_i^{2(p-2)} \leq \|x\|_2^2 \sum_i x_i^{2p-2} \leq n^{1-2/p} (\sum_i |x_i|^p)^2.$$

Now do average plus median.

Part V

ℓ_0 Sampling

ℓ_0 Sampling

Turnstile stream: x updated with positive and negative entries

At end of stream want to sample uniformly a coordinate i among all *non-zero* coordinates in x

Special case: sampling a uniform distinct element in cash register model

$x = (0, 0, 0, 0)$
$(0, -1, 1, 0)$
$(0, 0, 2, 0)$
$(1, 0, -1, -1)$
$(-1, 0, 0, 3)$

ℓ_0 Sampling

Turnstile stream: x updated with positive and negative entries

At end of stream want to sample uniformly a coordinate i among all *non-zero* coordinates in x

Special case: sampling a uniform distinct element in cash register model

Goal: illustrate a simple algorithm via two powerful hammers

Sparse Recovery

Recall sparse recovery using Count Sketch.

$\bar{x}$

Theorem

There is a linear sketch with size $O(\frac{k}{\epsilon^2} polylog(n))$ that returns z such that $\|z\|_0 \leq k$ and with high probability $\|x - z\|_2 \leq (1 + \epsilon) err_2^k(x)$.

$$err_2^k(x) = \min_{z:\|z\|_0 \leq k} \|x - z\|_2$$

Hence space is proportional to desired output. Assumption k is typically quite small compared to n, the dimension of x.

Note that if x is k-sparse vector is *exactly* reconstructed

Random Sampling plus Sparse Recovery

x is updated in turnstile streaming fashion. Let J be the non-zero indices of x eg $(-1, 0, 0, 3, 0) \Rightarrow J = \{1, 4\}$.

Suppose we knew $|J|$ is small, say $\leq s$. Then can use sparse recovering with $\tilde{O}(s)$ space to completely recover x and can then sample uniformly. ↖ $s \, \text{polylog}(n)$

Random Sampling plus Sparse Recovery

x is updated in turnstile streaming fashion. Let J be the non-zero indices of x

Suppose we knew $|J|$ is small, say $\leq s$. Then can use sparse recovering with $\tilde{O}(s)$ space to completely recover x and can then sample uniformly.

What if $|J|$ is large?

$|I_j| = \frac{n}{2^j}$

- Guess $|J|$ to within factor of $\mathbf{2}$.
- More formally, for $j = 0$ to $\log n$ let I_j be $n/2^j$ coordinates of $[n]$ sampled uniformly at random. Note $I_0 = [n]$.
- Let y^j be vector obtained by restricting x to coordinates in I_j. $y^0 = x$.

$x = (x_1, x_2, \ldots, x_8)$ ~~$I_0 = [n]$~~

$y^0 = (x_1, x_2, \ldots, x_8)$ $I_0 = [n]$

$= y^1 = (x_1, x_2, x_4, x_6)$ random order
$I_1 = \{1, 2,$

$y^2 = (x_3, x_4)$

$y^3 = (x_5)$

Random Sampling plus Sparse Recovery

Choose $s = \Omega(\log(1/\delta))$. 100

For $j = 0, 1, \ldots, \log n$

- Use s-sparse recovery on y^j.
- If y^j is not s-sparse discard. Else pick a random non-zero coordinate in y^j and output it. And stop.

Random Sampling plus Sparse Recovery

Choose $s = \Omega(\log(1/\delta))$.

For $j = 0, 1, \ldots, \log n$

- Use s-sparse recovery on y^j.
- If y^j is not s-sparse discard. Else pick a random non-zero coordinate in y^j and output it. And stop.

Uses $O(\log n)$ s-sparse recovery data structures and hence space is poly-logarithmic assuming δ is $\Omega(n^{-c})$ for some fixed constant c.

Random Sampling plus Sparse Recovery

Choose $s = \Omega(\log(1/\delta))$.

For $j = 0, 1, \ldots, \log n$

- Use s-sparse recovery on y^j.
- If y^j is not s-sparse discard. Else pick a random non-zero coordinate in y^j and output it. And stop.

Uses $O(\log n)$ s-sparse recovery data structures and hence space is poly-logarithmic assuming δ is $\Omega(n^{-c})$ for some fixed constant c.

How can we implement random coordinates of x? Cannot store them. So how can we run sparse recovery on y^j?

Random Sampling plus Sparse Recovery

Choose $s = \Omega(\log(1/\delta))$.

For $j = 0, 1, \ldots, \log n$

- Use s-sparse recovery on y^j.
- If y^j is not s-sparse discard. Else pick a random non-zero coordinate in y^j and output it. And stop.

Uses $O(\log n)$ s-sparse recovery data structures and hence space is poly-logarithmic assuming δ is $\Omega(n^{-c})$ for some fixed constant c.

How can we implement random coordinates of x? Cannot store them. So how can we run sparse recovery on y^j? Use Nisan's generator!

Analysis

Question: Will algorithm output a random non-zero coordinate?

Analysis

Question: Will algorithm output a random non-zero coordinate?

Lemma

Suppose $|J| \leq s$ then algorithm outputs a uniform non-zero coordinate of x with high probability.

$y^0 = x$ is s-sparse. Sparse recovery algorithm succeeds with high probability.

$|J|$ $\quad J = 1000 \quad K$

I_k

Analysis

Question: Will algorithm output a random non-zero coordinate?

Lemma

Suppose $|J| \leq s$ then algorithm outputs a uniform non-zero coordinate of x with high probability.

$y^0 = x$ is s-sparse. Sparse recovery algorithm succeeds with high probability.

Lemma

Assume $|J| > s$. There is an index k such that with probability $(1 - \delta)$, y^k is s-sparse and has at least one non-zero coordinate.

Analysis

Question: Will algorithm output a random non-zero coordinate?

Lemma

Suppose $|J| \leq s$ then algorithm outputs a uniform non-zero coordinate of x with high probability.

$y^0 = x$ is s-sparse. Sparse recovery algorithm succeeds with high probability.

Lemma

Assume $|J| > s$. There is an index k such that with probability $(1 - \delta)$, y^k is s-sparse and has at least one non-zero coordinate.

Expected number of coordinates of J in y^j is $|J|/2^j$. Find j such that expected number is between $s/4$ and s and use Chernoff bound.

Analysis continued

Lemma

Assume $|J| > s$. There is an index k such that with probability $(1-\delta)$, y^k is s-sparse and has at least one non-zero coordinate.

s-sparse recovery of y^k will reconstruct it exactly. y^k has random sample of coordinates of x hence has random sample of non-zero coordinates as well. Output random non-zero coordinate of y^k.

Algorithm fails only if every y^j fails sparse recovery and $|J| > 0$ but we see that y^{k+1} succeeds with probability at least $(1-\delta)$.