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 I

F_p Estimation

F_2 Estimation and JL

For F_2 estimation and JL and Euclidean LSH we used important "stability" property of the Normal distribution.

Lemma

Let $Y_1, Y_2, \ldots, Y_d$ be independent random variables with distribution $\mathcal{N}(0,1)$. $Z = \sum_i x_i Y_i$ has distribution $\|x\|_2 \mathcal{N}(0,1)$

Standard Gaussian is 2-stable.

p-stable distributions

Definition

A real-valued distribution $\mathcal{D}$ is p-stable if $Z = \sum_{i=1}^{n} x_i Y_i$ has distribution $\|x\|_p \mathcal{D}$ when the Y_i are independent and each of them is distributed as $\mathcal{D}$.

p-stable distributions

Definition

A real-valued distribution $\mathcal{D}$ is p-stable if $Z = \sum_{i=1}^{n} x_i Y_i$ has distribution $\|x\|_p \mathcal{D}$ when the Y_i are independent and each of them is distributed as $\mathcal{D}$.

Question: Do p-stable distributions exist for $p \neq 2$?

p-stable distributions

Fact: p-stable distributions exist for all $p \in (0, 2]$ and do not exist for $p > 2$.

$p = 1$ is the Cauchy distribution which is the distribution of the ratio of two independent Guassian random variables. Has a closed form density function $\frac{1}{\pi(1+x^2)}$. Mean and variance are *not* finite.

p-stable distributions

Fact: p-stable distributions exist for all $p \in (0, 2]$ and do not exist for $p > 2$.

$p = 1$ is the Cauchy distribution which is the distribution of the ratio of two independent Guassian random variables. Has a closed form density function $\frac{1}{\pi(1+x^2)}$. Mean and variance are *not* finite.

For general p no closed form formula for density but can sample from the distribution.

p-stable distributions

Fact: p-stable distributions exist for all $p \in (0, 2]$ and do not exist for $p > 2$.

$p = 1$ is the Cauchy distribution which is the distribution of the ratio of two independent Guassian random variables. Has a closed form density function $\frac{1}{\pi(1+x^2)}$. Mean and variance are *not* finite.

For general p no closed form formula for density but can sample from the distribution.

Streaming, sketching, LSH ideas for ℓ_2 generalize to ℓ_p for $p \in (0, 2]$ via p-stable distributions and additional technical work.

Sampling from p-stable distribution

For $p \in (0, 2]$ let $\mathcal{D}_p$ denote p-stable distribution. Sampling from $\mathcal{D}_p$ via Chambers-Mallows-Stuck method

- Sample θ uniformly from $[-\pi/2, \pi/2]$.
- Sample r uniformly from $[0, 1]$.
- Output

$$\frac{\sin(p\theta)}{(\cos\theta)^{1/p}} \left(\frac{\cos((1-p)\theta)}{\ln(1/r)} \right)^{(1-p)/p}.$$

p-stable distributions need not have finite mean/variance. Hence we need to work with *median* of distribution.

Definition

The median of a distribution $\mathcal{D}$ is θ if for $Y \sim \mathcal{D}$, $\Pr[Y \leq \mu] = 1/2$. If $\phi(x)$ is the probability density function of $\mathcal{D}$ then we have $\int_{-\infty}^{\mu} \phi(x)dx = 1/2$.

F_p estimation via p-stable distribution

For $p \in (0,2]$ due to [Indyk]

F_p-Estimate:

$k \leftarrow \Theta(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$

Let M be a $k \times n$ matrix where each $M_{ij} \sim \mathcal{D}_p$

$\mathbf{y} \leftarrow M\mathbf{x}$

Output $Y \leftarrow \frac{\texttt{median}(|y_1|,|y_2|,\ldots,|y_k|)}{\texttt{median}(|\mathcal{D}_p|)}$

F_p estimation via p-stable distribution

For $p \in (0, 2]$ due to [Indyk]

> **F_p-Estimate:**
>
> $k \leftarrow \Theta(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$
>
> `Let` M `be a` $k \times n$ `matrix where each` $M_{ij} \sim \mathcal{D}_p$
>
> $\mathbf{y} \leftarrow M\mathbf{x}$
>
> `Output` $Y \leftarrow \frac{\texttt{median}(|y_1|, |y_2|, \ldots, |y_k|)}{\texttt{median}(|\mathcal{D}_p|)}$

- Each y_j is distributed according to $\|x\|_p \mathcal{D}_p$
- Cannot take average of $|y_j|^p$ values since mean of distribution is not finite
- Take median of absolute values for k independent copies and normalize by median of distribution

Concentration Lemma

Lemma

Let $\epsilon > 0$ and let $\mathcal{D}$ be a distribution with density function ϕ and a unique median $\mu > 0$. Suppose ϕ is absolutely continuous on $[(1-\epsilon)\mu, (1+\epsilon)\mu]$ and let $\alpha = \min\{\phi(x) \mid x \in [(1-\epsilon)\mu, (1+\epsilon)\mu]$. Let $Y = median(Y_1, Y_2, \ldots, Y_k)$ where $Y_1, \ldots, Y_k$ are independent samples from the distribution $\mathcal{D}$. Then

$$\Pr[|Y - \mu| \geq \epsilon\mu] \leq 2e^{-\frac{2}{3}\epsilon^2\mu^2\alpha^2 k}.$$

See notes for proof idea.

Pseudorandom generator for F_p Estimation

For F_p estimation we need $M_{i,j}$ to be independent randomly distributed according to $\mathcal{D}_p$. Can use sampling from distribution even though it is not explicit.

How do we store M in small space?

Recall that for F_2 estimation and sketching we used matrix M where each row of M had 4-wise independent random variables. Needed separate proof to argue correctness.

Is there an equivalent limited independence hashing based algorithm for F_p estimation?

Pseudorandom generator for F_p Estimation

For F_p estimation we need $M_{i,j}$ to be independent randomly distributed according to $\mathcal{D}_p$. Can use sampling from distribution even though it is not explicit.

How do we store M in small space?

Recall that for F_2 estimation and sketching we used matrix M where each row of M had 4-wise independent random variables. Needed separate proof to argue correctness.

Is there an equivalent limited independence hashing based algorithm for F_p estimation? No but can use a powerful pseudorandomness tool from TCS.

Pseudorandom generator

- *P* class of decision problems decided in poly time.
- *RP* class of decision problems decided in randomized poly time with one-sided error
- *BPP* class of decision problems decided in randomized poly time with two-sided error allowed

Pseudorandom generator

- P class of decision problems decided in poly time.
- RP class of decision problems decided in randomized poly time with one-sided error
- BPP class of decision problems decided in randomized poly time with two-sided error allowed

Big Open Problem: Is $BPP = P$? Equivalently can every randomized polynomial time algorithm be derandomized with only polynomial-factor slow down?

Pseudorandom generator

- P class of decision problems decided in poly time.
- RP class of decision problems decided in randomized poly time with one-sided error
- BPP class of decision problems decided in randomized poly time with two-sided error allowed

Big Open Problem: Is $BPP = P$? Equivalently can every randomized polynomial time algorithm be derandomized with only polynomial-factor slow down?

Equivalently: Is there a pseudo-random generator that fools every poly-sized algorithm?

Nisan's pseudorandom generator

Nisan constructed explicit pseudo-random generator that fools space-bounded algorithms.

Theorem

Let $\mathcal{A}$ be an algorithm that uses space at most $S(n)$ on an input of length n. Then there is a pseudo-random generator G that fools $\mathcal{A}$ and has seed length $\ell = O(S(n) \log n)$ and which is computable in $O(\ell)$ space and $poly(\ell)$ time.

Corollary

For $S(n) = O(\log^c n)$ the generator uses space $S(n) = O(\log^{c+1} n)$ and can generate any of the desired random pseudo-random bits for algorithm in $poly(\log n)$ time.

Applying Nisan's generator as a hammer

At a high-level if a streaming algorithm uses small space (polylogarithmic in input size) assuming access to *true* random bits then one can use Nisan's generator to reduce space.

- Nisan's generator requires small random seed. Store it.
- Generate required (pseudo)random bits "on the fly". Note that Nisan's generator itself runs in small space so total space is small.

Note that algorithm still uses random bits!

Applying Nisan's generator as a hammer

At a high-level if a streaming algorithm uses small space (polylogarithmic in input size) assuming access to *true* random bits then one can use Nisan's generator to reduce space.

- Nisan's generator requires small random seed. Store it.
- Generate required (pseudo)random bits "on the fly". Note that Nisan's generator itself runs in small space so total space is small.

Note that algorithm still uses random bits!

With additional discretization tricks one can convert Indyk's F_p estimation algorithm via Nisan's generator into a true small space algorithm.
[Kane-Nelson-Woodruff] show how to use limited independence hashing for F_p estimation instead of above hammer.

Part II

Priority Sampling

Sampling for data reduction

- X set of n points in the plane $a_1, a_2, \ldots, a_n$.
- Want to answer queries of the form: given some shape C (say circles), how many points inside C?
- standard data structures or brute force linear search say

Sampling for data reduction

- X set of n points in the plane $a_1, a_2, \ldots, a_n$.
- Want to answer queries of the form: given some shape C (say circles), how many points inside C?
- standard data structures or brute force linear search say

Question: Suppose n is too large and we can only store k points for some $k < n$.

Sampling approach:

- S sample of size k (with replacement). Store only S
- Given query C, compute $|C \cap S|$. What should we report as an estimate for $|C \cap X|$?

Sampling for data reduction

- X set of n points in the plane $a_1, a_2, \ldots, a_n$.
- Want to answer queries of the form: given some shape C (say circles), how many points inside C?
- standard data structures or brute force linear search say

Question: Suppose n is too large and we can only store k points for some $k < n$.

Sampling approach:

- S sample of size k (with replacement). Store only S
- Given query C, compute $|C \cap S|$. What should we report as an estimate for $|C \cap X|$? $\frac{n}{k}|C \cap S|$ which is an unbiased estimator

Weighted case

- X set of n points in the plane $a_1, a_2, \ldots, a_n$. Each point a_i has a non-negative weight w_i
- Want to answer queries of the form: given some shape C (say circles), what is weight of point inside C?

Question: Suppose n is too large and we can only store k points for some $k < n$.

Sampling approach?

Weighted case

- X set of n points in the plane $a_1, a_2, \ldots, a_n$. Each point a_i has a non-negative weight w_i
- Want to answer queries of the form: given some shape C (say circles), what is weight of point inside C?

Question: Suppose n is too large and we can only store k points for some $k < n$.

Sampling approach?

- Easy to see that uniform sampling is not ideal
- Sample in proportion to weight? Say a_i sampled with $p_i = w_i/W$ where $W = \sum_i w_i$.
- What do we set the weight of the sampled points to? Can we control sample size? What is the variance?

Importance Sampling

- Decide sampling probabilities $p_1, p_2, \ldots, p_n$
- Choose a_i independently with probability p_i and if i is chosen set $\hat{w}_i = w_i / p_i$. If i is not chosen we implicitly set $\hat{w}_i = 0$.

Importance Sampling

- Decide sampling probabilities $p_1, p_2, \ldots, p_n$
- Choose a_i independently with probability p_i and if i is chosen set $\hat{w}_i = w_i/p_i$. If i is not chosen we implicitly set $\hat{w}_i = 0$.
- For any i, $\mathbf{E}[\hat{w}_i] = w_i$.

Importance Sampling

- Decide sampling probabilities $p_1, p_2, \ldots, p_n$
- Choose a_i independently with probability p_i and if i is chosen set $\hat{w}_i = w_i/p_i$. If i is not chosen we implicitly set $\hat{w}_i = 0$.
- For any i, $\mathbf{E}[\hat{w}_i] = w_i$. Hence for any C, $\mathbf{E}[\hat{w}(C \cap S)] = \mathbf{E}[w(C \cap S)]$.

Importance Sampling

- Decide sampling probabilities $p_1, p_2, \ldots, p_n$
- Choose a_i independently with probability p_i and if i is chosen set $\hat{w}_i = w_i/p_i$. If i is not chosen we implicitly set $\hat{w}_i = 0$.
- For any i, $\mathbf{E}[\hat{w}_i] = w_i$. Hence for any C, $\mathbf{E}[\hat{w}(C \cap S)] = \mathbf{E}[w(C \cap S)]$.

Question: How should we choose p_i's?

Importance Sampling

- Decide sampling probabilities $p_1, p_2, \ldots, p_n$
- Choose a_i independently with probability p_i and if i is chosen set $\hat{w}_i = w_i/p_i$. If i is not chosen we implicitly set $\hat{w}_i = 0$.
- For any i, $\mathbf{E}[\hat{w}_i] = w_i$. Hence for any C, $\mathbf{E}[\hat{w}(C \cap S)] = \mathbf{E}[w(C \cap S)]$.

Question: How should we choose p_i's?

- Choose to reduce variance for queries of interest (depends on queries)
- Expected number of chosen points is $\sum_i p_i$ and hence choose p_i's to roughly meet the memory bound. If we have memory of size k then can scale p_i values (sampling rate) to achieve this.

Importance Sampling in Streaming Setting

Setting:

- points $a_1, \ldots, a_n$ with weights arriving in stream
- have a memory size of k
- want to maintain a k-sample (to utilize memory as well as possible) such that we can estimate $w(C \cap X)$ accurately
- Stream length unknown! How can we adjust sampling rate?

Priority Sampling

[Duffield,Lund,Thorup]

- Queries are arbitrary subset sums so no structure there to exploit
- Focus on streaming aspect and using memory

Priority Sampling

[Duffield,Lund,Thorup]

- Queries are arbitrary subset sums so no structure there to exploit
- Focus on streaming aspect and using memory

Scheme:

1. For each $i \in [n]$ set priority $q_i = w_i/u_i$ where u_i is chosen uniformly (and independently from other items) at random from $[0, 1]$.
2. S is the set of items with the k highest priorities.
3. τ is the $(k+1)$'st highest priority. If $k \geq n$ we set $\tau = 0$.
4. If $i \in S$, set $\hat{w}_i = \max\{w_i, \tau\}$, else set $\hat{w}_i = 0$.

Priority Sampling

[Duffield,Lund,Thorup]

- Queries are arbitrary subset sums so no structure there to exploit
- Focus on streaming aspect and using memory

Scheme:

1. For each $i \in [n]$ set priority $q_i = w_i/u_i$ where u_i is chosen uniformly (and independently from other items) at random from $[0, 1]$.
2. S is the set of items with the k highest priorities.
3. τ is the $(k+1)$'st highest priority. If $k \geq n$ we set $\tau = 0$.
4. If $i \in S$, set $\hat{w}_i = \max\{w_i, \tau\}$, else set $\hat{w}_i = 0$.

Claim: Can maintain S, τ in streaming setting

Priority Sampling

Intuition: from uniform weight case

- Suppose $w_i = 1$ for all i. Then sampling k without repetition can be done via adaptation of reservoir sampling.
- A different approach: pick a uniformly random $r_i \in [0, 1]$ for each i. And pick top k in terms of r_i values (simulates random permutation) but can be done in streaming fashion. Many other distributions would work too and picking top k according to $1/r_i$ works too.
- Why $1/r_i$? What is the expected value of τ?

Priority Sampling: Properties

Lemma

$\mathbf{E}[\hat{w}_i] = w_i$.

Priority Sampling: Properties

Lemma

$\mathbf{E}[\hat{w}_i] = w_i$.

Lemma

$$Var[\hat{w}_i] = \mathbf{E}[\hat{v}_i] \textit{ where } \hat{v}_i = \begin{cases} \tau \max\{0, \tau - w_i\} & \textit{if } i \in S \\ 0 & \textit{if } i \notin S \end{cases}$$

Useful: storing τ and w_i gives $Var[\hat{w}_i]$.

Priority Sampling: Properties

Lemma

$\mathbf{E}[\hat{w}_i] = w_i$.

Lemma

$$Var[\hat{w}_i] = \mathbf{E}[\hat{v}_i] \text{ where } \hat{v}_i = \begin{cases} \tau \max\{0, \tau - w_i\} & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$$

Useful: storing τ and w_i gives $Var[\hat{w}_i]$.

Lemma

If $k \geq 2$ *for any* $i \neq j$, $\mathbf{E}[\hat{w}_i \hat{w}_j] = w_i w_j$.

Lemma

Fix any set $C \subset [n]$. $\mathbf{E}\left[\prod_{i \in C} \hat{w}_i\right] = \prod_{i \in C} w_i$ *if* $|C| \leq k$ *and is* 0 *if* $|C| > k$.

Variance of subset sum

Lemma

If $k \geq 2$ for any $i \neq j$, $\mathbf{E}[\hat{w}_i \hat{w}_j] = w_i w_j$.

Consequence:

- Fix C. Unbiased estimator of $w(C \cap X)$ is $\hat{w}(C \cap S)$.
- Can we know the variance of the estimate to know if we are doing ok?
- $Var[\hat{w}(C \cap S)] = \sum_{i \in C \cap S} Var[\hat{w}_i] = \sum_{i \in C \cap S} \mathbf{E}[\hat{v}_i]$. Hence, storing τ and $\hat{w}_i$ values suffices to estimate the variance of the estimate.

Priority Sampling: Properties

Lemma

$\mathbf{E}[\hat{w}_i] = w_i$.

Priority Sampling: Properties

Lemma

$\mathbf{E}[\hat{w}_i] = w_i$.

Fix i. Let $A(\tau')$ be the event that the k'th highest priority among items $j \neq i$ is τ'.

Note that u_i is independent of τ'. Hence $i \in S$ if $q_i = w_i/u_i \geq \tau'$ and if $i \in S$ then $\hat{w}_i = \max\{w_i, \tau'\}$, otherwise $\hat{w}_i = 0$. To evaluate $\mathbf{Pr}[i \in S \mid A(\tau')]$ we consider two cases.

Case 1: $w_i \geq \tau'$. Here we have $\mathbf{Pr}[i \in S \mid A(\tau')] = 1$ and $\hat{w}_i = w_i$.

Case 2: $w_i < \tau'$. Then $\mathbf{Pr}[i \in S \mid A(\tau')] = \frac{w_i}{\tau'}$ and $\hat{w}_i = \tau'$.

In both cases we see that $E[\hat{w}_i] = w_i$.

Variance

Lemma

$$Var[\hat{w}_i] = \mathbf{E}[\hat{v}_i] \text{ where } \hat{v}_i = \begin{cases} \tau \max\{0, \tau - w_i\} & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$$

Variance

Lemma

$$Var[\hat{w}_i] = \mathbf{E}[\hat{v}_i] \textit{ where } \hat{v}_i = \begin{cases} \tau \max\{0, \tau - w_i\} & \textit{if } i \in S \\ 0 & \textit{if } i \notin S \end{cases}$$

Fix i. We define $A(\tau')$ to be the event that τ' is the k'th highest priority among elements $j \neq i$.

Show that

$$E[\hat{v}_i \mid A(\tau')] = E[\hat{w}_i^2 \mid A(\tau')] - w_i^2.$$

Since u_i is independent of τ' we can remove conditioning

Variance

$$E[\hat{v}_i \mid A(\tau')] = E[\hat{w}_i^2 \mid A(\tau')] - w_i^2.$$

$$\begin{aligned}
\mathbf{E}[\hat{v}_i \mid A(\tau')] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}[\hat{v}_i \mid i \in S \wedge A(\tau')] \\
&= \min\{1, w_i/\tau'\} \times \tau' \max\{0, \tau' - w_i\} \\
&= \max\{0, w_i\tau' - w_i^2\}.
\end{aligned}$$

Variance

$$E[\hat{v}_i \mid A(\tau')] = E[\hat{w}_i^2 \mid A(\tau')] - w_i^2.$$

$$\begin{aligned}
\mathbf{E}[\hat{v}_i \mid A(\tau')] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}[\hat{v}_i \mid i \in S \wedge A(\tau')] \\
&= \min\{1, w_i/\tau'\} \times \tau' \max\{0, \tau' - w_i\} \\
&= \max\{0, w_i\tau' - w_i^2\}.
\end{aligned}$$

$$\begin{aligned}
\mathbf{E}\left[\hat{w}_i^2 \mid A(\tau')\right] &= \Pr[i \in S \mid A(\tau')] \times \mathbf{E}\left[\hat{w}_i^2 \mid i \in S \wedge A(\tau')\right] \\
&= \min\{1, w_i/\tau'\} \times (\max\{w_i, \tau'\})^2 \\
&= \max\{w_i^2, w_i\tau'\}.
\end{aligned}$$

Variance of subset sum

Lemma

If $k \geq 2$ *for any* $i \neq j$, $\mathbf{E}[\hat{w}_i \hat{w}_j] = w_i w_j$.

More generally

Lemma

Fix any set $C \subset [n]$. $\mathbf{E}\left[\prod_{i \in C} \hat{w}_i\right] = \prod_{i \in C} w_i$ *if* $|C| \leq k$ *and is* $\mathbf{0}$ *if* $|C| > k$.

Variance of subset sum

Lemma

If $k \geq 2$ *for any* $i \neq j$, $\mathbf{E}[\hat{w}_i \hat{w}_j] = w_i w_j$.

More generally

Lemma

Fix any set $C \subset [n]$. $\mathbf{E}\left[\prod_{i \in C} \hat{w}_i\right] = \prod_{i \in C} w_i$ *if* $|C| \leq k$ *and is* $\mathbf{0}$ *if* $|C| > k$.

Requires a proof by induction. See notes

Variance of subset sum

Lemma

If $k \geq 2$ for any $i \neq j$, $\mathbf{E}[\hat{w}_i \hat{w}_j] = w_i w_j$.

More generally

Lemma

Fix any set $C \subset [n]$. $\mathbf{E}\left[\prod_{i \in C} \hat{w}_i\right] = \prod_{i \in C} w_i$ if $|C| \leq k$ and is 0 if $|C| > k$.

Requires a proof by induction. See notes

Why is this interesting/non-obvious? In vanilla importance sampling the variables $\hat{w}_i$ are independent. However, here the variables are correlated because we choose exactly k. Nevertheless, they exhibit properties similar to independence.

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 $I = i$.

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

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?

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

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 $(1 - \delta)$.

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 $\|x\|_2$.

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

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 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 $\tilde{O}(n^{1-2/p})$.

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

To simplify analysis/notation assume sampling is exact.

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

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

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

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.

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 $\|x - z\|_2 \leq (1 + \epsilon) err_2^k(x)$.

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

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

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.

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?

- Guess $|J|$ to within factor of 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. $y^0 = x$.

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.

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.

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 $(1-\delta)$.