Data Streams

- Stream: Motivation and Applications
- Data Stream: Model of Computation
- Synopsis and Synopsis based exact stream computation
- Sliding Window, Sample, Histogram and Wavelets
- Linear Sketches
- Count-Min Sketch
- Count Sketch
- AMS Sketch

Imdad ullah Khan

Data Stream Model

Data Stream

Stream Processing: Analytics on a continuous stream of data items

The goal is to draw meaningful analytics from the stream subject to

- Single Pass: process each item exactly once (common requirement)
- Limited Memory: poly-logarithmic space (in length of stream or domain)
- Constant per item processing: near real time
- Arbitrary arrival order: No assumption on distribution or order of items

Outline

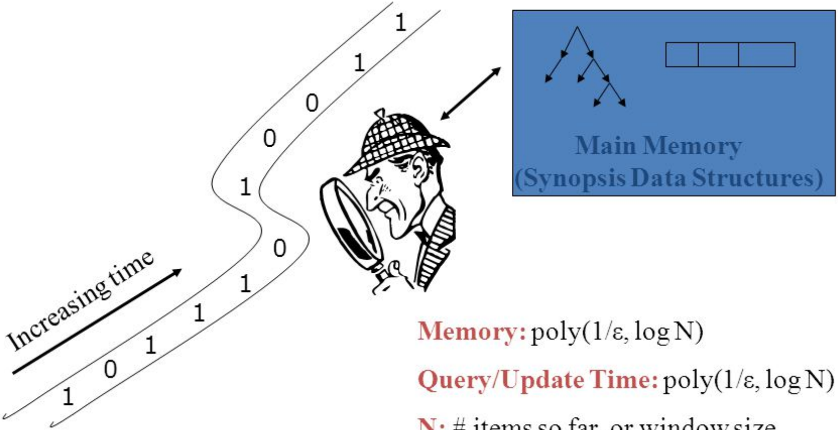
Characteristics of data streams [1]

- Huge volumes of continuous data, possibly infinite
- Fast changing and requires fast, real-time response
- Data stream captures nicely our data processing needs of today
- Random access is expensive
- Single scan algorithm (can only have one look)
- Store only the summary of the data seen thus far
- Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing

[1] Based on Han & Kamber, Data Mining Concepts & Techniques, 2nd Ed.

Stream Model of Computation

Motwani, PODS (2002)



Data Stream

Memory: poly($1/\varepsilon$, $\log N$)

Query/Update Time: poly($1/\varepsilon$, $\log N$)

N: # items so far, or window size

ε: error parameter

Data Stream

Stream data is fundamentally different than traditional datasets[2]

Traditional Data (DBMS)	Data Stream
Persistent storage	Transient stream(s)
One-time query	Continuous query
Random access	Sequential access
Unbounded disk storage	Bounded main memory
Only current state matters	Arrival-order is critical
No real time services	Real-time requirements
Low update rate	Possibly multi-GB arrival rate (dynamic & fast)
Mixed granularity	Data at fine granularity

[2] R. Motwani, PODS (2002)

Data Stream Processing Model

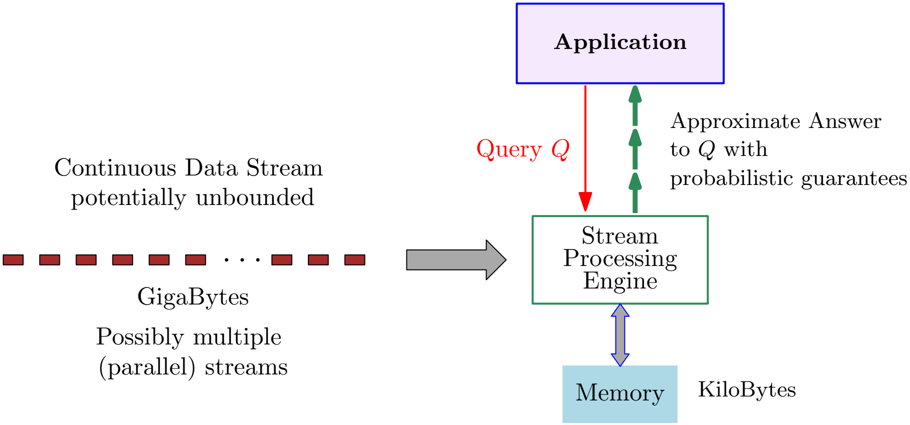
- Since streams are long (potentially unbounded) exact algorithms with limited memory are possible only for a few simple queries
- $\therefore$ we design approximate algorithms (they often suffice)

(ϵ, δ)-approximate algorithm

- $\mathcal{A}$: an algorithm to compute $f(\mathcal{S})$ ▷ (a function of stream)
- $\mathcal{A}(\mathcal{S})$: output of $\mathcal{A}$ on $\mathcal{S}$
- For $\epsilon > 0$, $0 \leq \delta \leq 1$, $\mathcal{A}$ is an (ϵ, δ)-approximation algorithm if

$$Pr\Big[\,|\mathcal{A}(\mathcal{S}) - f(\mathcal{S})| > \epsilon f(\mathcal{S})\,\Big] \leq \delta$$

Data Stream


Application
Query Q
Approximate Answer to Q with probabilistic guarantees
Continuous Data Stream potentially unbounded
Stream Processing Engine
GigaBytes
Possibly multiple (parallel) streams
Memory
KiloBytes

Data Stream: Applications

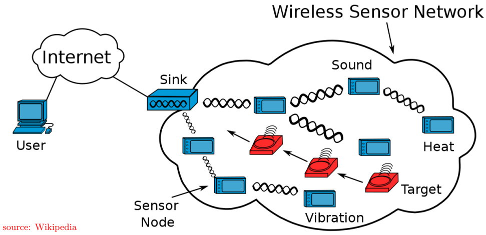
Application Domains

Stream data comes in many domains and has various applications[3]

- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Engineering & industrial processes: power supply & manufacturing
- Sensor, monitoring & surveillance: video streams, RFIDs
- Security monitoring
- Web logs and Web page click streams
- Massive data sets (even saved but random access is too expensive)

[3] Based on Han & Kamber, Data Mining Concepts & Techniques, 2nd Ed.

Applications: Sensor Networks

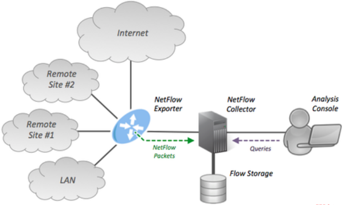


- Sensor nodes collect unlimited amount of data
- have very limited computation power and memory
- Limited battery power constrain communication of all collected data
- 1 bit transmission consumes power $\sim$ to executing 800 instructions[4]
- Streaming algorithm deployed onto nodes are ideally suited for drawing analytics from sensed data

[4] Madden et.al. (2002)

Application: Network Monitoring & Management

Network Monitoring and Management



source: Wikipedia

NetFlow: A Cisco tool for network administrators (performance metrics, security analysis, detection and forensics). For each Flow it reports (logs)

- Network Interface
- Source/Destination IP Addresses
- IP Protocol
- Source/Destination port
- TCP Flags
- Total packets/bytes in flow

- AT&T Processes over 567 billion flow records per day[5] ▷ ∼ 15 PBytes
- Detects and characterizes approximately 500 anomalies per day

[5] Fred Stinger (AT&T) FloCon (2017) Netflow Collection and Analysis ..

Application: Network Monitoring & Management

Network Monitoring and Management

Application Area

- Traffic Engineering
- Traffic Monitoring
- Volume estimation & analysis
- Load Balancing
- Efficient Resource Utilization
- (D)DOS Attack Detection
- SLA Voilation

Queries

- How many bytes sent b/w IP-1 and IP-2?
- How many IP addresses are active?
- Top 100 IP's by traffic volume
- Average duration of IP session?
- Meidan number of bytes in each IP session
- Find sessions that transmitted $> 1k$ bytes
- Find sessions with duration $>$ twice average
- List all IP's with a sudden spike in traffic
- List all IP involved in more than 1k sessions

Application: Click Stream Analysis

Web Click Stream Analysis: tracking and analysis of websites visits

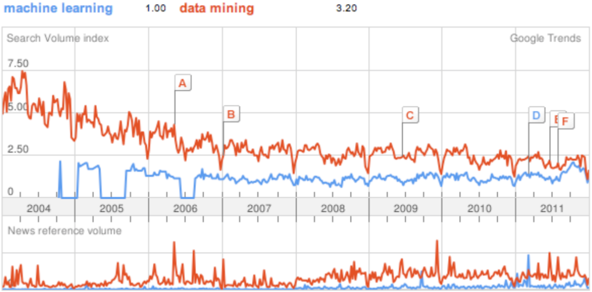
Figure credit: Alex Smola @Yahoo research & ANU

- Stream of user clicks on websites (tracked via cookies)
- Find hot links, frequent IP's, click probability
- Enhanced customer experience & conversion rates
- Digital marketing – Up-selling and cross-selling

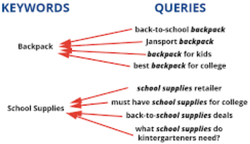
Application: Query Stream Analysis

Search Queries Stream:

Figure credit: Alex Smola @Yahoo research & ANU



- Discover trends and patterns
- Relevant keywords for website
- Estimate competition scores or difficulty
- Estimate keywords CPC (cost per click)



Application: AMI

Energy consumption Analysis:

- Electricity consumption data from AMI (Automatic Metering Interface)
- Find average hourly load, load surges, anamoly
- Short term load forecast (total or for individual consumer)
- Identify faults, drops, failures

Application: Time Series

Financial Time Series:

- Time stamped real time (multiple) stock data
- Need near real time prediction
- Algorithmic Trading

Application: Query Execution Plan

Query Execution Plan can be optimized using a synopsis of the database

Suppose we have data of $n = 1M$ people in a database and the query

SELECT * from Table WHERE $25 \leq age \leq 35$ and $54 \leq weight \leq 60$

Runtime of brute force execution is $2n$ comparisons

Suppose we have the following synopsis of distribution of an attribute

Age	Freq
$0-10$	7%
$11-20$	8%
$21-30$	10%
$31-40$	12%
$41-50$	13%
$51-60$	25%
$61-70$	20%
71+	5%

First filter on Age, then on weight

Runtime: $1.22n$

Weight	Freq.
$0-20$	20%
$21-40$	25%
$41-60$	10%
$61-80$	15%
81+	30%

First filter on Weight, then on age

Runtime: $1.1n$

Synopsis

Stream Model of Computation

Stream $\mathcal{S} := a_1, a_2, a_3, \ldots, a_m$ $\triangleright$ m may be unknown

Each $a_i \in [n]$

Goal: Compute a function of the stream $\mathcal{S}$ (e.g. mean, median, number of distinct elements, frequency moments..)

Subject to

- Single pass, read each element of $\mathcal{S}$ only once sequentially
- Per item processing time $O(1)$
- Use memory polynomial in $O(1/\epsilon, 1/\delta, \log n)$
- Return (ϵ, δ)-randomized approximate solution

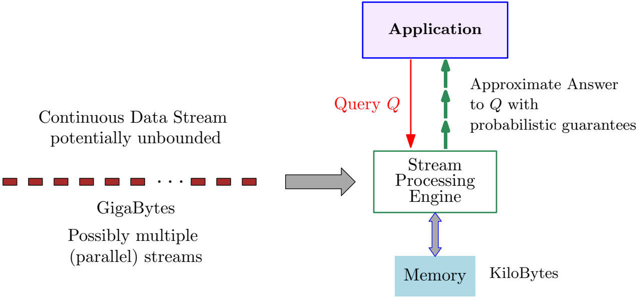
Data Stream: Synopsis

Fundamental Methodology: Keep a synopsis of the stream and answer query based on it. Update synopsis after examining each item in $O(1)$

Synopsis: Succinct summary of the stream (so far) (poly-log bits)

Families of Synopsis

- Sliding Window
- Random Sample
- Histogram
- Wavelets
- Sketch



Synopsis Based Exact Stream Computation

- Length of $\mathcal{S}$ (m): Computed by storing a running counter
- Sum of $\mathcal{S}$: Computed by storing a running sum
- Mean of $\mathcal{S}$: Computed from sum and length of $\mathcal{S}$
- Variance of $\mathcal{S}$: Computed from sum, sum of square, and length of $\mathcal{S}$

$$Var(X) \;=\; E(X^2) - (E(X))^2$$

Synopsis Based Exact Stream Computation

Missing Element

- $n - 1$ unique integers are streamed in from $[n]$
- Find the missing integer?
- Trivial to find it if we use n bits
- A better solution is to save sum S of the stream ▷ $O(\log n)$ bits
- The missing integer is $n(n+1)/2 - S$
- Can do it in exactly $\log n$ bits by storing the parity sum of each bits
- The final parity sum is the missing integer

Synopsis Based Exact Stream Computation

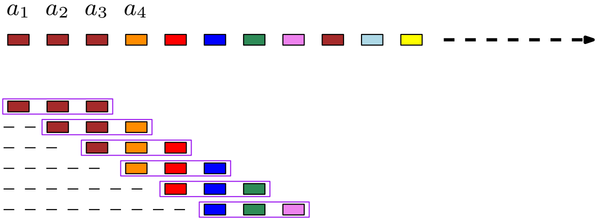
Two Missing Elements

- $n - 2$ unique integers are streamed in from $[n]$
- Find the missing integers?
- Trivial to find it if we use n bits
- Save sum of 1st and 2nd powers of stream elements $\triangleright$ $O(\log n)$ bits
- The missing integers are solution to 2 unknowns and two equations
- Readily generalizes to k missing elements

Data Stream: Sliding Window

Synopsis: Sliding Window

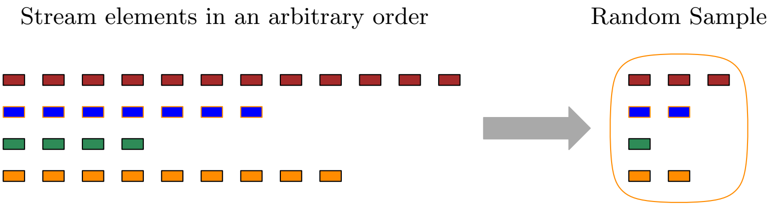
- Keep the last w elements as synopsis (w is length of window)
- On input a_i ($i \geq w$), a_{i-w} expires and a_i added to window
- Can be used for queries like mean, sum, variance, count of pre-specified element(s) (e.g. non-zero, even)
- Extended to compute approximate median, and k-median



Data Stream: Random Sample

Synopsis: Random Sample

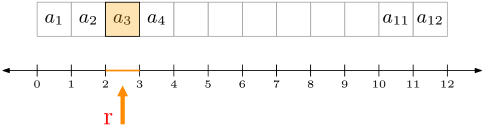
- Keep a "representative" subset of the stream
- Approximately compute query answer on sample (with appropriate scaling etc.)



Data Stream: Random Sample

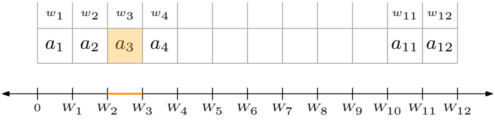
Sample a random element from array A of length n ▷ $A[i]$ with prob $1/n$

- Generate a random number $r \in [0, n]$ ▷ $r \leftarrow \text{RAND}() \times n$
- Return $A[\lceil r \rceil]$



Sample random element (by weight) from array A ▷ $A[i]$ with prob. w_i/W

- Generate a random number $r \in [0, \sum_{j=1}^{n} w_i]$ ▷ $r \leftarrow \text{RAND}() \times W_n$
- Return $A[i]$ if $W_{i-1} \leq r < W_i$



$$W_i = \sum_{j=1}^{i} w_j$$

Data Stream: Random Sample

Sample a random element from the stream S ▷ a_i with prob. $1/m$

- If m is known, use algorithm for sampling from array. For unknown m

Algorithm : Reservoir Sampling ($\mathcal{S}$)

$R \leftarrow a_1$ ▷ R (reservoir) maintains the sample

for $i \geq 2$ **do**

Pick a_i with probability $1/i$

Replace with current element in R

Prob. that a_i is in the sample R_m (m: stream length or query time)

$$= \underbrace{\text{Pr that } a_i \text{ was selected at time } i}_{\frac{1}{i}} \times \underbrace{\text{Pr that } a_i \text{ survived in } R \text{ until time } m}_{\prod_{j=i+1}^{m}\left(1-\frac{1}{j}\right)}$$

$$= \frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \frac{i+2}{i+3} \times \ldots \times \frac{m-2}{m-1} \times \frac{m-1}{m} = \frac{1}{m}$$

Data Stream: Random Sample

Sample k random elements from the stream S ▷ a_i with prob. k/m

Algorithm : Reservoir Sampling $(\mathcal{S}, k)$

$R \leftarrow a_1, a_2, \ldots, a_k$ ▷ R (reservoir) maintains the sample

for $i \geq k+1$ **do**

Pick a_i with probability k/i

If a_i is picked, replace with it a randomly chosen element in R

Prob. that a_i is in the sample R_m (m: stream length or query time)

$$= \underbrace{\text{Pr that } a_i \text{ was selected at time } i}_{\frac{k}{i}} \times \underbrace{\text{Pr that } a_i \text{ survived in } R \text{ untill time } m}_{\prod_{j=i+1}^{m}\left(1-\left(\frac{k}{j}\times\frac{1}{k}\right)\right)}$$

$$= \frac{k}{\cancel{i}} \times \frac{\cancel{i}}{\cancel{i+1}} \times \frac{\cancel{i+1}}{\cancel{i+2}} \times \frac{\cancel{i+2}}{\cancel{i+3}} \times \ldots \times \frac{\cancel{m-2}}{\cancel{m-1}} \times \frac{\cancel{m-1}}{m} = \frac{k}{m}$$

Data Stream: Histogram and Wavelets

Synopsis: Histogram

- The synopsis is some summary statistics (e.g. frequency, mean) of groups (subsets, buckets) in streams values
  - Equi-width histogram
  - Equidepth histogram
  - V-optimal histogram
  - Multi-dimensional histogram

Synopsis: Wavelets

- Essentially histograms of features (coefficients) in the frequency domain representation of the stream

Linear Sketch for Frequency

Data Stream: Linear Sketch

- Sample is a general purpose synopsis
- Process sample only – no advantage from observing the whole stream
- Sketches are specific to a particular purpose (query)
- Sketches (also histograms and wavelets) take advantage from the fact the processor see the whole stream (though can't remember all)

Data Stream: Linear Sketch

A linear sketch interprets the stream as defining the frequency vector

IP	Frequency
160.39.142.2	3
18.9.22.69	2
80.97.56.20	2

figure credit: Andoni@Columbia

Often we are interested in functions of the frequency vector from a stream

$\mathcal{S}: \; a_1, a_2, a_3, a_4, \ldots, a_m$

$a_i \in [n]$

F :

1	2	3		n
f_1	f_2	f_3	$\ldots \quad \ldots$	f_n

$f_j = |\{a_i \in \mathcal{S} : a_i = j\}|$ (frequency of j in $\mathcal{S}$)

$\mathcal{S}: \; 2,5,6,7,8,2,1,2,7,5,5,4,2,8,8,9,5,6,4,4,2,5,5$

F :

1	2	3	4	5	6	7	8	9
1	5	0	3	6	2	2	3	1

Stream: Frequency Moments

$$\mathcal{S} = < a_1, a_2, a_3, \ldots, a_m > \qquad a_i \in [n]$$

f_i : frequency of i in $\mathcal{S}$ $\qquad \mathbf{F} = \{f_1, f_2, \ldots, f_n\}$

$$F_0 := \sum_{i=1}^{n} f_i^0$$ ▷ number of distinct elements

$$F_1 := \sum_{i=1}^{n} f_i$$ ▷ length of stream, m

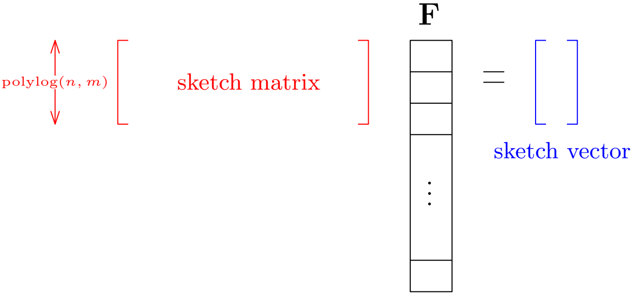
$$F_2 := \sum_{i=1}^{n} f_i^2$$ ▷ second frequency moment

Data Stream: Linear Sketch

Synopsis: Linear Sketches

Linear sketch is a synopsis that can be computed as a linear transform of **F**

- Best suited for data streams, highly parallelizable
- Very good for our problems of computing norms of **F**
- Can be readily extended to variations of the basic stream model



Data Stream Model: Time Series Model

Time Series Model

Every stream item gives the current frequency of an element ($\mathbf{F}[a_i]$)

Stream items are $a_i = \langle j, c_i \rangle$ and it means $\mathbf{F}[j] \leftarrow c_i$

For stream $\mathcal{S} : \langle 7, 3 \rangle, \langle 3, 3 \rangle, \langle 2, 9 \rangle, \langle 7, 2 \rangle, \langle 9, 1 \rangle, \langle 3, 1 \rangle$

The final frequency vector will be

$\mathbf{F} =$

1	2	3	4	5	6	7	8	9
0	9	1	0	0	0	2	0	1

- Used to measure link-bandwidth or energy consumption over time
- Very useful if there are multiple streams (e.g. stock prices for different companies

Cash-Register Model aka Arrivals-Only Stream

Every stream item is an increment to a frequency.

Stream items are $a_i = \langle j, c_i \rangle$ and it means $\mathbf{F}[j] \leftarrow \mathbf{F}[j] + c_i \quad c_i \geq 1$

For stream $\mathcal{S}$: $\langle 7,3\rangle, \langle 3,3\rangle, \langle 2,9\rangle, \langle 7,2\rangle, \langle 9,1\rangle, \langle 3,1\rangle$

The final frequency vector will be

$\mathbf{F}$ =

1	2	3	4	5	6	7	8	9
0	9	4	0	0	0	5	0	1

Can be used e.g. for packet counts in every flow

Data Stream Model: Turnstile Model

Turnstile Model aka Arrivals and Departures Stream

Every stream item is an update to a frequency

Stream items are $a_i = \langle j, c_i \rangle$ and it means $\mathbf{F}[j] \leftarrow \mathbf{F}[j] + c_i$ ~~$c_i \geq 1$~~

For stream $\mathcal{S} : \langle 7, 3\rangle, \langle 3, 3\rangle, \langle 2, 9\rangle, \langle 7, -2\rangle, \langle 9, 1\rangle, \langle 3, -1\rangle$

The final frequency vector will be

$\mathbf{F} =$

1	2	3	4	5	6	7	8	9
0	9	2	0	0	0	1	0	1

Generally, model has restriction of $\mathbf{F}[\cdot] \geq 0$

Universal hash functions

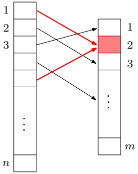
Hash functions/table is an efficient way to implement the Dictionary ADT

Hash functions map keys $A \subset U$ to m buckets labeled $\{0, 1, 2, \ldots, m-1\}$

A is not known in advance and $|A| = n$

Desired properties from hashing

- Fewer collisions
- Small range (m)
- Small space complexity to store hash function
- Easy to evaluate hash value for any key



A family of hash functions $\mathcal{H}$ is 2-universal if

$$\text{for any distinct keys } x, y \in U, \qquad \Pr_{h \in_R \mathcal{H}} \left[h(x) = h(y) \right] \leq \frac{1}{m}$$

Source of randomness is picking h (at random) from the family

Universal hash functions

A family of hash functions $\mathcal{H}$ is 2-universal if

$$\text{for any distinct keys } x, y \in U , \qquad \underset{h \in_R \mathcal{H}}{Pr}\left[h(x) = h(y)\right] \le \frac{1}{m}$$

Linear Congruential Generators for $U = \mathbb{Z}$

- Pick a prime number $p > m$
- For any two integers a and b $(1 \le a \le p-1)$, $(0 \le b \le p-1)$
- A hash function $h_{a,b} : U \mapsto [m]$ is defined as

$$h_{a,b}(x) = (ax + b) \pmod{p} \pmod{m}$$

$\mathcal{H} := \{h_{a,b} \ : \ 1 \le a \le p-1 \ , \ 0 \le b \le p-1\}$ is 2-universal

Picking a random $h \in \mathcal{H}$ amounts to picking random a and b

Count-Min Sketch

Count-Min Sketch

- Count-Min sketch (Cormode & Muthukrishnan 2005) for frequency estimates
- Cannot store frequency of every elements
- Store total frequency of random groups (elements in hash buckets)

Algorithm : Count-Min Sketch (k, ϵ, δ)

$\text{COUNT} \leftarrow \text{ZEROS}(k)$ ▷ sketch consists of k integers

Pick a random $h : [n] \mapsto [k]$ from a 2-universal family $\mathcal{H}$

On input a_i

  $\text{COUNT}[h(a_i)] \leftarrow \text{COUNT}[h(a_i)] + 1$ ▷ increment count at index $h(a_i)$

On query j ▷ query: $\mathbf{F}[j] =?$

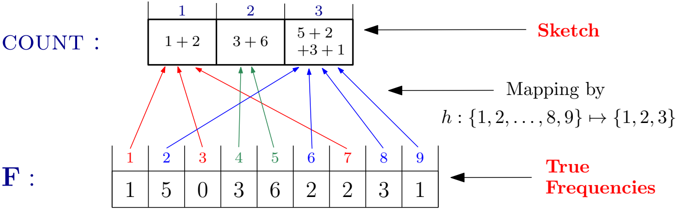
  return $\text{COUNT}[h(j)]$

Count-Min Sketch

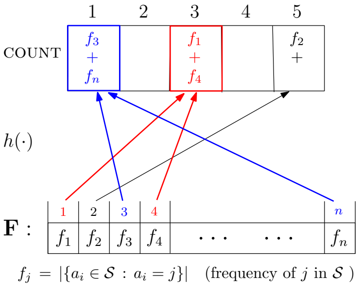
Algorithm : Count-Min Sketch (k, ϵ, δ)

COUNT $\leftarrow$ ZEROS(k) ▷ sketch consists of k integers
Pick a random $h : [n] \mapsto [k]$ from a 2-universal family $\mathcal{H}$
On input a_i
  COUNT$[h(a_i)] \leftarrow$ COUNT$[h(a_i)] + 1$ ▷ increment count at index $h(a_i)$
On query j ▷ query: $\mathbf{F}[j] =?$
  return COUNT$[h(j)]$

$$\mathcal{S} : \quad 2, 5, 6, 7, 8, 2, 1, 2, 7, 5, 5, 4, 2, 8, 8, 9, 5, 6, 4, 4, 2, 5, 5$$



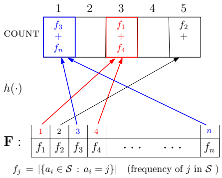
Count-Min Sketch



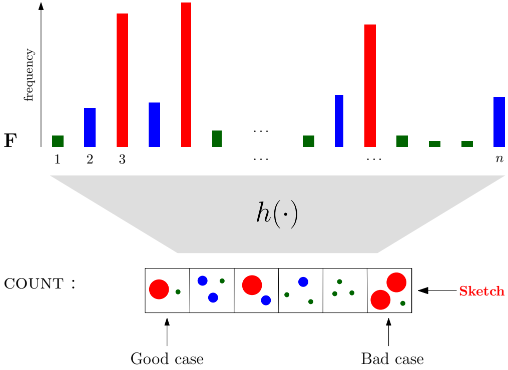
- $k = {}^{2}/_{\epsilon}$
- Large k means better estimate (smaller groups) but more space
- $\tilde{f}_j$: estimate for f_j – output of algorithm

Count-Min Sketch

- $k = 2/\epsilon$
- Large k means better estimate but more space
- $\tilde{f}_j$: estimate for f_j – output of algorithm

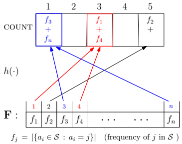


Bounds on $\tilde{f}_j$: (idea)



Count-Min Sketch

- $k = {}^2/_\epsilon$
- Large k means better estimate but more space
- $\tilde{f}_j$: estimate for f_j – output of algorithm



Bounds on $\tilde{f}_j$: (idea)

1. $\tilde{f} \geq f_j$
   - Other elements that hash to $h(j)$ contribute to $\tilde{f}_j$
2. $Pr\left[\, \tilde{f}_j \leq f_j + \epsilon \|F\|_1 \,\right] \geq \frac{1}{2}$
   - $X_j = \tilde{f}_j - f_j$ ▷ Excess in $\tilde{f}_j$ (error)
   - $X_j = \sum_{i \in [n] \setminus j} f_i \cdot 1_{h(i)=h(j)}$ ▷ $1_{condition}$ is indicator of condition

$$\mathbb{E}(X_j) = \mathbb{E}\left(\sum_{i \in [n] \setminus j} f_i \cdot 1_{h(i)=h(j)} \right) = \sum_{i \in [n] \setminus j} f_i \cdot \frac{1}{k} \leq \sum_{i \in [n] \setminus j} \|F\|_1 \cdot \frac{\epsilon}{2}$$

   - By Markov inequality we get the bound

Count-Min Sketch

Idea: Amplify the probability of the basic count-min sketch

Keep t over-estimates, $t = \log(1/\delta)$, $k = 2/\epsilon$ and return their minimum

Unlikely that all t functions hash j with very frequent elements

---

Algorithm : Count-Min Sketch (k, ϵ, δ)

$\text{COUNT} \leftarrow \text{ZEROS}(t \times k)$ ▷ sketch consists of t rows of k integers

Pick t random functions $h_1, \ldots, h_t : [n] \mapsto [k]$ from a 2-universal family

On input a_i

for $r = 1$ to t **do**

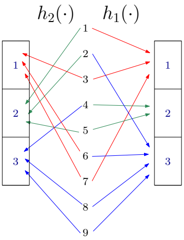
$\text{COUNT}[r][h_r(a_i)] \leftarrow \text{COUNT}[r][h_r(a_i)] + 1$

▷ increment $\text{COUNT}[r]$ at index $h_r(a_i)$

On query j ▷ query: $\mathbf{F}[j] = ?$

return $\min_{1 \leq r \leq t} \text{COUNT}[r][h_r(j)]$

Count-Min Sketch



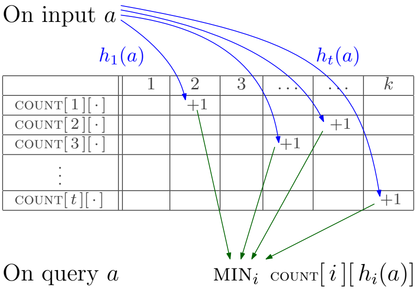
$\mathcal{S}$: 2, 5, 6, 7, 8, 2, 1, 2, 7, 5, 5, 4, 2, 8, 8, 9, 5, 6, 4, 4, 2, 5, 5

COUNT : ← **Sketch**

	1	2	3
$h_1(\cdot)$	$0+1+2$	$3+6$	$5+2$ $+3+1$
$h_2(\cdot)$	$0+2+2$	$1+5+6$	$3+3+1$

F : ← **True Frequencies**

1	2	3	4	5	6	7	8	9
1	5	0	3	6	2	2	3	1



Count-Min Sketch

1. $\tilde{f}_j \geq f_j$
   - For every r, other elements that hash to $h_r(j)$ contribute to $\tilde{f}_j$
2. $\tilde{f}_j \leq f_j + \epsilon\|F\|_1$ with probability at least $1 - \delta$
   - X_{jr} : contribution of other elements to $Count[r][h_r(j)]$
   - $\Pr\left[X_{jr} \geq \epsilon\|F\|_1\right] \leq \frac{1}{2}$ for $k = 2/\epsilon$
   - The event $\tilde{f}_j \geq f_j + \epsilon\|F\|_1$ is $\forall\ 1 \leq r \leq t \quad X_{jr} \geq \epsilon\|F\|_1$
   - $\Pr\left[\ \forall\ r\ X_{jr} \geq \epsilon\|F\|_1\ \right] \leq \left(\frac{1}{2}\right)^t$
   - $t = \log(\frac{1}{\delta}) \implies \Pr\left[\ \forall\ r\ X_{jr} \geq \epsilon\|F\|_1\ \right] \leq \left(\frac{1}{2}\right)^{\log 1/\delta} = \delta$

- Count-Min sketch is an $(\epsilon\|F\|_1, \delta)$-additive approximation algorithm
- Space required is $k \cdot t$ integers $= O(1/\epsilon \log(1/\delta) \log n)$ (plus constant)

The Count Sketch

The Count Sketch

- In Count-Min sketch error in frequency estimate accumulates (group total)
- The Count Sketch ▷ Charikar, Chen, Farach-Colton (2002)
- A frequency estimate where errors in a group cancel each other

---

Algorithm : Count Sketch (k, ϵ, δ)

Pick a random $h : [n] \mapsto [k]$ from a 2-universal family $\mathcal{H}$

Pick a random $g : [n] \mapsto \{-1, 1\}$ from a 2-universal family

$\textsc{count} \leftarrow \textsc{zeros}(k)$ ▷ sketch consists of k integers

On input a_i

$\textsc{count}[h(a_i)] \leftarrow \textsc{count}[h(a_i)] + g(a_i)$

▷ increment or decrement, depending on value of $g(a_i)$ $\textsc{count}$ at index $h(a_i)$

On query j ▷ query: $\mathbf{F}[j] =?$

return $g(j) \times \textsc{count}[h(j)]$

Count Sketch

Algorithm : Count Sketch (k, ϵ, δ)

Pick a random $h : [n] \mapsto [k]$ from a 2-universal family $\mathcal{H}$

Pick a random $g : [n] \mapsto \{-1, 1\}$ from a 2-universal family

$\textsc{count} \leftarrow \textsc{zeros}(k)$ ▷ sketch consists of k integers

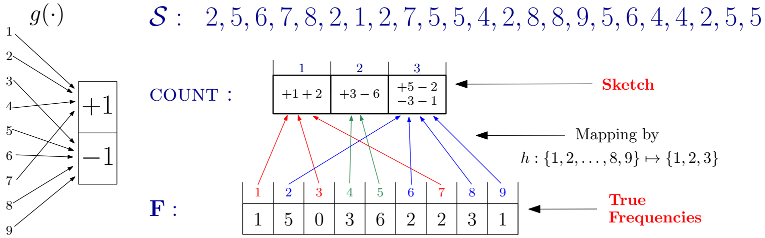
On input a_i

  $\textsc{count}[h(a_i)] \leftarrow \textsc{count}[h(a_i)] + g(a_i)$

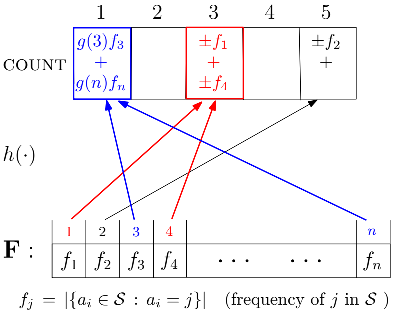
   ▷ increment or decrement, depending on value of $g(a_i)$ $\textsc{count}$ at index $h(a_i)$

On query j ▷ query: $\mathbf{F}[j] =?$

  return $g(j) \times \textsc{count}[h(j)]$



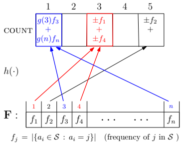
The Count Sketch



- $k = {}^{3}\!/_{\epsilon^2}$
- $\tilde{f}_j$: estimate for f_j – output of algorithm

The Count Sketch

- $k = {}^{3}/_{\epsilon^2}$
- $\tilde{f}_j$: estimate for f_j – output of algorithm



Bounds on $\tilde{f}_j$:

1. $E(\tilde{f}_j) = f_j$

$$\text{COUNT}[\,h(j)\,] \;=\; \textstyle\sum_{i\in[n]} f_i \cdot g(i) \cdot 1_{h(i)=h(j)}$$

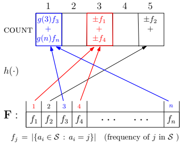
$$\tilde{f}_j \;=\; g(j) \sum_{i\in[n]} f_i \cdot g(i)\, 1_{h(i)=h(j)} = g(j)\Big(f(j)g(j) + \sum_{i\in[n]\setminus j} f_i \cdot g(i)\, 1_{h(i)=h(j)} \Big)$$

$$= f(j)(g(j))^2 + \sum_{i\in[n]\setminus j} f_i \cdot g(i)g(j) \cdot 1_{h(i)=h(j)} = f(j) + \sum_{i\in[n]\setminus j} f_i \cdot g(i)g(j)\, 1_{h(i)=h(j)}$$

$$\implies \mathbb{E}(\tilde{f}_j) = f_j \qquad \triangleright\ \mathbb{E}(1_{h(i)=h(j)}) = \tfrac{1}{k} \text{ and } \mathbb{E}(g(i)g(j)) = 0$$

The Count Sketch

- $k = {}^3/_{\epsilon^2}$
- $\tilde{f}_j$: estimate for f_j – output of algorithm



Bounds on $\tilde{f}_j$:

1. $E(\tilde{f}_j) \; = \; f_j$
2. $Var(\tilde{f}_j) \; \leq \; \frac{1}{k}\|F\|_2$ ▷ Read notes
3. $Pr\left[|\tilde{f}_j - f_j| \; \geq \; \epsilon\|F\|_2\right] \; \leq \; {}^1/_3$
   - substitute $k = {}^3/_{\epsilon^2}$ and use Chebychev inequality

Probability Amplification

Algorithm : Count Sketch (k, ϵ, δ)

$\text{COUNT} \leftarrow \text{ZEROS}(t \times k)$ ▷ sketch consists of t rows of k integers

Pick t random functions $h_1, \ldots, h_t : [n] \mapsto [k]$ from a 2-universal family

Pick t random functions $g_1, \ldots, g_t : [n] \mapsto \{-1, 1\}$ from a 2-uni. family

On input a_i

for $r = 1$ to t **do**

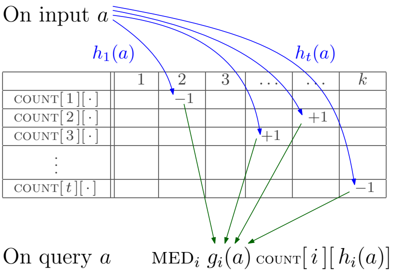
$\quad \text{COUNT}[r][h_r(a_i)] \leftarrow \text{COUNT}[r][h_r(a_i)] + g_r(a_i)$ ▷ inc/dec $\text{COUNT}[r]$ at index $h_r(a_i)$

On query j ▷ query: $\mathbf{F}[j] = ?$

$\quad$ **return** $\underset{1 \leq r \leq t}{\text{MEDIAN}} \; g_r(j) \times \text{COUNT}[r][h_r(j)]$

Count Sketch

Keep t unbiassed estimates, $t = \log(1/\delta)$, $k = 3/\epsilon^2$. Their median is a good estimate, unless at least $t/2$ estimates are very bad



1. $E(\tilde{f}_j) = f_j$
2. $|\tilde{f}_j - f_j| \leq \epsilon \|F\|_2$ with probability at least $1 - \delta$ ▷ Uses Chernoff bound

- Count sketch is an $(\epsilon\|F\|_2, \delta)$ additive approximation algorithm
- Space required is $k \cdot t$ integers $= O(1/\epsilon^2 \log(1/\delta) \log n)$ (plus constant)

AMS Sketch

Estimate F_2 : AMS Algorithm

- The AMS Sketch (Alon, Mathias, Szegedy, 1996)
- A sketch to estimate F_2 (paper has other algorithms for higher moments)

$$\mathcal{S} =< a_1, a_2, a_3, \ldots, a_m > \qquad a_i \in [n]$$

f_i: frequency of i in $\mathcal{S}$ $\quad$ $\mathbf{F} = \{f_1, f_2, \ldots, f_n\}$

$F_2 \ = \ \sum_{i=1}^{n} f_i^2$ $\triangleright$ second frequency moment

Easy to compute if we store F $\triangleright$ $O(n)$ space

Can store $f_1 + f_2 + \ldots + f_n$ $\triangleright$ $O(1)$ space

Also easy $(f_1 + f_2 + \ldots + f_n)^2$

Estimate F_2 : AMS Algorithm

F_2: $= \sum_{i=1}^{n} f_i^2$

Can store $f_1 + f_2 + \ldots + f_n$ $\triangleright$ $O(1)$ space

$(f_1 + f_2 + \ldots + f_n)^2$ can be computed by the following algorithm

Algorithm:

for each $a_i \in \mathcal{S}$

$X \leftarrow X + 1$

return X^2

$$X^2 = (f_1 + f_2 + \ldots + f_n)^2$$

Estimate F_2 : AMS Algorithm

$$F_2 = \sum_{i=1}^{n} f_i^2 = \underline{f_1^2 + f_2^2 + \ldots + f_n^2}$$ ▷ We want this

$$(f_1 + f_2 + \ldots + f_n)^2$$ ▷ Easy but overestimate

$$(f_1 + f_2 + f_3 + f_4)^2 = \underline{f_1^2 + f_2^2 + f_3^2 + f_4^2} + \underbrace{2(f_1 f_2 + f_1 f_3 + f_2 f_3 + f_1 f_4 + f_2 f_4 + f_3 f_4)}_{\text{error}}$$

$$(f_1 - f_2 + f_3 - f_4)^2 = \underline{f_1^2 + f_2^2 + f_3^2 + f_4^2} + 2(-f_1 f_2 + f_1 f_3 - f_2 f_3 - f_1 f_4 + f_2 f_4 - f_3 f_4)$$

Estimate F_2 : AMS Algorithm

Algorithm (AMS):

$g : [n] \rightarrow \{-1, +1\}$ ▷ random hash function

for each $a_i \in \mathcal{S}$

$X \leftarrow X + g(a_i)$

return X^2

$$X = f_1 g(1) + f_2 g(2) + \ldots + f_n g(n)$$

Estimate F_2 : AMS Algorithm

$$X^2 = \big(f_1 g(1) + f_2 g(2) + \ldots + f_n g(n) \big)^2$$

$$\begin{aligned}\mathbb{E}\left[X^2\right] &= \mathbb{E}\Big[\sum_i (f_i g(i))^2\Big] + \mathbb{E}\Big[\sum_{i \neq j} f_i g(i) f_j g(j)\Big] \\ &= \mathbb{E}\Big[\sum_i f_i^2\Big] + \mathbb{E}\Big[\sum_{i \neq j} f_i f_j g(i) g(j)\Big] \\ &= F_2 + \sum_{i \neq j} f_i f_j \, \mathbb{E}\big[g(i)g(j)\big] = F_2\end{aligned}$$

$$\mathbb{E}\left[X^2\right] = F_2$$

Estimate F_2 : AMS Algorithm

$X^2 = \left(f_1 g(1) + f_2 g(2) + \ldots + f_n g(n) \right)^2$ $\qquad \mathbb{E}\left[X^2\right] = F_2$

$Var(X^2) \;=\; \mathbb{E}\left[X^4\right] - (\mathbb{E}\left[X^2\right])^2$

$\mathbb{E}\left[X^4\right] \;=\; \mathbb{E}\left[\sum_i (f_i g(i))^4 + 6\sum_{i\neq j}(f_i g(i)^2 f_j g(j))^2\right] + \ldots$

other terms: $\mathbb{E}\left[g(i)g(j)g(k)g(l)\right] = \mathbb{E}\left[g(i)^2 g(j)g(k)\right] = \mathbb{E}\left[g(i)^3 g(j)\right] = 0$

▷ 4-wise independence

$\mathbb{E}\left[X^4\right] \;=\; \sum_i f_i^4 + 6\sum_{i\neq j} f_i^2 f_j^2$

$Var(X^2) \;=\; \sum_i f_i^4 + 6\sum_{i\neq j} f_i^2 f_j^2 - (\sum_i f_i^2)^2 \;=\; 4\sum_{i\neq j} f_i^2 f_j^2 \;\leq\; 2F_2^2$

Amplifying the probability of basic AMS Sketch

- Keep $k = {}^{8}/_{\epsilon^2} \times \log({}^{1}/_{\delta})$ estimates, $X_1, X_2, \ldots, X_k$
- Return $\bar{X}$: median of $\log({}^{1}/_{\delta})$ averages of groups of ${}^{8}/_{\epsilon^2}$ estimates

---

Algorithm : AMS sketch to estimate F_2 of $\mathcal{S}$ (ϵ, δ)

Pick $k = {}^{8}/_{\epsilon^2} \times \log({}^{1}/_{\delta})$ random hash functions $g_j : [n] \to \{-1, +1\}$

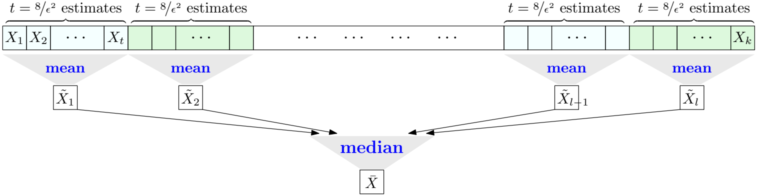
$X \leftarrow$ ZEROS(k) ▷ sketch consists of k integer

On input a_i

for $j = 1 \to k$ **do**

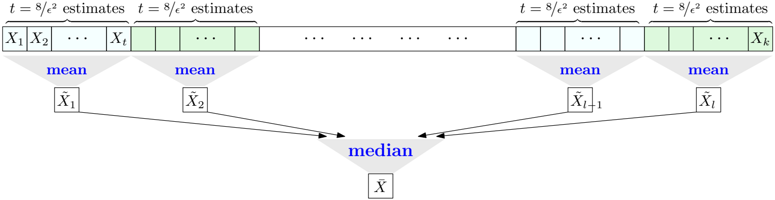
  $X[j] \leftarrow X[j] + g_j(a_i)$

  return $\bar{X}$: median of $\log({}^{1}/_{\delta})$ means of groups of ${}^{8}/_{\epsilon^2}$ estimates $(X[\cdot]^2)$



Amplifying the probability of basic AMS Sketch

- Keep $k = {}^{8}/_{\epsilon^2} \times \log({}^{1}/_{\delta})$ estimates, $X_1, X_2, \ldots, X_k$
- Return $\bar{X}$: median of $\log({}^{1}/_{\delta})$ averages of groups of ${}^{2}/_{\epsilon^2}$ estimates



- $\mathbb{E}\left[X_j^2\right] = F_2 \qquad Var\left(X_j^2\right) \leq 2F_2^2$
- $\mathbb{E}\left[\tilde{X}_j\right] = F_2 \qquad Var\left(\tilde{X}_j\right) \leq {}^{\epsilon^2}/_{4}F_2^2$
- $Pr\left[\,|\tilde{X}_j - F_2| \geq \epsilon F_2\right] \leq {}^{Var\left(\tilde{X}_j\right)}/_{\epsilon^2 F_2^2} = {}^{1}/_{4}$ ▷ Chebyshev Inequality
- $Pr\left[\,|\bar{X} - F_2| \geq \epsilon F_2\right] \leq \delta$

  The last inequality uses the Chernoff bound. For $\bar{X}$ to deviate this much from F_2 at least half of $\tilde{X}_j$ have to deviate more than that

Linear Transformation View of AMS Sketch

Algorithm : AMS sketch to estimate F_2 of $\mathcal{S}$

Pick k random hash functions $g : [n] \mapsto \{-1, +1\}$

$X \leftarrow \text{ZEROS}(k)$ ▷ sketch consists of 1 integer

On input a_i

for $j = 1 \to k$ **do**

$X[j] \leftarrow X[j] + g_j(a_i)$

$$\mathbf{g} = \begin{bmatrix} g(1) & g(2) & \dots & & g(n) \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \qquad = X$$

Linear Transformation View of AMS Sketch

Algorithm : AMS sketch to estimate F_2 of $\mathcal{S}$

Pick k random hash functions $g : [n] \mapsto \{-1, +1\}$

$X \leftarrow \textsc{zeros}(k)$ ▷ sketch consists of 1 integer

On input a_i

for $j = 1 \to k$ **do**

$\quad X[j] \leftarrow X[j] + g_j(a_i)$

$$\mathbf{g} = \begin{bmatrix} +1 & -1 & \dots & & +1 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \qquad = X$$

Linear Transformation View of AMS Sketch

Algorithm : AMS sketch to estimate F_2 of $\mathcal{S}$

Pick k random hash functions $g : [n] \mapsto \{-1, +1\}$

$X \leftarrow \text{ZEROS}(k)$ ▷ sketch consists of 1 integer

On input a_i

for $j = 1 \rightarrow k$ **do**

$X[j] \leftarrow X[j] + g_j(a_i)$

$\mathbf{G} =$

$+1$	-1	$\dots$		$+1$
-1	-1	$\dots$		-1

F

f_1
f_2
$\vdots$
$\vdots$
f_n

X

X_1
X_2

Linear Transformation View of AMS Sketch

Algorithm : AMS sketch to estimate F_2 of $\mathcal{S}$

Pick k random hash functions $g : [n] \mapsto \{-1, +1\}$

$X \leftarrow \text{ZEROS}(k)$ ▷ sketch consists of 1 integer

On input a_i

for $j = 1 \to k$ **do**

  $X[j] \leftarrow X[j] + g_j(a_i)$

$\mathbf{G} =$

$+1$	-1	$\dots$		$+1$
-1	-1	$\dots$		-1
$\vdots$		$\dots$		$\vdots$
-1	$+1$	$\dots$		-1

F

f_1
f_2
$\vdots$
$\vdots$
f_n

X

X_1
X_2
$\vdots$
X_k

Estimate F_2 : AMS Algorithm

$$\mathbf{G} = \begin{array}{|c|c|c|c|c|}\hline +1 & -1 & \dots & & +1 \\ \hline -1 & -1 & \dots & & -1 \\ \hline \vdots & & \dots & & \vdots \\ \hline -1 & +1 & \dots & & -1 \\ \hline \end{array} \qquad \mathbf{F} = \begin{array}{|c|}\hline f_1 \\ \hline f_2 \\ \hline \vdots \\ \hline \vdots \\ \hline f_n \\ \hline \end{array} \qquad \mathbf{X} = \begin{array}{|c|}\hline X_1 \\ \hline X_2 \\ \hline \vdots \\ \hline X_k \\ \hline \end{array}$$

$$\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i^2 \qquad \Pr\left[|\bar{X} - F_2| > \epsilon F_2\right] \leq \delta$$

With probability at leat $1 - \delta$

$$(1-\epsilon)\sum_{i=1}^{n} f_i^2 < \frac{1}{k}\sum_{i=1}^{k} X_i^2 < (1+\epsilon)\sum_{i=1}^{n} f_i^2$$

$$\sqrt{(1-\epsilon)}\|F\|_2 < \frac{1}{\sqrt{k}}\|X\|_2 < \sqrt{(1+\epsilon)}\|F\|_2$$

Estimate F_2 : AMS Algorithm

$$\mathbf{G} = \begin{array}{|c|c|c|c|c|}\hline +1 & -1 & \dots & & +1 \\ \hline -1 & -1 & \dots & & -1 \\ \hline \vdots & & \dots & & \vdots \\ \hline -1 & +1 & \dots & & -1 \\ \hline \end{array} \qquad \mathbf{F} = \begin{array}{|c|}\hline f_1 \\ \hline f_2 \\ \hline \vdots \\ \hline \vdots \\ \hline f_n \\ \hline \end{array} \qquad \mathbf{X} = \begin{array}{|c|}\hline X_1 \\ \hline X_2 \\ \hline \vdots \\ \hline X_k \\ \hline \end{array}$$

$$\sqrt{(1-\epsilon)}\|F\|_2 \ < \ \frac{1}{\sqrt{k}}\|X\|_2 \ < \ \sqrt{(1+\epsilon)}\|F\|_2$$

G is a random linear transformation reduces the dimension of F while preserving its ℓ_2 norm

Since G is linear it is easy to see that given $U, V \in \mathcal{R}^n$

$$\text{w.h.p} \qquad \|\frac{1}{\sqrt{k}}\mathbf{G}U\|_2 - \|\frac{1}{\sqrt{k}}\mathbf{G}V\|_2 \ \sim \ \|U - V\|_2$$

Johnson-Lindenstrauss Lemma

- Given $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\} \subset \mathcal{R}^d$
- For any $\epsilon \in (0, 1/2)$, there is a linear map $f : \mathcal{R}^d \mapsto \mathcal{R}^k$
- $k = c \log n/\epsilon^2$, such that for any $\mathbf{u}, \mathbf{v} \in V$

$$(1-\epsilon)\|\mathbf{u}-\mathbf{v}\|_2 \;\leq\; \|f(\mathbf{u})-f(\mathbf{v})\|_2 \;\leq\; (1+\epsilon)\|\mathbf{u}-\mathbf{v}\|_2$$

- This map can be obtained very easily
- Let $\mathbf{M}$ be a $k \times d$ matrix, with $M_{ij} \in \mathcal{N}(0,1)$, then

$$f(\mathbf{u}) = \frac{1}{\sqrt{k}}\mathbf{Mu}$$