Streaming Data Mining

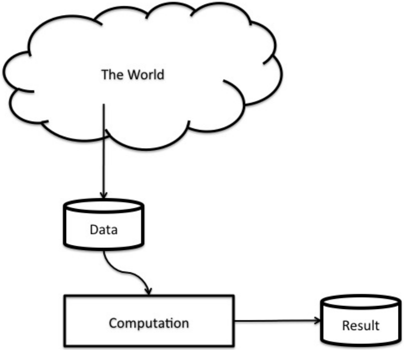
Edo Liberty [1] Jelani Nelson[2]

---

[1] Yahoo! Research, `edo.liberty@ymail.com`.

[2] Princeton University, `minilek@seas.harvard.edu`.

The need for Streaming Data Mining

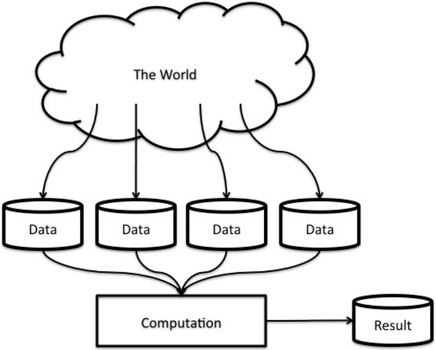


Standard Interface between data and mining algorithms

The need for Streaming Data Mining

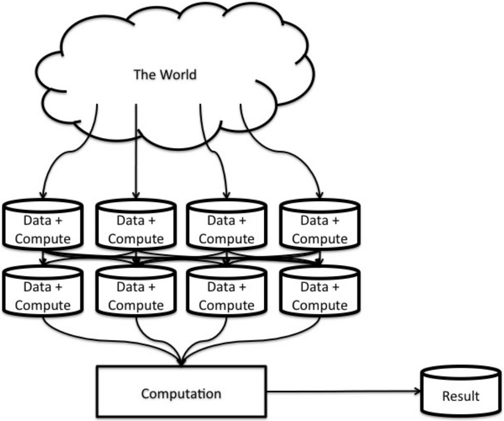
We have a lot of data...
Example: videos, images, email messages, webpages, chats, click data, search queries, shopping history, user browsing patterns, GPS trials, financial transactions, stock exchange data, electricity consumption, traffic records, Seismology, Astronomy, Physics, medical imaging, Chemistry, Computational Biology, weather measurements, maps, telephony data, SMSs, audio tracks and songs, applications, gaming scores, user ratings, questions answer forums, legal documentation, medical records, network trafic records, satellite mesurants, digital microscopy, cellular records...

The need for Streaming Data Mining



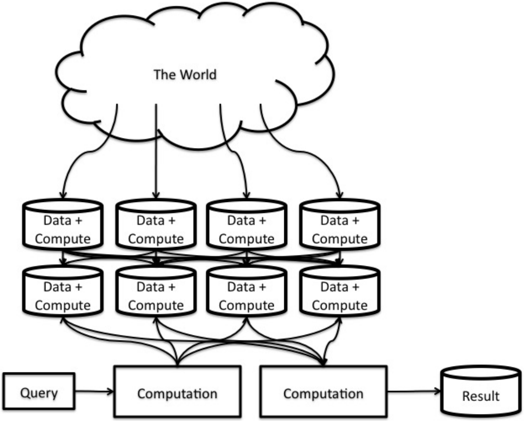
Distributed storage and file systems (HaddopFS, GFS, Cassandra, XIV)

The need for Streaming Data Mining



Distributed computation (MapReduce, Hadoop, Message passing)

The need for Streaming Data Mining



Distributed computation (Web search, Hbase/bigtable)

The need for Streaming Data Mining

"Study Projects Nearly 45-Fold Annual Data Growth by 2020" EMC Press Release.

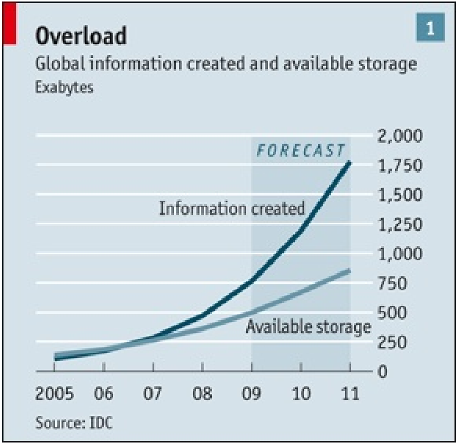
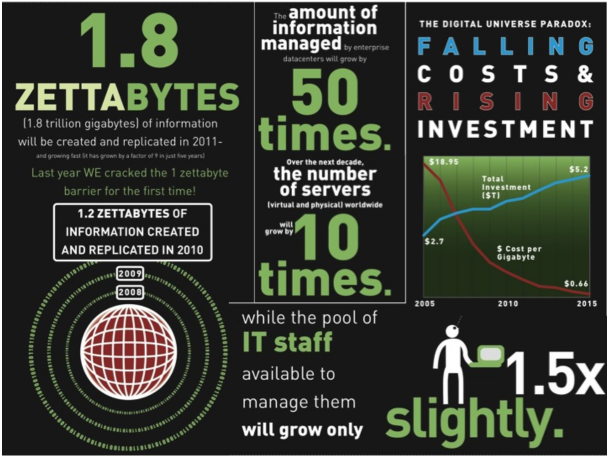


Figure: The Economist: Data, data everywhere

The need for Streaming Data Mining



1.8 ZETTABYTES

(1.8 trillion gigabytes) of information will be created and replicated in 2011-

and growing fast (it has grown by a factor of 9 in just five years)

Last year WE cracked the 1 zettabyte barrier for the first time!

1.2 ZETTABYTES OF INFORMATION CREATED AND REPLICATED IN 2010

The amount of information managed by enterprise datacenters will grow by 50 times.

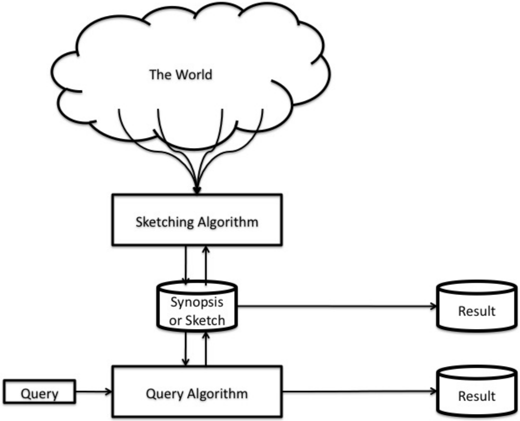
Over the next decade, the number of servers (virtual and physical) worldwide will grow by 10 times.

while the pool of IT staff available to manage them will grow only slightly. 1.5x

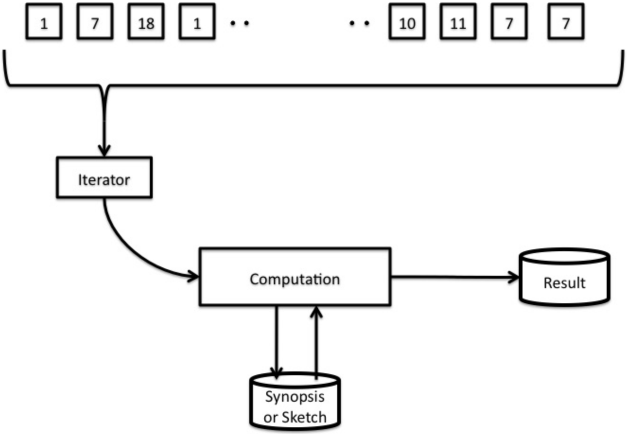
THE DIGITAL UNIVERSE PARADOX: FALLING COSTS & RISING INVESTMENT

IDC 2011 Digital Universe Study.

The need for Streaming Data Mining

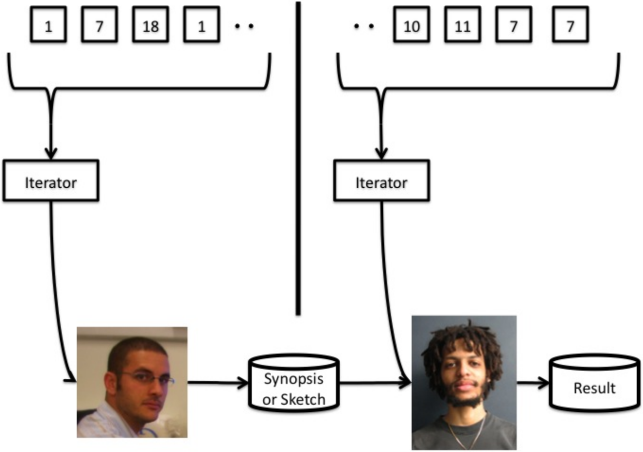


More exact model



Trivial tasks: count items, sum values, sample, find min/max.

Communication complexity



Impossible tasks: finding median, alert on new item, most frequent item.

Streaming Data Mining

When things are possible and not trivial:

Streaming Data Mining

When things are possible and not trivial:

1. Most tasks/query-types require different sketches

Streaming Data Mining

When things are possible and not trivial:

1. Most tasks/query-types require different sketches
2. Algorithms are usually randomized

Streaming Data Mining

When things are possible and not trivial:

1. Most tasks/query-types require different sketches
2. Algorithms are usually randomized
3. Results are, as a whole, approximated

Streaming Data Mining

When things are possible and not trivial:

1. Most tasks/query-types require different sketches
2. Algorithms are usually randomized
3. Results are, as a whole, approximated

But

Streaming Data Mining

When things are possible and not trivial:

1. Most tasks/query-types require different sketches
2. Algorithms are usually randomized
3. Results are, as a whole, approximated

But

1. Approximate result is expectable → significant speedup (one pass)

Streaming Data Mining

When things are possible and not trivial:

1. Most tasks/query-types require different sketches
2. Algorithms are usually randomized
3. Results are, as a whole, approximated

But

1. Approximate result is expectable → significant speedup (one pass)
2. Data cannot be stored → only option

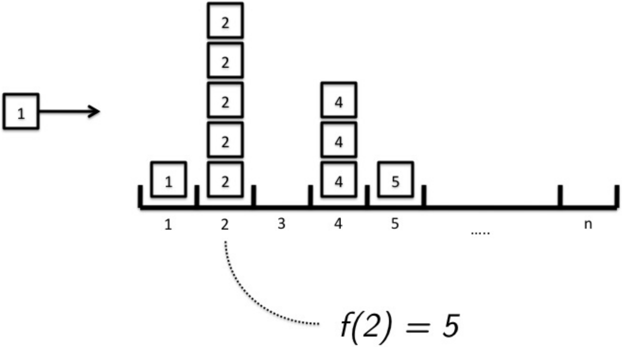
Streaming Data Mining

1. Items (words, IP-adresses, events, clicks,...):
   - Item frequencies
   - Distinct elements
   - Moment estimation
2. Vectors (text documents, images, example features,...)
   - Dimensionality reduction
   - k-means
   - Linear Regression
3. Matrices (text corpora, user preferences, social graphs,...)
   - Efficiently approximating the covariance matrix
   - Sparsification by sampling

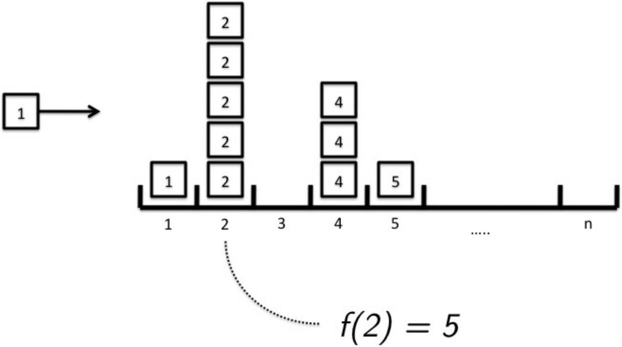
Streaming Data Mining

1. Items
    - **Item frequencies**
    - Distinct elements
    - Moment estimation
2. Vectors
    - Dimensionality reduction
    - k-means
    - Linear Regression
3. Matrices
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

Item frequencies

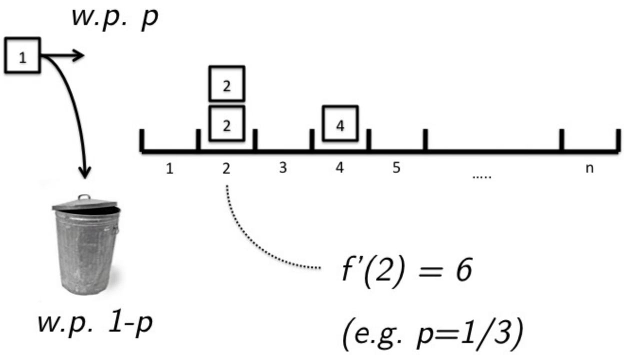


Item frequencies



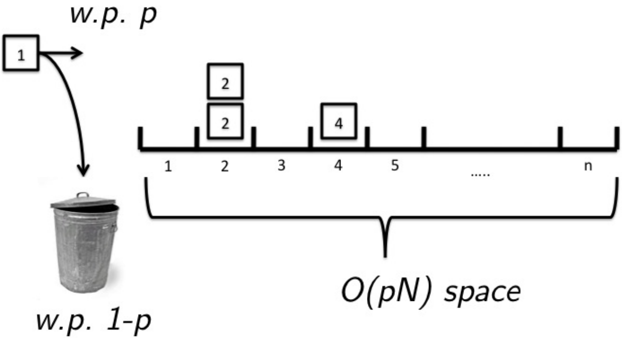
Computing $f(i)$ for all i is easy in $O(n)$ space.

Sampling



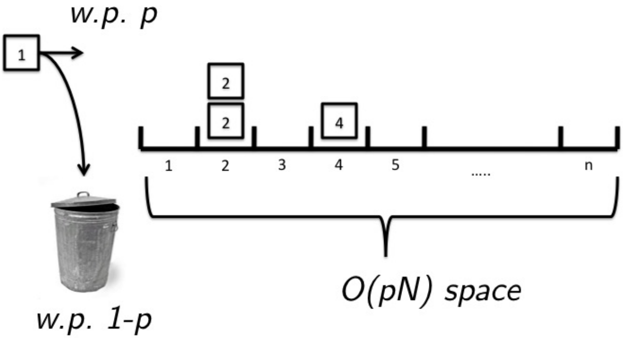
We sample with probability p and estimate $f'(i) = \frac{1}{p}A(i)$.

Sampling



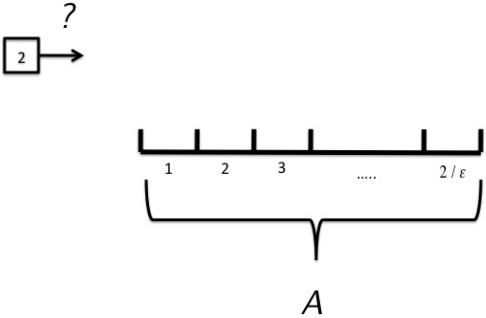
For $f'(i) = f(i) \pm \varepsilon N$ it suffices that $p \geq c \log(n/\delta)/N\varepsilon^2$.

Sampling



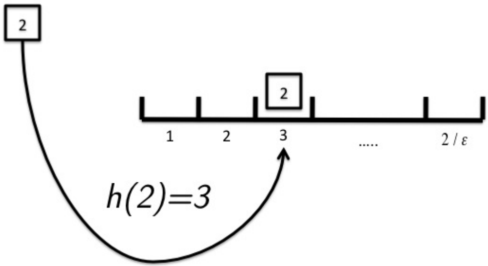
The space requirement is therefore $O(\log(n/\delta)/\varepsilon^2)$.

Count min sketches



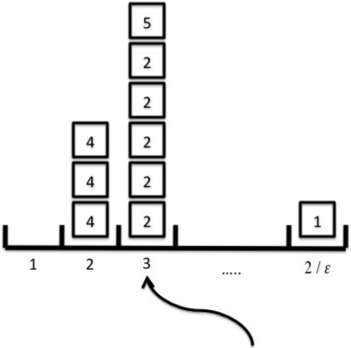
This time we keep only $2/\varepsilon$ counters in an array A

Count min sketches



Items are counted in buckets according to a hash function $h : [n] \rightarrow [2/\varepsilon]$.

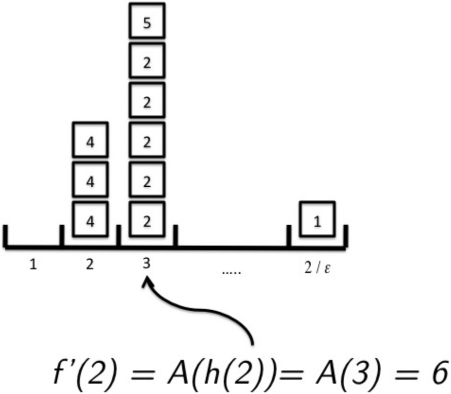
Count min sketches



$$f'(2) = A(h(2)) = A(3) = 6$$

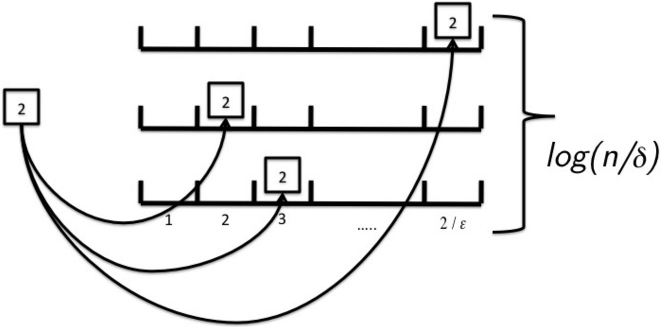
Obviously we have that $f'(i) \geq f(i)$

Count min sketches



But also $\Pr[f'(i) \leq f(i) + \varepsilon N] \geq 1/2$.

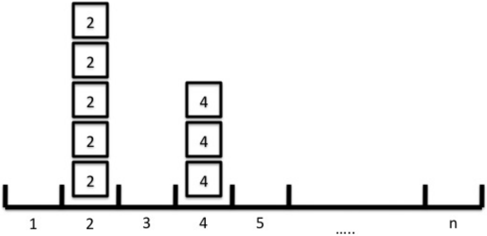
Count min sketches



$$f'(i) = min_j\ A_j(h(i))$$

This reduces the space requirement to $O(\log(n/\delta)/\varepsilon)$.

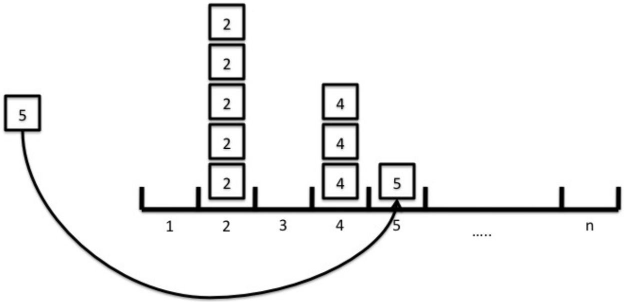
Lossy Counting



We show how to reduce the space requirement to $1/\varepsilon$.

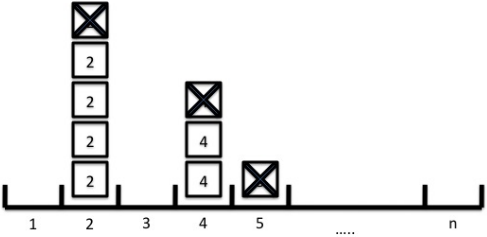
Misra, Gries. Finding repeated elements, 1982.
Demaine, Lopez-Ortiz, Munro. Frequency estimation of internet packet streams with limited space, 2002
Karp, Shenker, Papadimitriou. A simple algorithm for finding frequent elements in streams and bags, 2003
The name "Lossy Counting" was used for a different algorithm here by Manku and Motwani, 2002
Metwally, Agrawal, Abbadi, Efficient Computation of Frequent and Top-k Elements in Data Streams, 2006 (Space Saving)

Lossy Counting



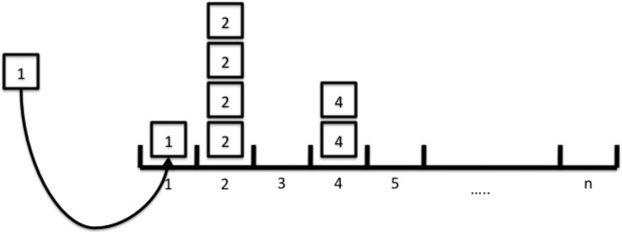
We keep at most $1/\varepsilon$ different items (in this case 2)

Lossy Counting



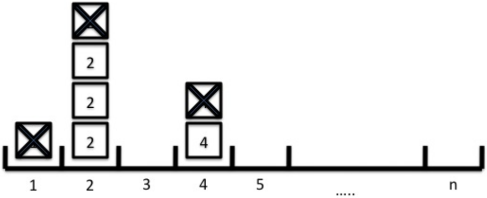
If we have more than $1/\varepsilon$ we reduce all counters

Lossy Counting



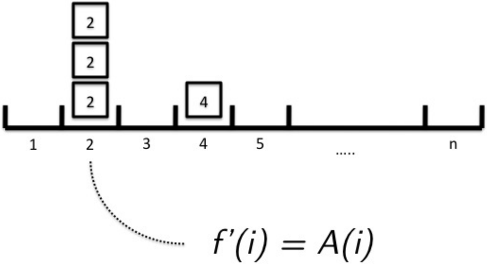
And repeat...

Lossy Counting



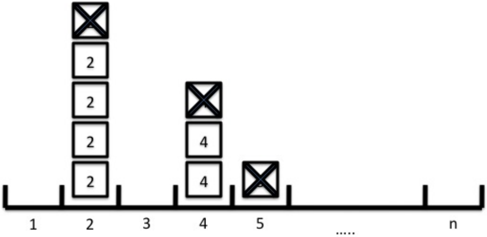
Until the stream is consumed

Lossy Counting



We have $f(i) \geq f'(i) \geq f'(i) - \varepsilon N$.

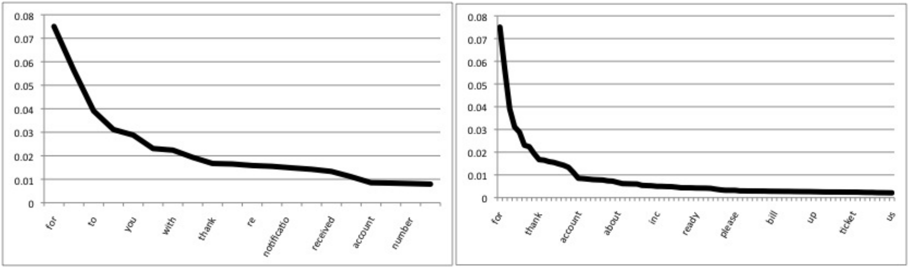
Lossy Counting



This is because we can delete $1/\varepsilon$ items at most εN times!

Error Relative to the Tail

Figure: Distribution of top 20 and top 1000 most frequent words a messaging text corpus. The heavy tail distribution is common to many data sources.

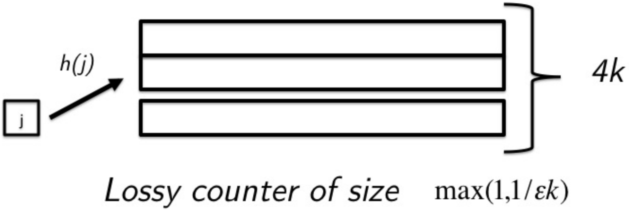


Therefore, $\varepsilon N = \varepsilon \sum_{i=1}^{n} f(i)$ might not be tight enough.

We now see how to reduce the approximation guarantee to

$$\varepsilon \sum_{i=k+1}^{n} f(i)$$

Count Max Sketches

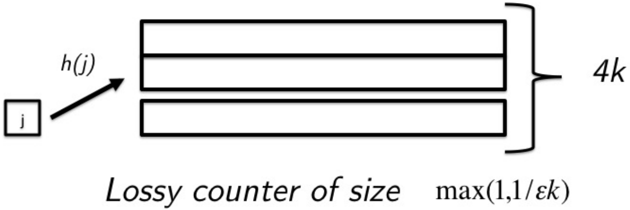


Lossy counter of size $\max(1,1/\varepsilon k)$

Distributing items with hash function $h : [n] \rightarrow [4k]$ to $4k$ lossy counters.

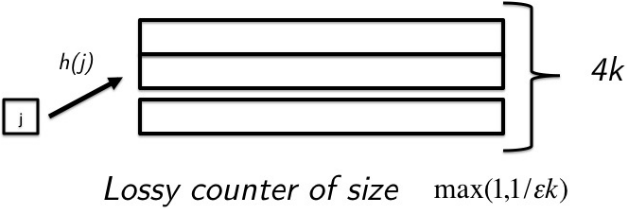
Beware: algorithm does not exist in the litrature, only described for didactic reasons.

Count Max Sketches



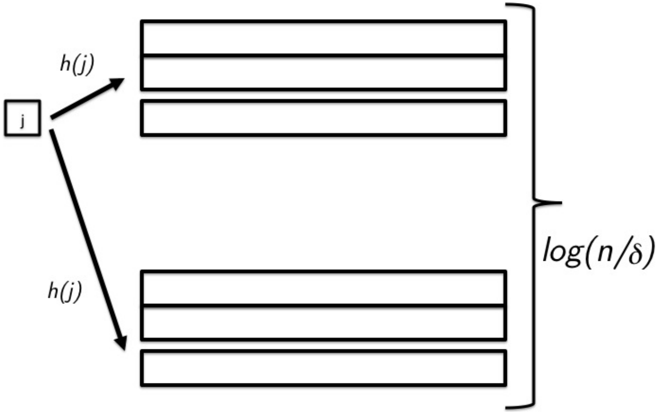
Then $n_j = \sum_{j:h(j)=h(i)} f(j) \leq \sum_{i=k+1}^{n} f(i)/k$ with probability at least $1/2$

Count Max Sketches



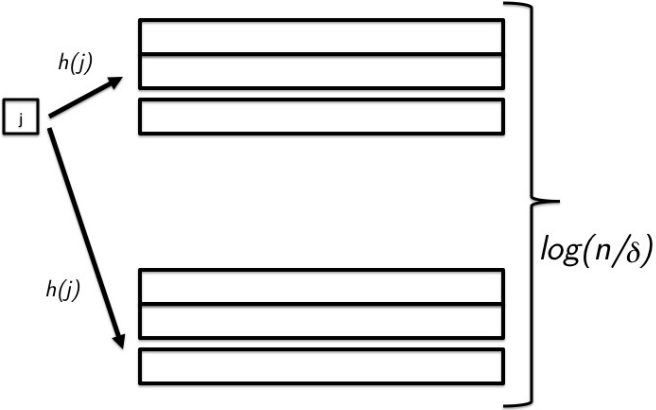
So, $f'(i) > f(i) - \varepsilon \sum_{i=k+1}^{n} f(i)$ with probability at least 1/2

Count Max Sketches



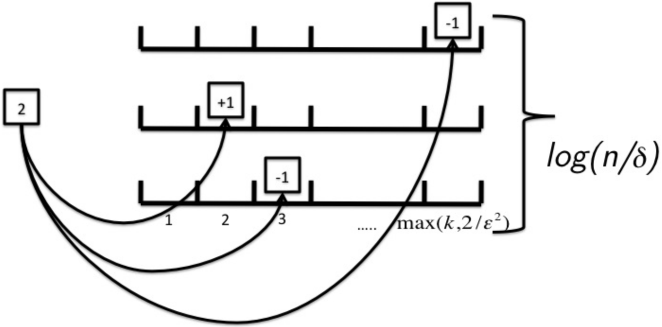
$f'(i)$ is taken as the maximum value over $\log(n/\delta)$ such structures.

Count Max Sketches



This gives a total size of $O(\log(n/\delta)/\min(\varepsilon, 1/k))$

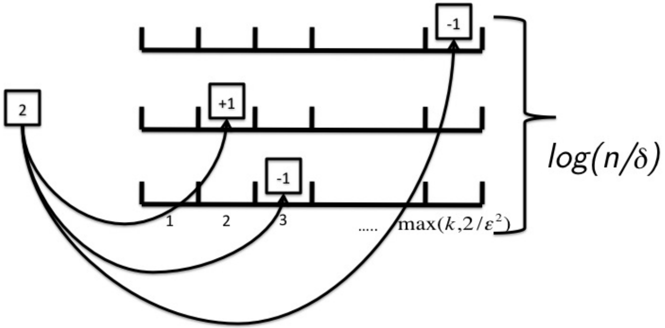
Count Sketches



Reduces the error to $\varepsilon[\sum_{i=k+1}^{n} f^2(i)]^{1/2}$

Charikar, Chen, Farach-Colton. Finding frequent items in data streams. 2002

Count Sketches



While the space increases slightly to $O(\log(n/\delta)/\min(\varepsilon^2, 1/k))$

Item frequency estimation

Table: Recap of the six algorithms presented. All quantities are given in the big-O notation.

	Space	Update	Query	Approximation	$\Pr_{fail}$
Naive	n	1	1	0	0
Sampling	$\frac{\log(n/\delta)}{\varepsilon^2}$	1	1	$\varepsilon \sum_{i=1}^{n} f(i)$	δ
Count Min Sketches	$\frac{\log(n/\delta)}{\varepsilon}$	$\log(\frac{n}{\delta})$	$\log(\frac{n}{\delta})$	$\varepsilon \sum_{i=1}^{n} f(i)$	δ
Lossy Counting	$\frac{1}{\varepsilon}$	1	1	$\varepsilon \sum_{i=1}^{n} f(i)$	0
Count Max Sketches	$\frac{\log(n/\delta)}{\min(\varepsilon,1/k)}$	$log(\frac{n}{\delta})$	$\log(\frac{n}{\delta})$	$\varepsilon \sum_{i=k+1}^{n} f(i)$	δ
Count Sketches	$\frac{\log(n/\delta)}{\min(\varepsilon^2,1/k)}$	$log(\frac{n}{\delta})$	$\log(\frac{n}{\delta})$	$\varepsilon[\sum_{i=k+1}^{n} f^2(i)]^{1/2}$	δ

See also: Berinde, Indyk, Cormode, Strauss, Space-optimal heavy hitters with strong error bounds, PODS 2009

Survey at: Gibbons, Matias, External memory algorithms, 1999

Streaming Data Mining

1. Items
    - Item frequencies
    - **Distinct elements**
    - Moment estimation
2. Vectors
    - Dimensionality reduction
    - k-means
    - Linear Regression
3. Matrices
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

Distinct elements

server

Distinct elements

server

`to:`	`cs.princeton.edu`
`from:`	`18.9.22.69`

packet

Distinct elements

server

`to:`	`cs.princeton.edu`
`from:`	`18.9.22.69`

packet

Addresses seen:

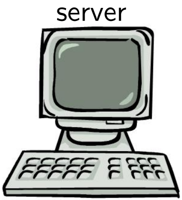
`18.9.22.69`

Distinct elements

Addresses seen:

18.9.22.69

Distinct elements



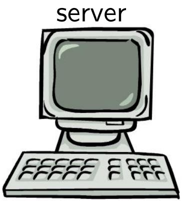
to: cs.princeton.edu
from: 69.172.200.24

packet

Addresses seen:

18.9.22.69

Distinct elements



to:	cs.princeton.edu
from:	69.172.200.24

packet

Addresses seen:

18.9.22.69

69.172.200.24

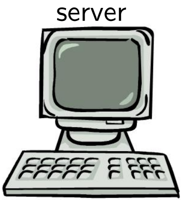
Distinct elements

Addresses seen:

`18.9.22.69`

`69.172.200.24`

Distinct elements



to:	cs.princeton.edu
from:	18.9.22.69

packet

Addresses seen:

18.9.22.69

69.172.200.24

Distinct elements

server

to:	cs.princeton.edu
from:	18.9.22.69

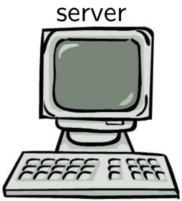
packet

Addresses seen:

18.9.22.69

69.172.200.24

Distinct elements

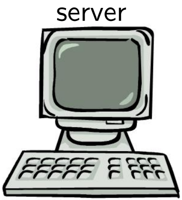


Addresses seen:

`18.9.22.69`

`69.172.200.24`

Distinct elements



to: cs.princeton.edu

from: 106.10.165.51

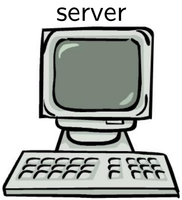
packet

Addresses seen:

18.9.22.69

69.172.200.24

Distinct elements



`to:`	`cs.princeton.edu`
`from:`	`106.10.165.51`

packet

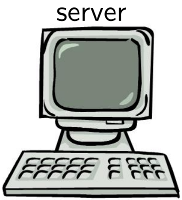
Addresses seen:

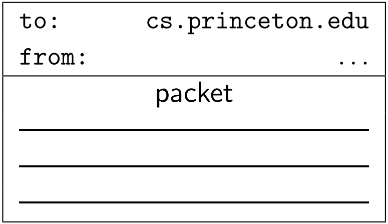
`18.9.22.69`

`69.172.200.24`

`106.10.165.51`

Distinct elements





Addresses seen:

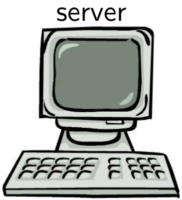
18.9.22.69

69.172.200.24

106.10.165.51

...

Distinct elements



`to:` `cs.princeton.edu`

`from:` ...

packet

Addresses seen:

`18.9.22.69`

`69.172.200.24`

`106.10.165.51`

...

Goal: Count number of distinct IP addresses that contacted server.

Distinct elements

Two obvious solutions

- Store a bitvector $x \in \{0,1\}^{2^{128}}$ ($x_i = 1$ if we've seen address i)
- Store a hash table: $O(N)$ words of memory

Distinct elements

Two obvious solutions

- Store a bitvector $x \in \{0,1\}^{2^{128}}$ ($x_i = 1$ if we've seen address i)
- Store a hash table: $O(N)$ words of memory

In general: sequence of N integers each in $\{1, \ldots, n\}$.
Can either use $O(N \log n)$ bits of memory, or n bits.

Distinct elements

Two obvious solutions

- Store a bitvector $x \in \{0,1\}^{2^{128}}$ ($x_i = 1$ if we've seen address i)
- Store a hash table: $O(N)$ words of memory

In general: sequence of N integers each in $\{1, \ldots, n\}$.
Can either use $O(N \log n)$ bits of memory, or n bits.

But we can do better!

Distinct elements

KMV algorithm

[Bar-Yossef, Jayram, Kumar, Sivakumar, Trevisan, RANDOM 2002]

also see [Beyer, Gemulla, Haas, Reinwald, Sismanis, Commun. ACM 52(10), 2009]

(first small-space algorithm published is due to [Flajolet, Martin, FOCS 1983])

Guarantee: Let F_0 be the number of distinct elements. Will output a value $\tilde{F}_0$ such that with probability at least 2/3, $|\tilde{F}_0 - F_0| \le \varepsilon F_0$.

Distinct elements

KMV algorithm

[Bar-Yossef, Jayram, Kumar, Sivakumar, Trevisan, RANDOM 2002]

also see [Beyer, Gemulla, Haas, Reinwald, Sismanis, Commun. ACM 52(10), 2009]

(first small-space algorithm published is due to [Flajolet, Martin, FOCS 1983])

Guarantee: Let F_0 be the number of distinct elements. Will output a value $\tilde{F}_0$ such that with probability at least 2/3, $|\tilde{F}_0 - F_0| \leq \varepsilon F_0$.

KMV algorithm

1. Pick random hash function $h : [n] \to [0, 1]$
2. Maintain $k = \Theta(1/\varepsilon^2)$ smallest distinct hash values seen in stream $X_1 < X_2 < \ldots < X_k$
3. **if** seen less than k distinct hash values at end of stream, output number of distinct hash values seen

   else output k/X_k

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**

Stream: 5

$$h(5) = 0.00239167$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167

Stream: 5

$$h(5) = 0.00239167$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167

Stream: 5 1

$$h(1) = 0.973811$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
0.973811

Stream: 5 1

$$h(1) = 0.973811$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
0.973811

Stream: 5 1 5

$$h(5) = 0.00239167$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
0.973811

Stream: 5 1 5

$h(5) = 0.00239167$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
0.973811

Stream: 5 1 5 2

$$h(2) = 0.0929362$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
0.973811
0.0929362

Stream: 5 1 5 2

$$h(2) = 0.0929362$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
0.973811
0.0929362

Stream: 5 1 5 2 7

$$h(7) = 0.425028$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
0.425028

Stream: 5 1 5 2 7

$h(7) = 0.425028$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
0.425028

Stream: 5 1 5 2 7 1

$$h(1) = 0.973811$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
0.425028

Stream: 5 1 5 2 7 1

$$h(1) = 0.973811$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
0.425028

Stream: 5 1 5 2 7 1 3

$$h(3) = 0.770643$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
0.425028

Stream: 5 1 5 2 7 1 3

$h(3) = 0.770643$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
0.425028

Stream: 5 1 5 2 7 1 3 8

$$h(8) = 0.223476$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
0.223476

Stream: 5 1 5 2 7 1 3 8

$h(8) = 0.223476$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
0.223476

Stream: 5 1 5 2 7 1 3 8 4

$h(4) = 0.204447$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
~~0.223476~~
0.204447

Stream: 5 1 5 2 7 1 3 8 4

$h(4) = 0.204447$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
~~0.223476~~
0.204447

Stream: 5 1 5 2 7 1 3 8 4 6

$$h(6) = 0.88464$$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
~~0.223476~~
0.204447

Stream: 5 1 5 2 7 1 3 8 4 6

$h(6) = 0.88464$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
~~0.223476~~
0.204447

Stream: 5 1 5 2 7 1 3 8 4 6

output $k/X_k = 3/.204447 = 14.6737$

Distinct elements — KMV algorithm example

$k = 3$

k **minimum (hash) values (i.e. kmv)**
0.00239167
~~0.973811~~
0.0929362
~~0.425028~~
~~0.223476~~
0.204447

Stream: 5 1 5 2 7 1 3 8 4 6

output $k/X_k = 3/.204447 = 14.6737$

Note: true answer is 8

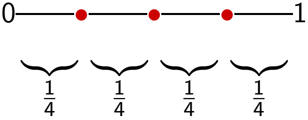
Distinct elements — Why does KMV work?

Question: Suppose we pick t random numbers $X_1, \ldots, X_t$ in the range $[0, 1]$. What do we expect the kth smallest X_i to be on average?

Distinct elements — Why does KMV work?

Question: Suppose we pick t random numbers $X_1, \ldots, X_t$ in the range $[0, 1]$. What do we expect the kth smallest X_i to be on average?

Answer: $\frac{k}{t+1}$

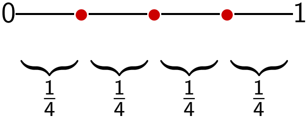


We expect the t random numbers to be evenly spaced when sorted from smallest to largest, so kth smallest is expected to be $k/(t+1)$.

Distinct elements — Why does KMV work?

Question: Suppose we pick t random numbers $X_1, \ldots, X_t$ in the range $[0, 1]$. What do we expect the kth smallest X_i to be on average?

Answer: $\frac{k}{t+1}$



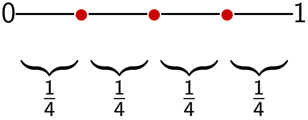
We expect the t random numbers to be evenly spaced when sorted from smallest to largest, so kth smallest is expected to be $k/(t+1)$.

For us $t = F_0$, so if things go according to expectation then $k/X_k = F_0 + 1$

Distinct elements — Why does KMV work?

Question: Suppose we pick t random numbers $X_1, \ldots, X_t$ in the range $[0, 1]$. What do we expect the kth smallest X_i to be on average?

Answer: $\frac{k}{t+1}$



We expect the t random numbers to be evenly spaced when sorted from smallest to largest, so kth smallest is expected to be $k/(t+1)$.

For us $t = F_0$, so if things go according to expectation then $k/X_k = F_0 + 1$
Of course, things don't always go exactly according to expectation

Distinct elements — KMV analysis

Assume $F_0 > k$.

Distinct elements — KMV analysis

Assume $F_0 > k$. Define good events:

- **Event $\mathcal{E}_1$:** fewer than k elements hash below $k/(F_0(1+\varepsilon))$
- **Event $\mathcal{E}_2$:** at least k elements hash below $k/(F_0(1-\varepsilon))$

As long as both $\mathcal{E}_1, \mathcal{E}_2$ happen, $(1-\varepsilon)F_0 \leq \tilde{F}_0 \leq (1+\varepsilon)F_0$, as we want

Distinct elements — KMV analysis

Assume $F_0 > k$. Define good events:

- **Event $\mathcal{E}_1$:** fewer than k elements hash below $k/(F_0(1+\varepsilon))$
- **Event $\mathcal{E}_2$:** at least k elements hash below $k/(F_0(1-\varepsilon))$

As long as both $\mathcal{E}_1, \mathcal{E}_2$ happen, $(1-\varepsilon)F_0 \leq \tilde{F}_0 \leq (1+\varepsilon)F_0$, as we want

What's $\Pr[\neg\mathcal{E}_1]$**?**

Distinct elements — KMV analysis

Assume $F_0 > k$. Define good events:

- **Event $\mathcal{E}_1$:** fewer than k elements hash below $k/(F_0(1+\varepsilon))$
- **Event $\mathcal{E}_2$:** at least k elements hash below $k/(F_0(1-\varepsilon))$

As long as both $\mathcal{E}_1, \mathcal{E}_2$ happen, $(1-\varepsilon)F_0 \leq \tilde{F}_0 \leq (1+\varepsilon)F_0$, as we want

What's $\Pr[\neg\mathcal{E}_1]$**?**
Y_i indicator random variable for event $h(i\text{th item}) \leq k/(F_0(1+\varepsilon))$
$Y = \sum_{i=1}^{F_0} Y_i$ is the number of items below threshold

Distinct elements — KMV analysis

Assume $F_0 > k$. Define good events:

- **Event $\mathcal{E}_1$:** fewer than k elements hash below $k/(F_0(1+\varepsilon))$
- **Event $\mathcal{E}_2$:** at least k elements hash below $k/(F_0(1-\varepsilon))$

As long as both $\mathcal{E}_1, \mathcal{E}_2$ happen, $(1-\varepsilon)F_0 \le \tilde{F}_0 \le (1+\varepsilon)F_0$, as we want

What's $\Pr[\neg\mathcal{E}_1]$**?**
Y_i indicator random variable for event $h(i\text{th item}) \le k/(F_0(1+\varepsilon))$
$Y = \sum_{i=1}^{F_0} Y_i$ is the number of items below threshold

$$\mathbb{E}Y = \sum_{i=1}^{F_0} \mathbb{E}Y_i = k/(1+\varepsilon)$$

Distinct elements — KMV analysis

Assume $F_0 > k$. Define good events:

- **Event $\mathcal{E}_1$:** fewer than k elements hash below $k/(F_0(1+\varepsilon))$
- **Event $\mathcal{E}_2$:** at least k elements hash below $k/(F_0(1-\varepsilon))$

As long as both $\mathcal{E}_1, \mathcal{E}_2$ happen, $(1-\varepsilon)F_0 \le \tilde{F}_0 \le (1+\varepsilon)F_0$, as we want

What's $\Pr[\neg\mathcal{E}_1]$**?**

Y_i indicator random variable for event $h(i\text{th item}) \le k/(F_0(1+\varepsilon))$

$Y = \sum_{i=1}^{F_0} Y_i$ is the number of items below threshold

$$\mathbb{E}Y = \sum_{i=1}^{F_0} \mathbb{E}Y_i = k/(1+\varepsilon)$$

$$\mathbf{Var}[Y] = \sum_{i=1}^{F_0} \mathbf{Var}[Y_i] \le k/(1+\varepsilon)$$

Distinct elements — KMV analysis

Assume $F_0 > k$. Define good events:

- **Event $\mathcal{E}_1$:** fewer than k elements hash below $k/(F_0(1+\varepsilon))$
- **Event $\mathcal{E}_2$:** at least k elements hash below $k/(F_0(1-\varepsilon))$

As long as both $\mathcal{E}_1, \mathcal{E}_2$ happen, $(1-\varepsilon)F_0 \leq \tilde{F}_0 \leq (1+\varepsilon)F_0$, as we want

What's $\Pr[\neg\mathcal{E}_1]$**?**
Y_i indicator random variable for event $h(i\text{th item}) \leq k/(F_0(1+\varepsilon))$
$Y = \sum_{i=1}^{F_0} Y_i$ is the number of items below threshold

$$\mathbb{E}Y = \sum_{i=1}^{F_0} \mathbb{E}Y_i = k/(1+\varepsilon)$$

$$\mathbf{Var}[Y] = \sum_{i=1}^{F_0} \mathbf{Var}[Y_i] \leq k/(1+\varepsilon)$$

Chebyshev: $\Pr(\neg\mathcal{E}_1) = \Pr(Y \geq k) \leq \mathbf{Var}[Y]/(k - \mathbb{E}Y)^2 \leq (1+\varepsilon)/(\varepsilon^2 k)$

Distinct elements

See algorithms with even better space performance in

- [Durand, Flajolet, 2003] (with implementation!)
- [Flajolet, Fusy, Gandouet, Meunier, Disc. Math. and Theor. Comp. Sci., 2007] (with implementation!)
- [Kane, N., Woodruff, 2010]

Streaming Data Mining

1. Items
    - Item frequencies
    - Distinct elements
    - **Moment estimation**
2. Vectors
    - Dimensionality reduction
    - k-means
    - Linear Regression
3. Matrices
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

Moment estimation

Problem: Compute value $\tilde{F}_p$ which, with 2/3 probability, lies in the interval $[(1-\varepsilon)F_p, (1+\varepsilon)F_p]$.

$$F_p = \|f\|_p^p = \sum_{i=1}^{n} |f(i)|^p$$

Moment estimation

Problem: Compute value $\tilde{F}_p$ which, with $2/3$ probability, lies in the interval $[(1-\varepsilon)F_p, (1+\varepsilon)F_p]$.

$$F_p = \|f\|_p^p = \sum_{i=1}^{n} |f(i)|^p$$

- $p = 0$**:** Distinct elements
- $p = \infty$**:** Most frequent item (actually "$F_\infty^{1/\infty}$", or $\|f\|_\infty$)
- **larger** p**:** Closer approximation to F_∞

Moment estimation

Problem: Compute value $\tilde{F}_p$ which, with $2/3$ probability, lies in the interval $[(1-\varepsilon)F_p, (1+\varepsilon)F_p]$.

$$F_p = \|f\|_p^p = \sum_{i=1}^{n} |f(i)|^p$$

- $p = 0$**:** Distinct elements
- $p = \infty$**:** Most frequent item (actually "$F_\infty^{1/\infty}$", or $\|f\|_\infty$)
- **larger** p**:** Closer approximation to F_∞

Unfortunately known that $\mathrm{poly}(n)$ space required for $p > 2$.

[Bar-Yossef, Jayram, Kumar, Sivakumar, JCSS 68(4), 2004]

[Chakrabarti, Khot, Sun, CCC 2003]

Moment estimation — $p = 2$

A look at $p = 2$

- $p = 2$ is as close as we can get to $\|f\|_\infty$ while not taking polynomial space ("is there an outlier?")
- Linear sketches give a way to estimate dot product

  (read: cosine similarity)

Moment estimation — $p = 2$

A look at $p = 2$

- $p = 2$ is as close as we can get to $\|f\|_\infty$ while not taking polynomial space ("is there an outlier?")
- Linear sketches give a way to estimate dot product

  (read: cosine similarity)

  Suppose x, y are unit vectors and $\|\tilde{z}\|_2$ is some estimate $(1 \pm \varepsilon)\|z\|_2$

Moment estimation — $p = 2$

A look at $p = 2$

- $p = 2$ is as close as we can get to $\|f\|_\infty$ while not taking polynomial space ("is there an outlier?")
- Linear sketches give a way to estimate dot product

  (read: cosine similarity)

  Suppose x, y are unit vectors and $\|\tilde{z}\|_2$ is some estimate $(1 \pm \varepsilon)\|z\|_2$

  Recall $\|x - y\|_2^2 = \|x\|_2^2 + \|y\|_2^2 - 2\langle x, y\rangle$

  so $\frac{1}{2} \cdot (\|\widetilde{(x - y)}\|_2^2 - \|\tilde{x}\|_2^2 - \|\tilde{y}\|_2^2) = \langle x, y\rangle \pm 2\varepsilon$

Moment estimation — $p = 2$

TZ sketch

[Thorup, Zhang, SODA 2004]

also see [Charikar, Chen, Farach-Colton, ICALP 2002]

(first small-space algorithm published is due to [Alon, Matias, Szegedy, STOC 1996])

TZ sketch

- Initialize counters $A_1, \ldots, A_k$ to 0 for $k = \Theta(1/\varepsilon^2)$
- Pick random hash functions $h : [n] \to [k]$ and $\sigma : [n] \to \{-1, 1\}$
- **upon update** $f_i \leftarrow f_i + v$ add $\sigma(i) \cdot v$ to $A_{h(i)}$
- **output** $\sum_{j=1}^{k} A_j^2$

Just $O(1/\varepsilon^2)$ words of space and constant update time!

Moment estimation — TZ sketch

$n = 10$

h values

$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values

$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	0	0	0

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
0	0	0	0	0	0	0	0	0	0

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$$A = \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{A_1} & \multicolumn{1}{c}{A_2} & \multicolumn{1}{c}{A_3} & \multicolumn{1}{c}{A_4} & \multicolumn{1}{c}{A_5} \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$f = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{f(1)} & \multicolumn{1}{c}{f(2)} & \multicolumn{1}{c}{f(3)} & \multicolumn{1}{c}{f(4)} & \multicolumn{1}{c}{f(5)} & \multicolumn{1}{c}{f(6)} & \multicolumn{1}{c}{f(7)} & \multicolumn{1}{c}{f(8)} & \multicolumn{1}{c}{f(9)} & \multicolumn{1}{c}{f(10)} \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \multicolumn{1}{c}{+1} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \end{array}$$

Moment estimation — TZ sketch

$n = 10$

$k = 5$

$$A = \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{A_1} & \multicolumn{1}{c}{A_2} & \multicolumn{1}{c}{A_3} & \multicolumn{1}{c}{A_4} & \multicolumn{1}{c}{A_5} \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$$f = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{f(1)} & \multicolumn{1}{c}{f(2)} & \multicolumn{1}{c}{f(3)} & \multicolumn{1}{c}{f(4)} & \multicolumn{1}{c}{f(5)} & \multicolumn{1}{c}{f(6)} & \multicolumn{1}{c}{f(7)} & \multicolumn{1}{c}{f(8)} & \multicolumn{1}{c}{f(9)} & \multicolumn{1}{c}{f(10)} \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \multicolumn{1}{c}{+1} \end{array}$$

Moment estimation — TZ sketch

$n = 10$

h values	σ values
$h(1)$: 4	$\sigma(1)$: $+1$
$h(2)$: 1	$\sigma(2)$: $+1$
$h(3)$: 3	$\sigma(3)$: $+1$
$h(4)$: 1	$\sigma(4)$: -1
$h(5)$: 3	$\sigma(5)$: $+1$
$h(6)$: 4	$\sigma(6)$: $+1$
$h(7)$: 4	$\sigma(7)$: -1
$h(8)$: 4	$\sigma(8)$: $+1$
$h(9)$: 2	$\sigma(9)$: $+1$
$h(10)$: 4	$\sigma(10)$: $+1$

$k = 5$

	A_1	A_2	A_3	A_4	A_5
$A =$	0	0	0	0	0
				$+1$	

	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
$f =$	1	0	0	0	0	0	0	0	0	0
	$+1$									

Moment estimation — TZ sketch

$n = 10$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	0	1	0
			$+1$	

h values

$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values

$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
1	0	0	0	0	0	0	0	0	0
$+1$									

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$$A = \begin{array}{|c|c|c|c|c|} \multicolumn{1}{c}{A_1} & \multicolumn{1}{c}{A_2} & \multicolumn{1}{c}{A_3} & \multicolumn{1}{c}{A_4} & \multicolumn{1}{c}{A_5} \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

$$f = \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \multicolumn{1}{c}{f(1)} & \multicolumn{1}{c}{f(2)} & \multicolumn{1}{c}{f(3)} & \multicolumn{1}{c}{f(4)} & \multicolumn{1}{c}{f(5)} & \multicolumn{1}{c}{f(6)} & \multicolumn{1}{c}{f(7)} & \multicolumn{1}{c}{f(8)} & \multicolumn{1}{c}{f(9)} & \multicolumn{1}{c}{f(10)} \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{-4} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \end{array}$$

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	0	1	0

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
1	0	0	0	-4	0	0	0	0	0
				-4					

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$$A = \begin{array}{ccccc} A_1 & A_2 & A_3 & A_4 & A_5 \\ \hline \boxed{0} & \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} \\ & & -4 & & \end{array}$$

$$f = \begin{array}{cccccccccc} f(1) & f(2) & f(3) & f(4) & f(5) & f(6) & f(7) & f(8) & f(9) & f(10) \\ \hline \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{-4} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} \\ & & & & -4 & & & & & \end{array}$$

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	1	0
		-4		

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
1	0	0	0	-4	0	0	0	0	0
				-4					

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	1	0

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
1	0	0	0	-4	0	0	0	0	0

-5

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	1	0

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-4	0	0	0	-4	0	0	0	0	0
-5									

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	1	0
			-5	

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-4	0	0	0	-4	0	0	0	0	0
-5									

Moment estimation — TZ sketch

$n = 10$

$k = 5$

	A_1	A_2	A_3	A_4	A_5
$A =$	0	0	-4	-4	0
				-5	

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
$f =$	-4	0	0	0	-4	0	0	0	0	0
	-5									

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

	A_1	A_2	A_3	A_4	A_5
$A =$	0	0	-4	-4	0

	$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
$f =$	-4	0	0	0	-4	0	0	0	0	0
							$+2$			

Moment estimation — TZ sketch

$n = 10$

h values	σ values
$h(1)$: 4	$\sigma(1)$: $+1$
$h(2)$: 1	$\sigma(2)$: $+1$
$h(3)$: 3	$\sigma(3)$: $+1$
$h(4)$: 1	$\sigma(4)$: -1
$h(5)$: 3	$\sigma(5)$: $+1$
$h(6)$: 4	$\sigma(6)$: $+1$
$h(7)$: 4	$\sigma(7)$: -1
$h(8)$: 4	$\sigma(8)$: $+1$
$h(9)$: 2	$\sigma(9)$: $+1$
$h(10)$: 4	$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	-4	0

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-4	0	0	0	-4	0	2	0	0	0

$+2$

Moment estimation — TZ sketch

$n = 10$

h values
$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values
$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	-4	0
			-2	

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-4	0	0	0	-4	0	2	0	0	0
						$+2$			

Moment estimation — TZ sketch

$n = 10$

h values	σ values
$h(1)$: 4	$\sigma(1)$: $+1$
$h(2)$: 1	$\sigma(2)$: $+1$
$h(3)$: 3	$\sigma(3)$: $+1$
$h(4)$: 1	$\sigma(4)$: -1
$h(5)$: 3	$\sigma(5)$: $+1$
$h(6)$: 4	$\sigma(6)$: $+1$
$h(7)$: 4	$\sigma(7)$: -1
$h(8)$: 4	$\sigma(8)$: $+1$
$h(9)$: 2	$\sigma(9)$: $+1$
$h(10)$: 4	$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	-6	0
			-2	

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-4	0	0	0	-4	0	2	0	0	0
						$+2$			

Moment estimation — TZ sketch

$n = 10$

h values

$h(1)$: 4
$h(2)$: 1
$h(3)$: 3
$h(4)$: 1
$h(5)$: 3
$h(6)$: 4
$h(7)$: 4
$h(8)$: 4
$h(9)$: 2
$h(10)$: 4

σ values

$\sigma(1)$: $+1$
$\sigma(2)$: $+1$
$\sigma(3)$: $+1$
$\sigma(4)$: -1
$\sigma(5)$: $+1$
$\sigma(6)$: $+1$
$\sigma(7)$: -1
$\sigma(8)$: $+1$
$\sigma(9)$: $+1$
$\sigma(10)$: $+1$

$k = 5$

$A =$

A_1	A_2	A_3	A_4	A_5
0	0	-4	-6	0

output $0^2 + 0^2 + (-4)^2 + (-6)^2 + 0^2 = 52$

(note $\|f\|_2^2 = (-4)^2 + (-4)^2 + 2^2 = 38$**)**

$f =$

$f(1)$	$f(2)$	$f(3)$	$f(4)$	$f(5)$	$f(6)$	$f(7)$	$f(8)$	$f(9)$	$f(10)$
-4	0	0	0	-4	0	2	0	0	0

Moment estimation — TZ sketch analysis

Why does it work?

Moment estimation — TZ sketch analysis

Why does it work?

- Let $\delta_{i,r}$ be an indicator random variable for the event $h(i) = r$

Moment estimation — TZ sketch analysis

Why does it work?

- Let $\delta_{i,r}$ be an indicator random variable for the event $h(i) = r$
- $A_r = \sum_{i=1}^{n} \delta_{i,r}\sigma(i)f(i)$

Moment estimation — TZ sketch analysis

Why does it work?

- Let $\delta_{i,r}$ be an indicator random variable for the event $h(i) = r$
- $A_r = \sum_{i=1}^{n} \delta_{i,r}\sigma(i)f(i)$
  $\Rightarrow A_r^2 = \sum_{i=1}^{n} \delta_{i,r}f(i)^2 + \sum_{i \neq j} \delta_{i,r}\delta_{j,r}\sigma(i)\sigma(j)f(i)f(j)$

Moment estimation — TZ sketch analysis

Why does it work?

- Let $\delta_{i,r}$ be an indicator random variable for the event $h(i) = r$
- $A_r = \sum_{i=1}^{n} \delta_{i,r}\sigma(i)f(i)$
  $\Rightarrow A_r^2 = \sum_{i=1}^{n} \delta_{i,r}f(i)^2 + \sum_{i\neq j} \delta_{i,r}\delta_{j,r}\sigma(i)\sigma(j)f(i)f(j)$
  $\Rightarrow \mathbb{E}A_r^2 = \sum_{i=1}^{n}(\mathbb{E}\delta_{i,r})f(i)^2 + \sum_{i\neq j}(\mathbb{E}\delta_{i,r}\delta_{j,r})(\mathbb{E}\sigma(i)\mathbb{E}\sigma(j))f(i)f(j)$

Moment estimation — TZ sketch analysis

Why does it work?

- Let $\delta_{i,r}$ be an indicator random variable for the event $h(i) = r$
- $A_r = \sum_{i=1}^{n} \delta_{i,r}\sigma(i)f(i)$
  $\Rightarrow A_r^2 = \sum_{i=1}^{n} \delta_{i,r}f(i)^2 + \sum_{i \neq j} \delta_{i,r}\delta_{j,r}\sigma(i)\sigma(j)f(i)f(j)$
  $\Rightarrow \mathbb{E}A_r^2 = \sum_{i=1}^{n}(\mathbb{E}\delta_{i,r})f(i)^2 + \sum_{i \neq j}(\mathbb{E}\delta_{i,r}\delta_{j,r})(\mathbb{E}\sigma(i)\mathbb{E}\sigma(j))f(i)f(j)$
  $\Rightarrow \mathbb{E}A_r^2 = \|f\|_2^2/k$

Moment estimation — TZ sketch analysis

Why does it work?

- Let $\delta_{i,r}$ be an indicator random variable for the event $h(i) = r$
- $A_r = \sum_{i=1}^{n} \delta_{i,r}\sigma(i)f(i)$
  $\Rightarrow A_r^2 = \sum_{i=1}^{n} \delta_{i,r}f(i)^2 + \sum_{i \neq j} \delta_{i,r}\delta_{j,r}\sigma(i)\sigma(j)f(i)f(j)$
  $\Rightarrow \mathbb{E}A_r^2 = \sum_{i=1}^{n}(\mathbb{E}\delta_{i,r})f(i)^2 + \sum_{i \neq j}(\mathbb{E}\delta_{i,r}\delta_{j,r})(\mathbb{E}\sigma(i)\mathbb{E}\sigma(j))f(i)f(j)$
  $\Rightarrow \mathbb{E}A_r^2 = \|f\|_2^2/k$
  $\Rightarrow \mathbb{E}\sum_{r=1}^{k} A_j^2 = \sum_{r=1}^{k} \mathbb{E}A_r^2 = \|f\|_2^2$

Moment estimation — TZ sketch analysis

Expectation is unbiased . . . what about the variance?

- $\mathbf{Var}(\|A\|_2^2) = \mathbb{E}(\|A\|_2^2 - \mathbb{E}\|A\|_2^2)^2 = \mathbb{E}\|A\|_2^4 - \|f\|_2^4$
- After some calculations I'll omit

$$\mathrm{Var}(\|A\|_2^2) = \sum_{r=1}^{k} \sum_{i \neq j} (\mathbb{E}\delta_{i,r}\delta_{j,r}) f_i^2 f_j^2 \leq \sum_{r=1}^{k} \frac{1}{k^2} (\|f\|_2^2)^2 = \frac{\|f\|_2^4}{k}$$

Moment estimation — TZ sketch analysis

Expectation is unbiased ... **what about the variance?**

- $\mathbf{Var}(\|A\|_2^2) = \mathbb{E}(\|A\|_2^2 - \mathbb{E}\|A\|_2^2)^2 = \mathbb{E}\|A\|_2^4 - \|f\|_2^4$
- After some calculations I'll omit

$$\mathrm{Var}(\|A\|_2^2) = \sum_{r=1}^{k} \sum_{i \neq j} (\mathbb{E}\delta_{i,r}\delta_{j,r}) f_i^2 f_j^2 \leq \sum_{r=1}^{k} \frac{1}{k^2} (\|f\|_2^2)^2 = \frac{\|f\|_2^4}{k}$$

- Chebyshev: $\Pr(|\|A\|_2^2 - \mathbb{E}\|A\|_2^2| > \varepsilon\|f\|_2^2) < \frac{\mathbf{Var}(\|A\|_2^2)}{\varepsilon^2\|f\|_2^4}$

Moment estimation — TZ sketch analysis

Expectation is unbiased . . . **what about the variance?**

- $\mathbf{Var}(\|A\|_2^2) = \mathbb{E}(\|A\|_2^2 - \mathbb{E}\|A\|_2^2)^2 = \mathbb{E}\|A\|_2^4 - \|f\|_2^4$
- After some calculations I'll omit

$$\mathrm{Var}(\|A\|_2^2) = \sum_{r=1}^{k} \sum_{i \neq j} (\mathbb{E}\delta_{i,r}\delta_{j,r}) f_i^2 f_j^2 \leq \sum_{r=1}^{k} \frac{1}{k^2} (\|f\|_2^2)^2 = \frac{\|f\|_2^4}{k}$$

- Chebyshev: $\Pr(|\|A\|_2^2 - \mathbb{E}\|A\|_2^2| > \varepsilon\|f\|_2^2) < \frac{\mathbf{Var}(\|A\|_2^2)}{\varepsilon^2\|f\|_2^4}$
- So, we can set $k = 3/\varepsilon^2$ to get error probability $1/3$

Streaming Data Mining

1. Items
    - Item frequencies
    - Distinct elements
    - Moment estimation
2. Vectors
    - **Dimensionality reduction**
    - k-means
    - Linear Regression
3. Matrices
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

Dimensionality reduction

Many problems, as one would expect, become computationally harder as the dimensionality of the underlying input data grows

- Nearest neighbor search (exponential preprocessing time to get sublinear query)
- Optimization problems for geometric problems: closest pair, diameter, minimum spanning tree, . . .
- Linear algebra problems: regression, low-rank approximation
- Clustering

Dimensionality reduction

Many problems, as one would expect, become computationally harder as the dimensionality of the underlying input data grows

- Nearest neighbor search (exponential preprocessing time to get sublinear query)
- Optimization problems for geometric problems: closest pair, diameter, minimum spanning tree, . . .
- Linear algebra problems: regression, low-rank approximation
- Clustering

How can we reduce dimensionality in such a way that we can still (approximately) solve the above problems above?

Dimensionality reduction

Good news when underlying distance metric is $\|\cdot\|_2$

Theorem (Johnson-Lindenstrauss (JL) lemma, 1984)

For every $0 < \varepsilon \leq 1/2$ *and set of points* $x_1, \ldots, x_N \in \mathbb{R}^n$*, there exists a matrix* $A \in \mathbb{R}^{m \times n}$ *for* $m = O(\varepsilon^{-2} \log N)$ *such that*

$$\forall i \neq j,\ (1-\varepsilon)\|x_i - x_j\|_2 \leq \|Ax_i - Ax_j\|_2 \leq (1+\varepsilon)\|x_i - x_j\|_2$$

Dimensionality reduction

Good news when underlying distance metric is $\|\cdot\|_2$

Theorem (Johnson-Lindenstrauss (JL) lemma, 1984)

For every $0 < \varepsilon \leq 1/2$ *and set of points* $x_1, \ldots, x_N \in \mathbb{R}^n$, *there exists a matrix* $A \in \mathbb{R}^{m \times n}$ *for* $m = O(\varepsilon^{-2} \log N)$ *such that*

$$\forall i \neq j,\ (1-\varepsilon)\|x_i - x_j\|_2 \leq \|Ax_i - Ax_j\|_2 \leq (1+\varepsilon)\|x_i - x_j\|_2$$

Bad news when underlying distance metric is not $\|\cdot\|_2$

Work of [Naor, Johnson, SODA 2009] shows that $\|\cdot\|_2$ (or metric spaces "close" to it) are the only spaces where we could hope to embed into $O(\log N)$ dimensions with any constant distortion guarantee.

Dimensionality reduction

How do you prove the JL lemma?

Theorem (Johnson-Lindenstrauss (JL) lemma, 1984)

For every $0 < \varepsilon \le 1/2$ *and set of points* $x_1, \ldots, x_N \in \mathbb{R}^n$*, there exists a matrix* $A \in \mathbb{R}^{m \times n}$ *for* $m = O(\varepsilon^{-2} \log N)$ *such that*

$$\forall i \neq j,\ (1-\varepsilon)\|x_i - x_j\|_2 \le \|Ax_i - Ax_j\|_2 \le (1+\varepsilon)\|x_i - x_j\|_2$$

Dimensionality reduction

How do you prove the JL lemma?

Theorem (Johnson-Lindenstrauss (JL) lemma, 1984)

For every $0 < \varepsilon \leq 1/2$ *and set of points* $x_1, \ldots, x_N \in \mathbb{R}^n$*, there exists a matrix* $A \in \mathbb{R}^{m \times n}$ *for* $m = O(\varepsilon^{-2} \log N)$ *such that*

$$\forall i \neq j, \ (1-\varepsilon)\|x_i - x_j\|_2 \leq \|Ax_i - Ax_j\|_2 \leq (1+\varepsilon)\|x_i - x_j\|_2$$

Use the Distributional JL (DJL) lemma

Theorem

For every $0 < \varepsilon, \delta \leq 1/2$ *there exists a distribution* $\mathcal{D}_{\varepsilon,\delta}$ *over* $R^{m \times n}$ *for* $m = O(\varepsilon^{-2} \log(1/\delta))$ *such that for any* $x \in \mathbb{R}^n$ *with* $\|x\|_2 = 1$

$$\Pr_{A \sim \mathcal{D}_{\varepsilon,\delta}} \left(\left| \|Ax\|_2^2 - 1 \right\| > \varepsilon \right) < \delta$$

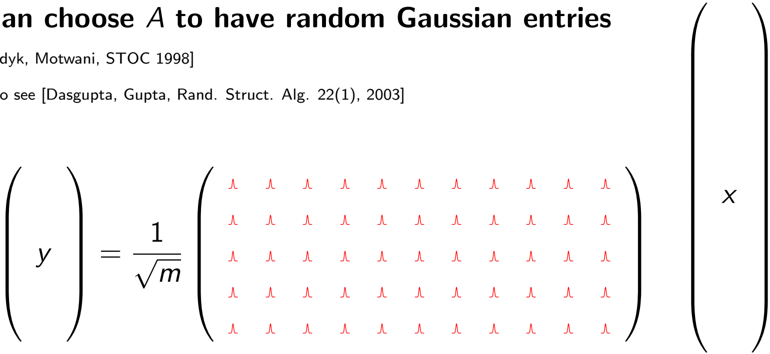
Proof of JL via DJL: Set $\delta = 1/N^2$ so that $(x_i - x_j)/\|x_i - x_j\|_2$ is preserved w.p. $1 - 1/N^2$. Union bound over all $\binom{N}{2}$ $i < j$.

Dimensionality reduction

Can choose A to have random Gaussian entries

[Indyk, Motwani, STOC 1998]

also see [Dasgupta, Gupta, Rand. Struct. Alg. 22(1), 2003]



⋀ has density function $p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$

Can choose $m = (4 + o(1))\varepsilon^{-2} \ln N$

Dimensionality reduction

Can choose A to have random Gaussian entries

Dimensionality reduction

Can choose A to have random Gaussian entries

Sketch of why it works:

- Recall, want $\Pr(|\|Ax\|_2^2 - 1| > \varepsilon) < \delta$ for all x with unit norm.

Dimensionality reduction

Can choose A to have random Gaussian entries

Sketch of why it works:

- Recall, want $\Pr(|\|Ax\|_2^2 - 1| > \varepsilon) < \delta$ for all x with unit norm.
- We have $\|Ax\|_2^2 - 1 = x^T A^T Ax - 1$

Dimensionality reduction

Can choose A to have random Gaussian entries

Sketch of why it works:

- Recall, want $\Pr(|\|Ax\|_2^2 - 1| > \varepsilon) < \delta$ for all x with unit norm.
- We have $\|Ax\|_2^2 - 1 = x^T A^T Ax - 1$
- $x^T A^T Ax = (1/m) \cdot \sum_{r=1}^m \sum_{i \neq j} g_{r,i} g_{r,j} x_i x_j$. Let $g = (g_{1,1}, \ldots, g_{1,n}, g_{2,1}, \ldots)$ be the vector of rows of A concatenated.

Dimensionality reduction

Can choose A to have random Gaussian entries

Sketch of why it works:

- Recall, want $\Pr(|\|Ax\|_2^2 - 1| > \varepsilon) < \delta$ for all x with unit norm.
- We have $\|Ax\|_2^2 - 1 = x^T A^T Ax - 1$
- $x^T A^T Ax = (1/m) \cdot \sum_{r=1}^{m} \sum_{i \neq j} g_{r,i} g_{r,j} x_i x_j$. Let $g = (g_{1,1}, \ldots, g_{1,n}, g_{2,1}, \ldots)$ be the vector of rows of A concatenated.
- This is $(1/m) \cdot g^T Bg$ where B is a block-diagonal matrix with m blocks each equaling xx^T. Orthogonal change of basis!

Dimensionality reduction

Can choose A to have random Gaussian entries

Sketch of why it works:

- Recall, want $\Pr(|\|Ax\|_2^2 - 1| > \varepsilon) < \delta$ for all x with unit norm.
- We have $\|Ax\|_2^2 - 1 = x^T A^T Ax - 1$
- $x^T A^T Ax = (1/m) \cdot \sum_{r=1}^m \sum_{i \neq j} g_{r,i} g_{r,j} x_i x_j$. Let $g = (g_{1,1}, \ldots, g_{1,n}, g_{2,1}, \ldots)$ be the vector of rows of A concatenated.
- This is $(1/m) \cdot g^T Bg$ where B is a block-diagonal matrix with m blocks each equaling xx^T. Orthogonal change of basis!
- $g^T Bg = g^T Q^T \Lambda Qg = (Qg)^T \Lambda (Qg) = g'^T \Lambda g' = \sum_{i=1}^m g_i^2$. Thus we just want to show $(1/m)(\sum_{i=1}^m (g_i^2 - 1))$ is small with high probability. Can use statistics about the chi-squared distribution.

Dimensionality reduction

Another perhaps useful fact . . .

Dimensionality reduction

Another perhaps useful fact ...

- [Klartag, Mendelson, J. Funct. Anal. 225(1), 2005]: Don't need $m = \Omega((\log N)/\varepsilon^2)$ all the time, but rather can take $m = O(\gamma_2^2(\mathbf{X})/\varepsilon^2)$ where $\mathbf{X} = \{x_i - x_j\}_{i \neq j}$. Here $\gamma_2(\mathbf{X}) = \mathbb{E} \sup_{x \in \mathbf{X}} \langle g, x \rangle^2$ for Gaussian vector g.

Dimensionality reduction

Another perhaps useful fact …

- [Klartag, Mendelson, J. Funct. Anal. 225(1), 2005]: Don't need $m = \Omega((\log N)/\varepsilon^2)$ all the time, but rather can take $m = O(\gamma_2^2(\mathbf{X})/\varepsilon^2)$ where $\mathbf{X} = \{x_i - x_j\}_{i \neq j}$. Here $\gamma_2(\mathbf{X}) = \mathbb{E} \sup_{x \in \mathbf{X}} \langle g, x \rangle^2$ for Gaussian vector g.
- **Bottom line:** Given your data can estimate γ_2 by picking a few independent Gaussian vectors and taking the empirical mean of $\sup_{x \in \mathbf{X}} \langle g, x \rangle^2$. Might improve dimensionality reduction on your data.

Dimensionality reduction

Another perhaps useful fact . . .

- [Klartag, Mendelson, J. Funct. Anal. 225(1), 2005]: Don't need $m = \Omega((\log N)/\varepsilon^2)$ all the time, but rather can take $m = O(\gamma_2^2(\mathbf{X})/\varepsilon^2)$ where $\mathbf{X} = \{x_i - x_j\}_{i \neq j}$. Here $\gamma_2(\mathbf{X}) = \mathbb{E} \sup_{x \in \mathbf{X}} \langle g, x \rangle^2$ for Gaussian vector g.
- **Bottom line:** Given your data can estimate γ_2 by picking a few independent Gaussian vectors and taking the empirical mean of $\sup_{x \in \mathbf{X}} \langle g, x \rangle^2$. Might improve dimensionality reduction on your data.
- **Caveat:** Takes $\Omega(N^2)$ time to estimate γ_2 ($\mathbf{X}$ is the set of pairwise differences), so probably only makes sense when $n \gg N$ (note: Gram-Schmidt is $O(N^2 n)$, so assuming $n < N$ is expensive).

Dimensionality reduction

Can also choose A to have random sign entries

[Achlioptas, PODS 2001]

$$\begin{pmatrix} \\ y \\ \\ \end{pmatrix} = \frac{1}{\sqrt{m}} \begin{pmatrix} + & - & + & + & + & + & + & - & + & + \\ + & + & - & - & + & - & + & - & - & + \\ - & - & - & + & + & + & - & + & - & + \\ + & + & + & + & + & + & - & - & - & - \\ + & - & + & - & - & - & - & - & - & - \end{pmatrix} \begin{pmatrix} x \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

Can choose $m = (4 + o(1))\varepsilon^{-2} \ln N$

Dimensionality reduction

Downside of last two constructions: The preprocessing (performing the embedding) is dense matrix-vector multiplication: $O(mn)$ time

Dimensionality reduction

Downside of last two constructions: The preprocessing (performing the embedding) is dense matrix-vector multiplication: $O(mn)$ time

Achlioptas actually showed that we can take a random sign matrix with 2/3rds of its entries zero (sparser, so faster)

Dimensionality reduction

Downside of last two constructions: The preprocessing (performing the embedding) is dense matrix-vector multiplication: $O(mn)$ time

Achlioptas actually showed that we can take a random sign matrix with 2/3rds of its entries zero (sparser, so faster)

Then there was the the **Fast Johnson-Lindenstrauss Transform*** (FJLT)

[Ailon, Chazelle, SIAM J. Comput. 39(1), 2009]

$$\frac{1}{\sqrt{m}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{\text{random samples}} \underbrace{\begin{pmatrix} & & \\ & \mathsf{H} & \\ & & \end{pmatrix}}_{\text{Hadamard (or Fourier)}} \begin{pmatrix} \pm1 & & & & \\ & \pm1 & & & \\ & & \pm1 & & \\ & & & \pm1 & \\ & & & & \pm1 \end{pmatrix} \begin{pmatrix} \\ x \\ \\ \end{pmatrix}$$

Dimensionality reduction

Downside of last two constructions: The preprocessing (performing the embedding) is dense matrix-vector multiplication: $O(mn)$ time

Achlioptas actually showed that we can take a random sign matrix with 2/3rds of its entries zero (sparser, so faster)

Then there was the the **Fast Johnson-Lindenstrauss Transform*** (FJLT)

[Ailon, Chazelle, SIAM J. Comput. 39(1), 2009]

$$\frac{1}{\sqrt{m}}\underbrace{\begin{pmatrix}1&0&0&0&0\\0&0&0&1&0\\0&1&0&0&0\end{pmatrix}}_{\text{random samples}}\underbrace{\begin{pmatrix}&&\\&\mathsf{H}&\\&&\end{pmatrix}}_{\text{Hadamard (or Fourier)}}\begin{pmatrix}\pm1&&&&\\&\pm1&&&\\&&\pm1&&\\&&&\pm1&\\&&&&\pm1\end{pmatrix}\begin{pmatrix}x\end{pmatrix}$$

***** Actual Ailon-Chazelle construction does something a tad better than the sampling matrix (see their paper).

Dimensionality reduction — FJLT

$$y = \frac{1}{\sqrt{m}} SHDx$$

Recall H is the matrix with $H_{i,j} = (-1)^{(\langle \vec{i}, \vec{j} \rangle \mod 2)}$.

Actually just need any matrix that has entries bounded in magnitude by 1, and is orthogonal when we normalize by $1/\sqrt{n}$.

Dimensionality reduction — FJLT

$$y = \frac{1}{\sqrt{m}} SHDx$$

Recall H is the matrix with $H_{i,j} = (-1)^{(\langle \vec{i}, \vec{j} \rangle \mod 2)}$.

Actually just need any matrix that has entries bounded in magnitude by 1, and is orthogonal when we normalize by $1/\sqrt{n}$.

Why does this work?

- Randomly sampling entries of x has correct ℓ_2^2 expectation but poor variance (e.g. when x has exactly one non-zero coordinate)

Dimensionality reduction — FJLT

$$y = \frac{1}{\sqrt{m}} SHDx$$

Recall H is the matrix with $H_{i,j} = (-1)^{(\langle \vec{i}, \vec{j} \rangle \bmod 2)}$.

Actually just need any matrix that has entries bounded in magnitude by 1, and is orthogonal when we normalize by $1/\sqrt{n}$.

Why does this work?

- Randomly sampling entries of x has correct ℓ_2^2 expectation but poor variance (e.g. when x has exactly one non-zero coordinate)
- Can show $\|HDx\|_\infty \le \sqrt{\log(nN)/n} \approx \sqrt{\log(N)/n}$ w.p. $1 - 1/N^2$

Dimensionality reduction — FJLT

$$y = \frac{1}{\sqrt{m}} SHDx$$

Recall H is the matrix with $H_{i,j} = (-1)^{(\langle \vec{i}, \vec{j} \rangle \mod 2)}$.

Actually just need any matrix that has entries bounded in magnitude by 1, and is orthogonal when we normalize by $1/\sqrt{n}$.

Why does this work?

- Randomly sampling entries of x has correct ℓ_2^2 expectation but poor variance (e.g. when x has exactly one non-zero coordinate)
- Can show $\|HDx\|_\infty \leq \sqrt{\log(nN)/n} \approx \sqrt{\log(N)/n}$ w.p. $1 - 1/N^2$
- Conditioned on above item, apply the Chernoff bound to say sampling by S works with high probability as long as we have enough samples.

Dimensionality reduction — FJLT

$$y = \frac{1}{\sqrt{m}} SHDx$$

Recall H is the matrix with $H_{i,j} = (-1)^{(\langle \vec{i}, \vec{j} \rangle \mod 2)}$.

Actually just need any matrix that has entries bounded in magnitude by 1, and is orthogonal when we normalize by $1/\sqrt{n}$.

Why does this work?

- Randomly sampling entries of x has correct ℓ_2^2 expectation but poor variance (e.g. when x has exactly one non-zero coordinate)
- Can show $\|HDx\|_\infty \leq \sqrt{\log(nN)/n} \approx \sqrt{\log(N)/n}$ w.p. $1 - 1/N^2$
- Conditioned on above item, apply the Chernoff bound to say sampling by S works with high probability as long as we have enough samples.
- Unfortunately requires $m = \Theta(\varepsilon^{-2} \log^2 N)$ samples (extra $\log N$). Can fix by finishing off with a slow matrix (e.g. Gaussian or sign) for an additional $O(m\varepsilon^{-2} \log N)$ time. Total time is $O(n \log n + \varepsilon^{-4} \log^3 N)$.

Dimensionality reduction — FJLT

Improvements to the original FJLT since Ailon and Chazelle's work

Dimensionality reduction — FJLT

Improvements to the original FJLT since Ailon and Chazelle's work

- [Ailon, Liberty, SODA 2008]: $m = O(\varepsilon^{-2} \log N)$ in $O(n \log n + m^{2+\gamma})$ time
- [Ailon, Liberty, SODA 2011], [Krahmer, Ward, SIAM J. Math. Anal. 43(3), 2011]: $m = O(\varepsilon^{-2} \log N \log^4 n)$ in $O(n \log n)$ time (faster for huge N). Construction is actually what we just saw ($(1/\sqrt{m})SHD$), but with a much different-looking analysis (doesn't use DJL).

Dimensionality reduction — FJLT

Improvements to the original FJLT since Ailon and Chazelle's work

- [Ailon, Liberty, SODA 2008]: $m = O(\varepsilon^{-2} \log N)$ in $O(n \log n + m^{2+\gamma})$ time
- [Ailon, Liberty, SODA 2011], [Krahmer, Ward, SIAM J. Math. Anal. 43(3), 2011]: $m = O(\varepsilon^{-2} \log N \log^4 n)$ in $O(n \log n)$ time (faster for huge N). Construction is actually what we just saw $((1/\sqrt{m})SHD)$, but with a much different-looking analysis (doesn't use DJL).
- It's conceivable the $(1/\sqrt{m})SHD$ construction actually gets the optimal $m = O(\varepsilon^{-2} \log N)$, but we just don't know how to prove it yet. So, can try with smaller m but buyer beware.

Dimensionality reduction

One downside to FJLT: kills sparsity

Dimensionality reduction

One downside to FJLT: kills sparsity

Works by randomly spreading mass around to decrease variance for sampling. Takes $O(n \log n)$ time, but dense matrices (Gaussian or sign) take only $O(m \cdot \|x\|_0)$.

Dimensionality reduction

One downside to FJLT: kills sparsity

Works by randomly spreading mass around to decrease variance for sampling. Takes $O(n \log n)$ time, but dense matrices (Gaussian or sign) take only $O(m \cdot \|x\|_0)$.

Who cares about sparsity?

Dimensionality reduction

Who cares about sparsity?

Dimensionality reduction

Who cares about sparsity?

You do

- **Document as bag of words:** $x_i =$ number of occurrences of word i. Compare documents using cosine similarity.

  $d =$ lexicon size; most documents aren't dictionaries
- **Network traffic:** $x_{i,j} = \#$bytes sent from i to j

  $d = 2^{64}$ (2^{256} in IPv6); most servers don't talk to each other
- **User ratings:** x_i is user's score for movie i on Netflix

  $d = \#\text{movies}$; most people haven't rated all movies
- **Streaming:** x receives a stream of updates of the form: "add v to x_i". Maintaining Sx requires calculating $v \cdot Se_i$.
- ...

Dimensionality reduction — SparseJL

One way to embed sparse vectors faster: use sparse matrices

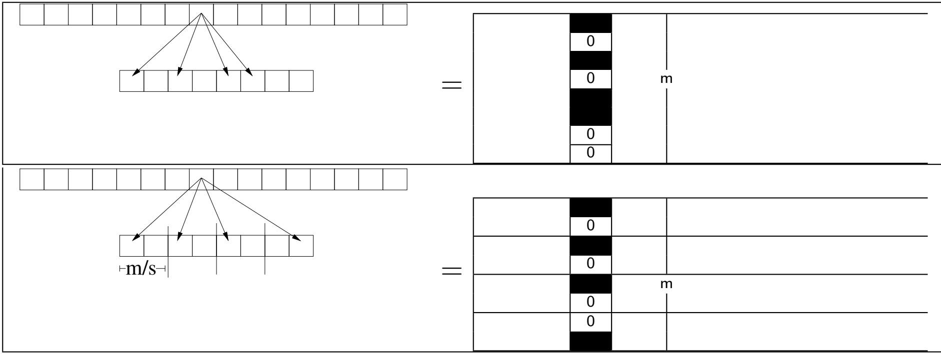
Dimensionality reduction — SparseJL

One way to embed sparse vectors faster: use sparse matrices

[Kane, N., SODA 2012]

building on work of [Weinberger, Dasgupta, Langford, Smola, Attenberg, ICML, 2009], [Dasgupta, Kumar, Sarlós, STOC 2010]

Embedding time $O(m + s\|x\|_0) = O(m + \varepsilon m\|x\|_0)$



Each black cell is $\pm 1/\sqrt{s}$ at random

$m = O(\varepsilon^{-2} \log N), s = O(\varepsilon^{-1} \log N)$

Dimensionality reduction — SparseJL

Why does it work?

Dimensionality reduction — SparseJL

Why does it work?

Let's look at construction where each column has non-zeroes in exactly s random locations. Let $\delta_{r,i}$ be 1 if $A_{r,i} \neq 0$, and let $\sigma_{r,i}$ be a random sign so that $A_{r,i} = \delta_{r,i}\sigma_{r,i}$. Then,

- $(Ax)_r = \frac{1}{\sqrt{s}} \sum_{i=1}^{n} \delta_{r,i}\sigma_{r,i}x_i$

Dimensionality reduction — SparseJL

Why does it work?

Let's look at construction where each column has non-zeroes in exactly s random locations. Let $\delta_{r,i}$ be 1 if $A_{r,i} \neq 0$, and let $\sigma_{r,i}$ be a random sign so that $A_{r,i} = \delta_{r,i}\sigma_{r,i}$. Then,

- $(Ax)_r = \frac{1}{\sqrt{s}} \sum_{i=1}^{n} \delta_{r,i}\sigma_{r,i}x_i$
- $(Ax)_r^2 = \frac{1}{s} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_ix_j$

Dimensionality reduction — SparseJL

Why does it work?

Let's look at construction where each column has non-zeroes in exactly s random locations. Let $\delta_{r,i}$ be 1 if $A_{r,i} \neq 0$, and let $\sigma_{r,i}$ be a random sign so that $A_{r,i} = \delta_{r,i}\sigma_{r,i}$. Then,

- $(Ax)_r = \frac{1}{\sqrt{s}} \sum_{i=1}^{n} \delta_{r,i}\sigma_{r,i}x_i$
- $(Ax)_r^2 = \frac{1}{s} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
- $\|Ax\|_2^2 = \frac{1}{s} \sum_{r=1}^{n} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$

Dimensionality reduction — SparseJL

Why does it work?

Let's look at construction where each column has non-zeroes in exactly s random locations. Let $\delta_{r,i}$ be 1 if $A_{r,i} \neq 0$, and let $\sigma_{r,i}$ be a random sign so that $A_{r,i} = \delta_{r,i}\sigma_{r,i}$. Then,

- $(Ax)_r = \frac{1}{\sqrt{s}} \sum_{i=1}^{n} \delta_{r,i}\sigma_{r,i}x_i$
- $(Ax)_r^2 = \frac{1}{s} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{i\neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
- $\|Ax\|_2^2 = \frac{1}{s} \sum_{r=1}^{n} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{r=1}^{n} \sum_{i\neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
  $= \frac{1}{m} \sum_{i=1}^{n} x_i^2 \cdot (\sum_{r=1}^{m} \delta_{r,i}) + \frac{1}{s} \sum_{r=1}^{n} \sum_{i\neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$

Dimensionality reduction — SparseJL

Why does it work?

Let's look at construction where each column has non-zeroes in exactly s random locations. Let $\delta_{r,i}$ be 1 if $A_{r,i} \neq 0$, and let $\sigma_{r,i}$ be a random sign so that $A_{r,i} = \delta_{r,i}\sigma_{r,i}$. Then,

- $(Ax)_r = \frac{1}{\sqrt{s}} \sum_{i=1}^{n} \delta_{r,i}\sigma_{r,i}x_i$
- $(Ax)_r^2 = \frac{1}{s} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
- $\|Ax\|_2^2 = \frac{1}{s} \sum_{r=1}^{n} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
  $= \frac{1}{m} \sum_{i=1}^{n} x_i^2 \cdot (\sum_{r=1}^{m} \delta_{r,i}) + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
  $= \|x\|_2^2 + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$

Dimensionality reduction — SparseJL

Why does it work?

Let's look at construction where each column has non-zeroes in exactly s random locations. Let $\delta_{r,i}$ be 1 if $A_{r,i} \neq 0$, and let $\sigma_{r,i}$ be a random sign so that $A_{r,i} = \delta_{r,i}\sigma_{r,i}$. Then,

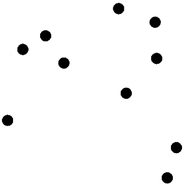
- $(Ax)_r = \frac{1}{\sqrt{s}} \sum_{i=1}^{n} \delta_{r,i}\sigma_{r,i}x_i$
- $(Ax)_r^2 = \frac{1}{s} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
- $\|Ax\|_2^2 = \frac{1}{s} \sum_{r=1}^{n} \sum_{i=1}^{n} \delta_{r,i}x_i^2 + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
  $= \frac{1}{m} \sum_{i=1}^{n} x_i^2 \cdot (\sum_{r=1}^{m} \delta_{r,i}) + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$
  $= \|x\|_2^2 + \frac{1}{s} \sum_{r=1}^{n} \sum_{i \neq j} \delta_{r,i}\delta_{r,j}\sigma_{r,i}\sigma_{r,j}x_i x_j$

The error term above is some quadratic form $\sigma^T B\sigma$. Can argue that it's small with high probability using known tail bounds for quadratic forms (the "Hanson-Wright inequality").

Streaming Data Mining

1. Items
    - Item frequencies
    - Distinct elements
    - Moment estimation
2. Vectors
    - Dimensionality reduction
    - **k-means**
    - Linear Regression
3. Matrices
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

k-means clustering



Input: $x_1, \ldots, x_N \in \mathbb{R}^n$, integer $k > 1$

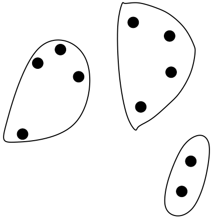
Output: A partition of input points into k clusters $C_1, \ldots, C_k$

Goal: Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \|x_i - \mu_r\|_2^2,$$

where μ_r is the centroid of C_r, i.e. $\mu_r = \frac{1}{|C_r|} \sum_{i \in C_r} x_i$.

k-means clustering



Input: $x_1, \ldots, x_N \in \mathbb{R}^n$, integer $k > 1$
Output: A partition of input points into k clusters $C_1, \ldots, C_k$
Goal: Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \|x_i - \mu_r\|_2^2,$$

where μ_r is the centroid of C_r, i.e. $\mu_r = \frac{1}{|C_r|} \sum_{i \in C_r} x_i$.

k-means clustering

Let's look at the objective function

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

Let's focus on a particular r with $|C_r| > 1$

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

Let's focus on a particular r with $|C_r| > 1$

- Look at each term in the inner sum as

$$\left\langle x_i - \frac{1}{|C_r|} \sum_{j \in C_r|} x_j , x_i - \frac{1}{|C_r} \sum_{j \in C_r} x_j \right\rangle$$

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

Let's focus on a particular r with $|C_r| > 1$

- Look at each term in the inner sum as

$$\left\langle x_i - \frac{1}{|C_r|} \sum_{j \in C_r|} x_j , x_i - \frac{1}{|C_r} \sum_{j \in C_r} x_j \right\rangle$$

- For each $i \in C_r$ the coefficient of $\|x_i\|_2^2$ in the inner sum is $1 - 2/|C_r| + |C_r| \cdot (1/|C_r|^2) = (1 - 1/|C_r|)$

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

Let's focus on a particular r with $|C_r| > 1$

- Look at each term in the inner sum as

$$\left\langle x_i - \frac{1}{|C_r|} \sum_{j \in C_r|} x_j , x_i - \frac{1}{|C_r} \sum_{j \in C_r} x_j \right\rangle$$

- For each $i \in C_r$ the coefficient of $\|x_i\|_2^2$ in the inner sum is $1 - 2/|C_r| + |C_r| \cdot (1/|C_r|^2) = (1 - 1/|C_r|)$
- For each $i < j \in C_r$ the coefficient of $\langle x_i, x_j \rangle$ is $-2/|C_r|$

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

Let's focus on a particular r with $|C_r| > 1$

- Look at each term in the inner sum as
$$\left\langle x_i - \frac{1}{|C_r|} \sum_{j \in C_r|} x_j , x_i - \frac{1}{|C_r} \sum_{j \in C_r} x_j \right\rangle$$
- For each $i \in C_r$ the coefficient of $\|x_i\|_2^2$ in the inner sum is $1 - 2/|C_r| + |C_r| \cdot (1/|C_r|^2) = (1 - 1/|C_r|)$
- For each $i < j \in C_r$ the coefficient of $\langle x_i, x_j \rangle$ is $-2/|C_r|$
- Noting that $\|x_i - x_j\|_2^2 = \|x_i\|_2^2 + \|x\|_2^2 - 2 \langle x_i, x_j \rangle$,
$$(*) \text{ equals } \sum_{r=1}^{k} \frac{1}{|C_r|} \cdot \left(\sum_{i<j \in C_r} \|x_i - x_j\|_2^2 \right)$$

k-means clustering

Let's look at the objective function

Minimize

$$\sum_{r=1}^{k} \sum_{i \in C_r} \left\| x_i - \frac{1}{|C_r|} \cdot \sum_{j \in C_r} x_j \right\|_2^2 (*)$$

Let's focus on a particular r with $|C_r| > 1$

- Look at each term in the inner sum as

$$\left\langle x_i - \frac{1}{|C_r|} \sum_{j \in C_r|} x_j, x_i - \frac{1}{|C_r} \sum_{j \in C_r} x_j \right\rangle$$

- For each $i \in C_r$ the coefficient of $\|x_i\|_2^2$ in the inner sum is $1 - 2/|C_r| + |C_r| \cdot (1/|C_r|^2) = (1 - 1/|C_r|)$
- For each $i < j \in C_r$ the coefficient of $\langle x_i, x_j \rangle$ is $-2/|C_r|$
- Noting that $\|x_i - x_j\|_2^2 = \|x_i\|_2^2 + \|x\|_2^2 - 2 \langle x_i, x_j \rangle$,

$$(*) \text{ equals } \sum_{r=1}^{k} \frac{1}{|C_r|} \cdot \left(\sum_{i<j \in C_r} \|x_i - x_j\|_2^2 \right)$$

$\Rightarrow$ JL with $O(\varepsilon^{-2} \log N)$ rows preserves optimality of k-means up to $1 \pm \varepsilon$

k-means clustering

Also see [Boutsidis, Zouzias, Mahoney, Drineas, arXiv abs/1110.2897]**, which shows that $O(k/\varepsilon^2)$ dimensions preserves optimal solution up to $2 \pm \varepsilon$**

k-means clustering

Also see [Boutsidis, Zouzias, Mahoney, Drineas, arXiv abs/1110.2897]**, which shows that** $O(k/\varepsilon^2)$ **dimensions preserves optimal solution up to** $2 \pm \varepsilon$

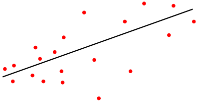
For papers on how to actually do fast k**-means clustering with provable guarantees, see**

- [Guha, Meyerson, Mishra, Motwani, O'Callaghan, IEEE Trans. Knowl. Data Eng. 15(3), 2003]
- [Har-Peled, Mazumdar, STOC 2004]
- [Ostrovsky, Rabani, Schulman, Swamy, FOCS 2006]
- [Arthur, Vasilvitskii, SODA 2007]
- [Aggarwal, Deshpande, Kannan, APPROX 2009]
- [Ailon, Jaiswal, Monteleoni, NIPS 2009]
- [Jaiswal, Garg, RANDOM 2012]
- . . .

Streaming Data Mining

1. Items
    - Item frequencies
    - Distinct elements
    - Moment estimation
2. Vectors
    - Dimensionality reduction
    - k-means
    - **Linear Regression**
3. Matrices
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

Linear regression

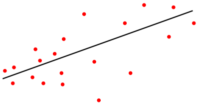


Want to find a linear function that best matches data, quadratic penalty

$$\min_x \|Sx - b\|_2 (*)$$

If rows of S are S_i, we want our linear function to have $f(S_i) = b_i$ for all i

Linear regression



Want to find a linear function that best matches data, quadratic penalty

$$\min_x \|Sx - b\|_2 (*)$$

If rows of S are S_i, we want our linear function to have $f(S_i) = b_i$ for all i

The JL connection:
$\|Sx - b\|_2^2 = \|Sx - b_{||} - b_\perp\|_2^2 = \|Sx - b_{||}\|_2^2 + \|b_\perp\|_2^2$, where $b_{||}$ is in the column span of S, and $b_\perp$ is in the orthogonal complement. So, $b_{||} = Sy_{||}$ for some $y_{||}$, so $Sx - b_{||} = S(x - y_{||})$. Thus if we apply some JL matrix A which satisfies $\|ASz\|_2 \approx \|Sz\|_2$ for *all* z, we preserve (*). Can get away with $m = O(d/\varepsilon^2)$ [Arora, Hazan, Kale, RANDOM 2006].

Linear regression

For more on fast linear regression, low rank approximation, and other numerical linear algebra problems:

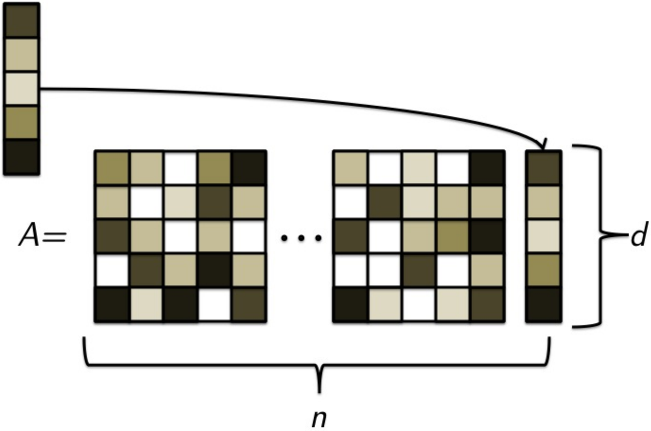
- [Sarlós, FOCS 2006]
- [Clarkson, Woodruff, STOC 2009]
- [Mahoney, *Randomized Algorithms for Matrices and Data*, 2011]
- [Halko, Martinsson, Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, 2009]
- [Boutsidis, Drineas, Magdon-Ismail, arXiv abs/1202.3505]
- [Clarkson, Woodruff, arXiv abs/1207.6365]
- . . .

Last paper listed can do linear regression on $d \times n$ matrices in $O(\mathrm{nnz}(S) + \mathrm{poly}(d/\varepsilon))$ time! ($nnz(S)$ is the number of non-zero entries in S)

Streaming Data Mining

1. Items
    - Item frequencies
    - Distinct elements
    - Moment estimation
2. Vectors
    - Dimensionality reduction
    - k-means
    - Linear Regression
3. Matrices
    - **Efficiently approximating the covariance matrix**
    - Sparsification by sampling

Streaming Matrices



A stream of vectors can be viewed a very large matrix A.

Streaming Matrices

What do we usually want to get from large matrices?

- Low rank approximation
- Singular Value Decomposition
- Principal Component Analysis
- Latent Dirichlet allocation
- Latent Semantic Indexing
- ...

For the above, it is sufficient to compute AA^T.

Streaming Matrices

Computing AA^T is trivial from the stream of columns A_i

$$AA^T = \sum_{i=1}^{n} A_i A_i^T$$

In words, AA^T is sum of outer products of the columns of A.

Streaming Matrices

Computing AA^T is trivial from the stream of columns A_i

$$AA^T = \sum_{i=1}^{n} A_i A_i^T$$

In words, AA^T is sum of outer products of the columns of A.

Naïve solution

Time $O(nd^2)$ and space $O(d^2)$.

Streaming Matrices

Computing AA^T is trivial from the stream of columns A_i

$$AA^T = \sum_{i=1}^{n} A_i A_i^T$$

In words, AA^T is sum of outer products of the columns of A.

Naïve solution

Time $O(nd^2)$ and space $O(d^2)$.

What is $d = 10,000$? or $100,000$?
Order of d^2 time or space is out of the question...

Matrix Column Sampling

We want to create a “sketch” B which approximates A.

1. B is as small as possible
2. Computing B is as efficient as possible
3. $AA^T \approx BB^T$

More accurately:

$$\|AA^T - BB^T\| \leq \varepsilon \|A\|_f^2$$

Amazingly enough, we can get this by sampling columns from A!

[Alan Frieze, Ravi Kannan, and Santosh Vempala, Fast monte-carlo algorithms for finding low-rank approximations, 1998]
[Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels, 2002]
[Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. 2003]
[Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis, 2007]
[Roberto Imbuzeiro Oliveira. Sums of random hermitian matrices and an inequality by Rudelson, 2010]
[Christos Boutsidis, Petros Drineas. and Malik Magdon-Ismail. Near optimal column-based matrix reconstruction, 2011]

Matrix Column Sampling

Let B contain ℓ **independently** chosen columns from A

$$B_j = A_i / \sqrt{\ell p_i} \quad \text{w.p.} \quad p_i = \|A_i\|_2^2 / \|A\|_f^2$$

To see why this makes sense, let us compute the expectation of BB^T

$$\mathbb{E}[B_j B_j^T] = \sum_{i=1}^{n} p_i \frac{A_i}{\sqrt{\ell p_i}} \frac{A_i^T}{\sqrt{\ell p_i}} = \frac{1}{\ell} \sum_{i=1}^{n} A_i A_i^T = \frac{1}{\ell} AA^T$$

So,

$$\mathbb{E}[BB^T] = \sum_{j=1}^{\ell} \mathbb{E}[B_j B_j^T] = AA^T$$

Note, $BB^T = \sum_{j=1}^{\ell} B_j B_j^T$ is a sum of **independent** random variables...

Matrix Column Sampling

Figuring out the minimal value for ℓ is non trivial.
But, it is enough to require

$$\ell = c \log(r)/r\varepsilon^2 \ .$$

- $r = \|A\|_f^2 / \|A\|_2^2$ is the numeric rank of A.
- $1 \leq r \leq Rank(A) \leq d$.
- $r \approx \mathit{const}$ for “Interesting” matrices.

Matrix Column Sampling

Figuring out the minimal value for ℓ is non trivial.
But, it is enough to require

$$\ell = c \log(r)/r\varepsilon^2 \ .$$

- $r = \|A\|_f^2 / \|A\|_2^2$ is the numeric rank of A.
- $1 \leq r \leq Rank(A) \leq d$.
- $r \approx const$ for “Interesting” matrices.

Matrix Column Sampling

We can compute B in time $O(dn)$ (same as reading the matrix).
B requires at most $O(d/\varepsilon^2)$ space and:

$$\|AA^T - BB^T\| \leq \varepsilon \|A\|_f^2$$

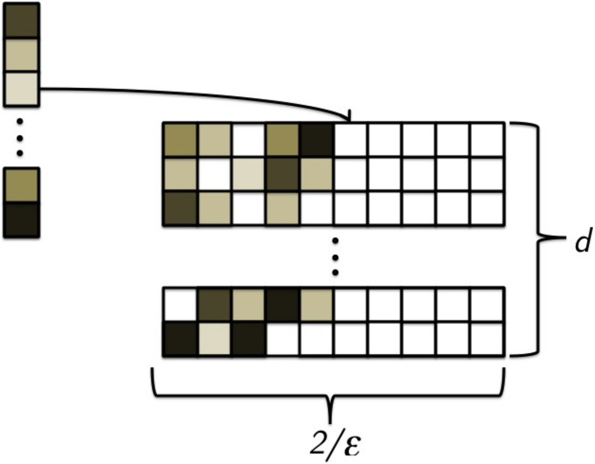
Matrix Column Sampling

A streaming procedure for matrix column sampling.

```
Input: $\varepsilon \in (0,1]$,  $A \in \mathbb{R}^{d \times n}$
$\ell \leftarrow \lceil c/\varepsilon^2 \rceil$
$B \leftarrow$ all zeros matrix $\in \mathbb{R}^{d \times \ell}$
$\mathrm{T} = 0$
for $i \in [n]$ do
    $t = \|A_i\|^2$
    $T \leftarrow T + t$
    for $j \in [\ell]$ do
        w.p. $t/T$
            $B_j \leftarrow \frac{1}{\sqrt{\ell t/T}} A_i$
Return: $B$
```

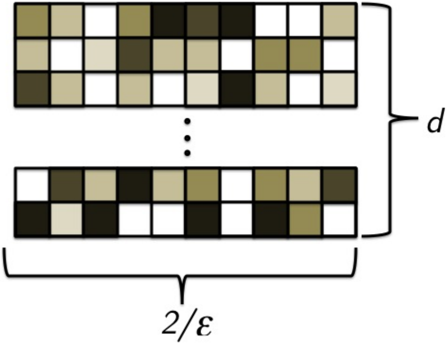
The dependency on $1/\varepsilon^2$ is problematic for small ε.

Matrix Sketching



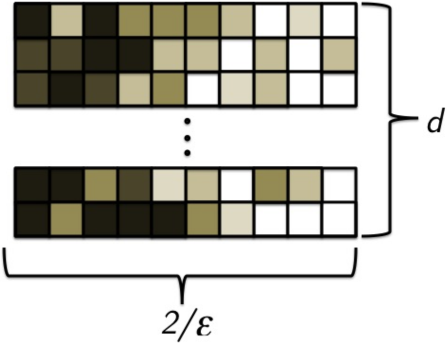
The space requirement can be reduced to $O(d/\varepsilon)$

Matrix Sketching



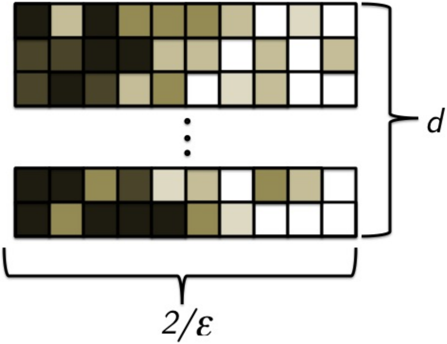
Columns are added until the sketch is 'full'

Matrix Sketching



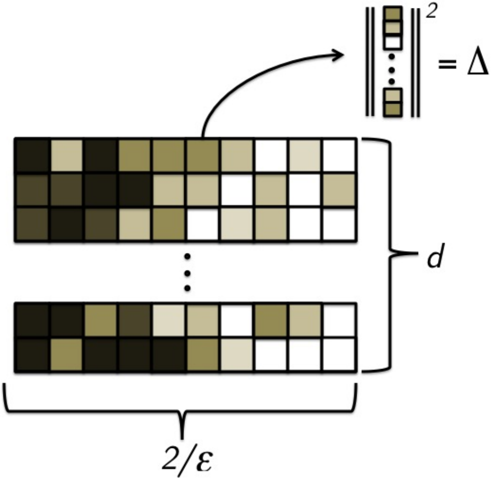
Then, we compute the SVD of the sketch and rotate it

Matrix Sketching



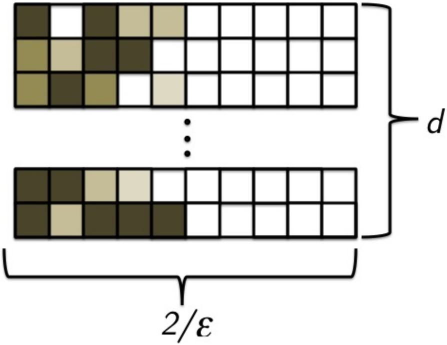
$B = USV$ is rotated to $B_{new} = US$ (note that $BB^T = B_{new}B_{new}^T$)

Matrix Sketching



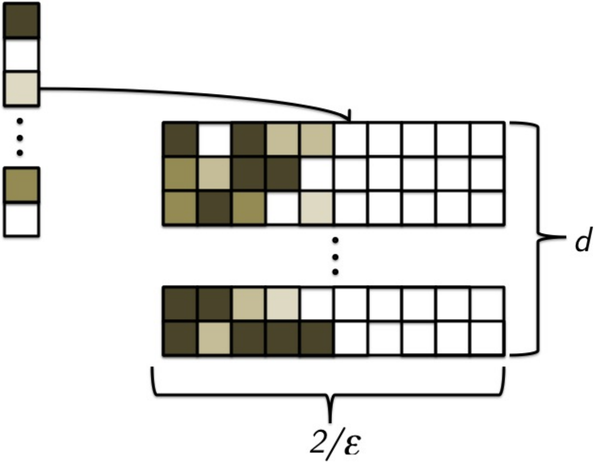
Let $\Delta = \|B_{1/\varepsilon}\|$

Matrix Sketching



Shrink all the columns such that their ℓ_2^2 is reduced by Δ

Matrix Sketching



Start aggregating columns again...

Matrix Sketching

```
**Input:** $\varepsilon \in (0, 1],\ \ A \in \mathbb{R}^{n \times m}$
$\ell \leftarrow \lceil 1/\varepsilon \rceil$
$B \leftarrow$ all zeros matrix $\in \mathbb{R}^{\ell \times m}$
**for** $i \in [n]$ **do**
    $B_\ell \leftarrow A_i$
    $[U, \Sigma, V] \leftarrow SVD(B)$
    $\Delta \leftarrow \Sigma^2_{c\ell, c\ell}$
    $\bar{\Sigma} \leftarrow \sqrt{\max(\Sigma^2 - I_\ell \Delta, 0)}$
    $B \leftarrow \bar{\Sigma} V$
**Return:** $B$
```

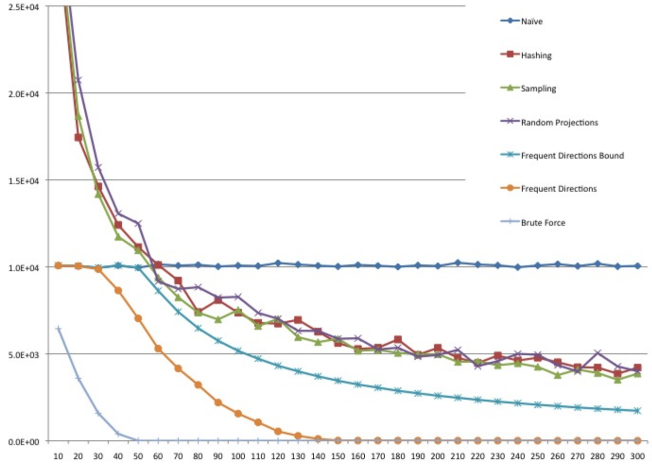
Liberty, 2012

Matrix Sketching

We can compute B in $O(dn/\varepsilon)$ time and $O(d/\varepsilon)$ space.

$$\|AA^T - BB^T\| \leq \varepsilon \|A\|_f^2$$

Matrix Sketching

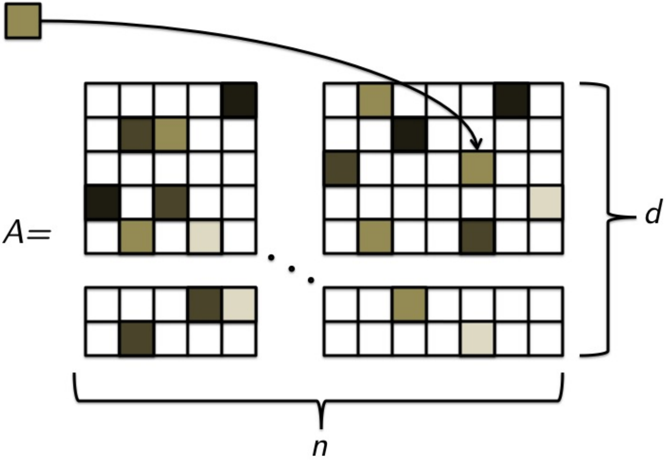


Example approximation of a synthetic matrix of size $1,000 \times 10,000$

Streaming Data Mining

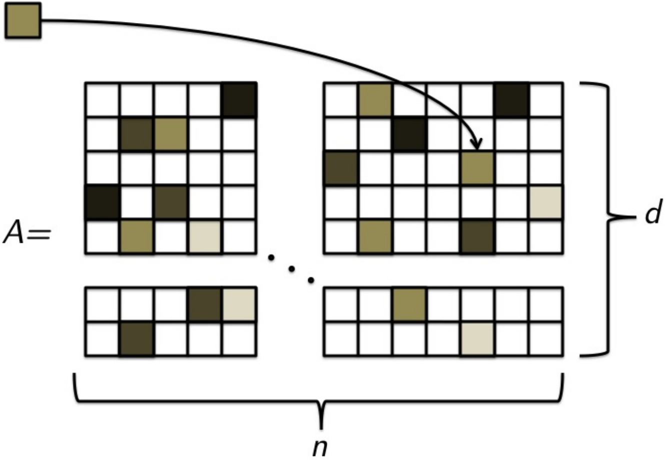
1. Items
    - Item frequencies
    - Distinct elements
    - Moment estimation
2. Vectors
    - Dimensionality reduction
    - k-means
    - Linear Regression
3. Matrices
    - Efficiently approximating the covariance matrix
    - **Sparsification by sampling**

Sparsification by sampling



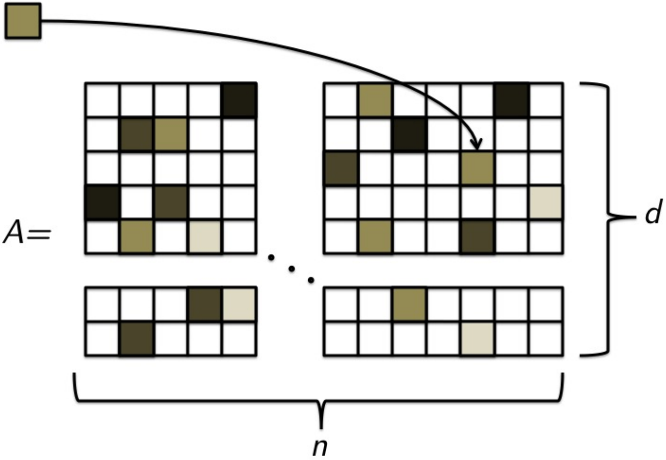
Sometimes, we only get one matrix entry at a time...

Sparsification by sampling



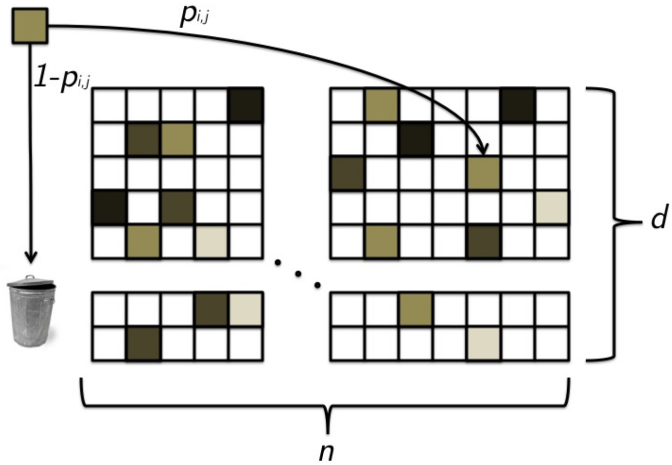
Recommender system example: user i rated item j with $A_{i,j}$-stars.

Sparsification by sampling



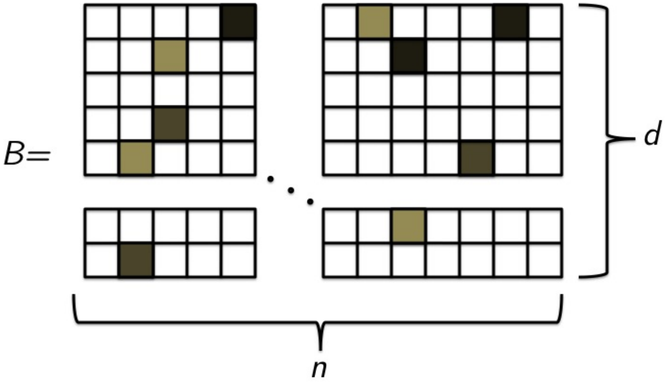
Note that these matrices are usually very sparse!

Sparsification by sampling



If we sample entries we can only improve sparsity.

Sparsification by sampling



How close is B the original matrix A, $\|A-B\|_2 \leq ?$

Sparsification by sampling

Generic sampling algorithm:

```
$B \leftarrow$ all zeros matrix
for $(i, j, A_{i,j})$ do
  w.p. $p_{i,j}$
      $B_{i,j} \leftarrow A_{i,j}/p_{i,j}$
Return: $B$
```

For any choice of $p_{i,j}$ we have $\mathbb{E}[A - B] = 0$ or:

$$\mathbb{E}[B] = A$$

Sparsification by sampling

Generic sampling algorithm:

```
$B \leftarrow$ all zeros matrix
for $(i, j, A_{i,j})$ do
    w.p. $p_{i,j}$
        $B_{i,j} \leftarrow A_{i,j}/p_{i,j}$
Return: $B$
```

For any choice of $p_{i,j}$ we have $\mathbb{E}[A - B] = 0$ or:

$$\mathbb{E}[B] = A$$

What about $\text{Var}[A - B]$? In other words, when is $\|A - B\|_2$ small?

Sparsification by sampling

Assume w.l.o.g. that $|A_{i,j}| \leq 1$ we can get that:

$$\|A - B\|_2 \leq \varepsilon$$

[Fast Computation of Low Rank Matrix Approximations, Achlioptas, McSherry]

$$p_{i,j} = \tilde{O}(1) \cdot \max(A_{i,j}^2/n, |A_{i,j}|/\sqrt{n})$$

- $|B| = \tilde{O}(n + \frac{n}{\varepsilon^2} \sum_{i,j} A_{i,j}^2)$

[A Fast Random Sampling Algorithm for Sparsifying Matrices, Arora, Hazan, Kale]

$$p_{i,j} = \min(|A_{i,j}|\sqrt{n/\varepsilon}, 1)$$

- $|B| = \tilde{O}(\frac{\sqrt{n}}{\varepsilon} \sum_{i,j} |A_{i,j}|)$

The latter has a very simple proof, see paper for more details.

Streaming Data Mining

1. Items (words, IP-adresses, events, clicks,...):
    - Item frequencies
    - Distinct elements
    - Moment estimation
2. Vectors (text documents, images, example features,...)
    - Dimensionality reduction
    - k-means
    - Linear Regression
3. Matrices (text corpora, user preferences, social graphs,...)
    - Efficiently approximating the covariance matrix
    - Sparsification by sampling

Thank you!

We would like to thank: Nir Ailon, Moses S. Charikar, Sudipto Guha, Elad Hazan, Yishay Mansour, Muthu Muthukrishnan, David P. Woodruff, Petros Drineas, Piotr Indyk, Graham Cormode, Andrew McGregor, and Dimitris Achlioptas for pointing out important algorithms and references.