Stream Algorithmics

Albert Bifet (@abifet)

Paris, 25 November 2016

albert.bifet@telecom-paristech.fr

Data Streams

Big Data & Real Time

Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Big Data & Real Time

Data Stream Algorithmics

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Big Data & Real Time

Data Stream Algorithmics

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \dots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Use a n-bit vector to memorize all the numbers ($O(n)$ space)

Big Data & Real Time

Data Stream Algorithmics

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Data Streams:
$O(\log(n))$ space.

Big Data & Real Time

Data Stream Algorithmics

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Data Streams: $O(\log(n))$ space. Store

$$\frac{n(n+1)}{2} - \sum_{j \leq i} \pi_{-1}[j].$$

Big Data & Real Time

Data Streams

Approximation algorithms

- Small error rate with high probability
- An algorithm $(\epsilon, \delta)-$approximates F if it outputs $\tilde{F}$ for which $\Pr[|\tilde{F} - F| > \epsilon F] < \delta$.

Big Data & Real Time

Data Stream Algorithmics

Examples

1. Compute different number of pairs of IP addresses seen in a router
2. Compute top-k most used words in tweets

Two problems: find number of distinct items and find most frequent items.

8 Bits Counter

1	0	1	0	1	0	1	0

What is the largest number we can store in 8 bits?

8 Bits Counter

Programming Techniques — S.L. Graham, R.L. Rivest, Editors

Counting Large Numbers of Events in Small Registers

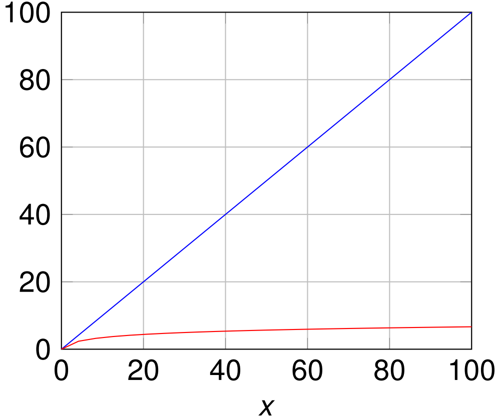
Robert Morris
Bell Laboratories, Murray Hill, N.J.

It is possible to use a small counter to keep approximate counts of large numbers. The resulting expected error can be rather precisely controlled. An example is given in which 8-bit counters (bytes) are used to keep track of as many as 130,000 events with a relative error which is substantially independent of the number n of events. This relative error can be expected to be 24 percent or less 95 percent of the time (i.e. $\sigma = n/8$). The techniques could be used to advantage in multichannel counting hardware or software used for the monitoring of experiments or processes.

What is the largest number we can store in 8 bits?

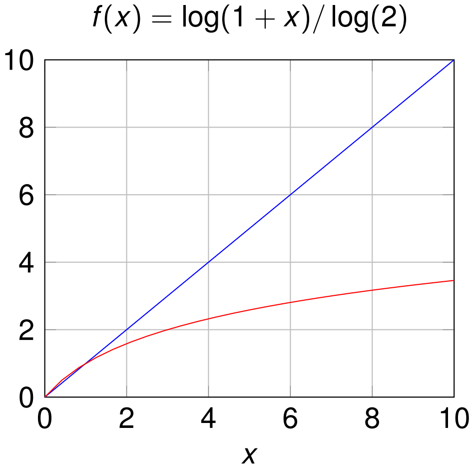
8 Bits Counter

$$f(x) = \log(1 + x)/\log(2)$$



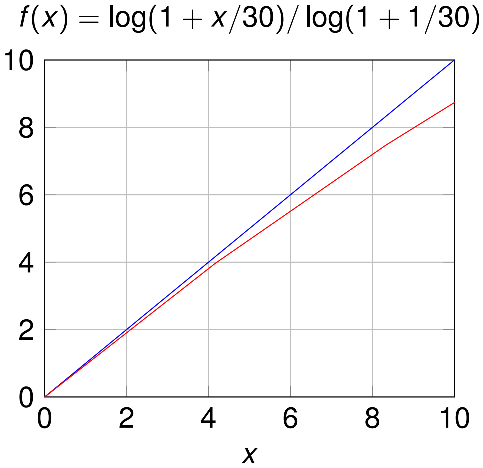
$f(0) = 0, f(1) = 1$

8 Bits Counter



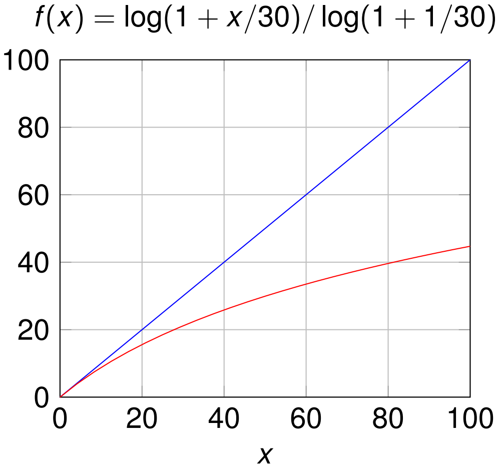
$f(0) = 0, f(1) = 1$

8 Bits Counter



$f(0) = 0, f(1) = 1$

8 Bits Counter



$f(0) = 0, f(1) = 1$

8 bits Counter

MORRIS APPROXIMATE COUNTING ALGORITHM

```
Init counter c ← 0
for every event in the stream
        do rand = random number between 0 and 1
            if rand < p
                then c ← c + 1
```

What is the largest number we can store in 8 bits?

8 bits Counter

MORRIS APPROXIMATE COUNTING ALGORITHM

```
Init counter c ← 0
for every event in the stream
        do rand = random number between 0 and 1
            if rand < p
              then c ← c + 1
```

With $p = 1/2$ we can store 2×256
with standard deviation $\sigma = \sqrt{n}/2$

8 bits Counter

MORRIS APPROXIMATE COUNTING ALGORITHM

```
Init counter c ← 0
for every event in the stream
    do rand = random number between 0 and 1
        if rand < p
            then c ← c + 1
```

With $p = 2^{-c}$ then $E[2^c] = n + 2$ with variance $\sigma^2 = n(n+1)/2$

8 bits Counter

MORRIS APPROXIMATE COUNTING ALGORITHM

Init counter $c \leftarrow 0$
for every event in the stream
  do *rand* = random number between 0 and 1
    if $rand < p$
      then $c \leftarrow c + 1$

$$\text{If } p = b^{-c} \text{ then } E[b^c] = n(b-1) + b,$$
$$\sigma^2 = (b-1)n(n+1)/2$$

Data Stream Algorithmics

Examples

1. **Compute different number of pairs of IP addresses seen in a router**
   IPv4: 32 bits
   IPv6: 128 bits
2. Compute top-k most used words in tweets

Find number of distinct items

Data Stream Algorithmics

Memory unit	Size	Binary size
kilobyte (kB/KB)	10^3	2^{10}
megabyte (MB)	10^6	2^{20}
gigabyte (GB)	10^9	2^{30}
terabyte (TB)	10^{12}	2^{40}
petabyte (PB)	10^{15}	2^{50}
exabyte (EB)	10^{18}	2^{60}
zettabyte (ZB)	10^{21}	2^{70}
yottabyte (YB)	10^{24}	2^{80}

Find number of distinct items

IPv4: 32 bits IPv6: 128 bits

Data Stream Algorithmics

Example

1. **Compute different number of pairs of IP addresses seen in a router**
   IPv4: 32 bits, IPv6: 128 bits
   Using 256 words of 32 bits accuracy of 5%

Find number of distinct items

Data Stream Algorithmics

Example

1. **Compute different number of pairs of IP addresses seen in a router**

   Selecting n random numbers,

   - half of these numbers have the first bit as zero,
   - a quarter have the first and second bit as zero,
   - an eigth have the first, second and third bit as zero..

   A pattern 0^i1 appears with probability $2^{-(i+1)}$, so $n \approx 2^{i+1}$

Find number of distinct items

Data Stream Algorithmics

FLAJOLET-MARTIN PROBABILISTIC COUNTING ALGORITHM

Init $bitmap[0 \ldots L-1] \leftarrow 0$
for every item x in the stream
    do $index = \rho(hash(x))$ ▷ position of the least significant 1-bit
        if $bitmap[index] = 0$
            then $bitmap[index] = 1$
$b \leftarrow$ position of leftmost zero in bitmap
return $2^b/0.77351$

$$E[pos] \approx \log_2 \phi n \approx \log_2 0.77351 \cdot n$$
$$\sigma(pos) \approx 1.12$$

Data Stream Algorithmics

item x	$hash(x)$	$\rho(hash(x))$	bitmap
a	0110	1	01000
b	1001	0	11000
c	0111	1	11000
d	1100	0	11000
a			
b			
e	0101	1	11000
f	1010	0	11000
a			
b			

$$b = 2, n \approx 2^2/0.77351 = 5.17$$

Data Stream Algorithmics

Flajolet-Martin Probabilistic Counting Algorithm

Init $bitmap[0 \dots L-1] \leftarrow 0$
for every item x in the stream
  do $index = \rho(hash(x))$ $\triangleright$ position of the least significant 1-bit
    if $bitmap[index] = 0$
      then $bitmap[index] = 1$
$b \leftarrow$ position of leftmost zero in bitmap
return $2^b/0.77351$

Init $M \leftarrow -\infty$
for every item x in the stream
  do $M = max(M, \rho(h(x))$
$b \leftarrow M + 1$ $\triangleright$ position of leftmost zero in bitmap
return $2^b/0.77351$

Data Stream Algorithmics

Stochastic Averaging

Perform m experiments in parallel

$$\sigma' = \sigma/\sqrt{m}$$

Relative accuracy is $0.78/\sqrt{m}$

HyperLogLog Counter

- the stream is divided in $m = 2^b$ substreams
- the estimation uses harmonic mean
- Relative accuracy is $1.04/\sqrt{m}$

Data Stream Algorithmics

HYPERLOGLOG COUNTER

```
Init M[0 ... b − 1] ← −∞
for every item x in the stream
      do index = h_b(x)
         M[index] = max(M[index], ρ(h^b(x)))
return α_m m^2 / Σ_{j=0}^{m−1} 2^{−M[j]}
```

$$h(x) = 010011000111$$
$$h_3(x) = 001 \text{ and } h^3(x) = 011000111$$

Methodology

Paolo Boldi
Facebook Four degrees of separation

Big Data does not need big machines,
it needs big **intelligence**

Data Stream Algorithmics

Examples

1. Compute different number of pairs of IP addresses seen in a router
2. **Compute top-k most used words in tweets**

Find most frequent items

Data Stream Algorithmics

MAJORITY

```
Init counter c ← 0
for every item s in the stream
    do if counter is zero
         then pick up the item
       if item is the same
         then increment counter
         else decrement counter
```

Find the item that it is contained in more than half of the instances

Data Stream Algorithmics

FREQUENT

```
for every item i in the stream
    do if item i is not monitored
          do if < k items monitored
                then add a new item with count 1
                else if an item z whose count is zero exists
                        then replace this item z by the new one
                        else decrement all counters by one
       else ▷ item i is monitored
            increase its counter by one
```

Figure: Algorithm FREQUENT to find most frequent items

Data Stream Algorithmics

LossyCounting

```
for every item i in the stream
    do if item i is not monitored
          then add a new item with count 1 + Δ
          else ▷ item i is monitored
               increase its counter by one
       if ⌊n/k⌋ ≠ Δ
          then Δ = ⌊n/k⌋
               decrement all counters by one
               remove items with zero counts
```

Figure: Algorithm LossyCounting to find most frequent items

Data Stream Algorithmics

SPACE SAVING

```
for every item i in the stream
    do if item i is not monitored
        do if < k items monitored
            then add a new item with count 1
            else replace the item with lower counter
                 increase its counter by one
        else ▷ item i is monitored
             increase its counter by one
```

Figure: Algorithm SPACE SAVING to find most frequent items

Data Stream Algorithmics

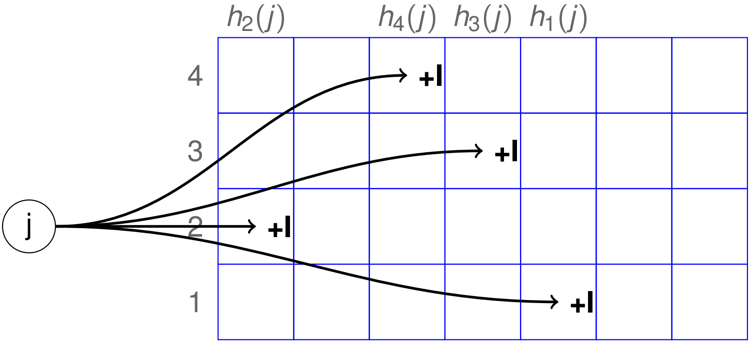


Figure: A CM sketch structure example of $\epsilon = 0.4$ and $\delta = 0.02$

Count-Min Sketch

A two dimensional array with width w and depth d

$$w = \left\lceil \frac{e}{\epsilon} \right\rceil, \qquad d = \left\lceil \ln \frac{1}{\delta} \right\rceil$$

It uses space wd with update time d

CM-Sketch computes frequency data adding and removing real values.

Count-Min Sketch

A two dimensional array with width w and depth d

$$w = \left\lceil \frac{e}{\epsilon} \right\rceil, \qquad d = \left\lceil \ln \frac{1}{\delta} \right\rceil$$

It uses space $wd = \frac{e}{\epsilon} \ln \frac{1}{\delta}$ with update time $d = \ln \frac{1}{\delta}$

CM-Sketch computes frequency data adding and removing real values.

Data Stream Algorithmics

Problem

Given a data stream, choose k items with the same probability, storing only k elements in memory.

Reservoir Sampling

Data Stream Algorithmics

RESERVOIR SAMPLING

for every item i in the first k items of the stream
  do store item i in the reservoir
$n = k$
for every item i in the stream after the first k items of the stream
  do select a random number r between 1 and n
    if $r < k$
      then replace item r in the reservoir with item i
    $n = n + 1$

Figure: Algorithm RESERVOIR SAMPLING

Mean and Variance

Given a stream $x_1, x_2, \ldots, x_n$

$$\bar{x}_n = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

$$\sigma_n^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x}_i)^2.$$

Mean and Variance

Given a stream $x_1, x_2, \ldots, x_n$

$$s_n = \sum_{i=1}^{n} x_i, \;\; q_n = \sum_{i=1}^{n} x_i^2$$

$$s_n = s_{n-1} + x_n, \;\; q_n = q_{n-1} + x_n^2$$

$$\bar{x}_n = s_n / n$$

$$\sigma_n^2 = \frac{1}{n-1} \cdot (\sum_{i=1}^{n} x_i^2 - n\bar{x}_i^2) = \frac{1}{n-1} \cdot (q_n - s_n^2/n)$$

Data Stream Sliding Window

1011000111 $\boxed{1010101}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ϵ is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Stream Sliding Window

10110001111 $\boxed{0101011}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ϵ is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Stream Sliding Window

101100011110 $\boxed{1010111}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ϵ is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Stream Sliding Window

1011000111101 $\boxed{0101110}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ϵ is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Stream Sliding Window

$$10110001111010\ \boxed{1011101}$$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ϵ is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Stream Sliding Window

1011000111101 $\boxed{0111010}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\epsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ϵ is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Exponential Histograms

$M = 2$

	1010101	101	11	1	1	1
Content:	4	2	2	1	1	1
Capacity:	7	3	2	1	1	1

	1010101	101	11	11	1
Content:	4	2	2	2	1
Capacity:	7	3	2	2	1

	1010101	10111	11	1
Content:	4	4	2	1
Capacity:	7	5	2	1

Exponential Histograms

	1010101	101	11	1	1
Content:	4	2	2	1	1
Capacity:	7	3	2	1	1

Error $<$ content of the last bucket W/M

$\epsilon = 1/(2M)$ and $M = 1/(2\epsilon)$

$M \cdot \log(W/M)$ buckets to maintain the data stream sliding window

Exponential Histograms

1010101	101	11	1	1

Content:	4	2	2	1	1
Capacity:	7	3	2	1	1

To give answers in $O(1)$ time,
it maintain three counters LAST, TOTAL and VARIANCE.

$M \cdot \log(W/M)$ buckets to maintain the data stream sliding window