Data Streams & Sketches

Everything Data
CompSci 290.01 Spring 2014
Announcements (Thu. Apr 3)

• **Homework #12** will be posted by noon tomorrow.
This Class

- Key ideas
  - Read the data \emph{exactly once}.
  - Maintain a small \emph{sketch} as you read the data.
  - Answer queries \emph{approximately} using only the sketch.
Motivating Example 0

• How to compute the average number of hashtags per tweet without looking at all the data?

• Answer: Sampling
Follow up question

• How to sample n records without looking at all the N records in the data?

• Answer: Reservoir Sampling
Motivating Example 1

• How to build a URL shortening service?

• Answer:
  – Maintain a table from short URL to original URL.
  – When a new original URL arrives,
    (i) Compute a short URL
    (ii) Check that the short URL has not been used.
Follow up question

• How to quickly check whether a new item $i$ is in large set $S$ of items?

• Answer: *Bloom Filter*
Motivating Example 2

• How to find the number of unique IP addresses that visit a website?

• Answer:
  – Maintain a set of IP addresses
  – But this set will become very large very soon!
  – FM sketch
Outline

• Sampling
  – Reservoir Sampling

• Filtering Sketch
  – Bloom Filter

• Distinct elements Sketch
  – FM Sketch
Reservoir Sampling

Highlights:

• Make one pass over the data
• Maintain a reservoir of $n$ records.
• After reading $t$ rows, the reservoir is a simple random sample of the first $t$ rows.
Reservoir Sampling

Simple Algorithm

• Initialize reservoir to the first n rows.

• For the \((t+1)\text{st}\) row \(R\),
  
  – Pick a random number \(m\) between 1 and \(t+1\)

  – If \(m \leq n\), then replace the \(m\text{th}\) row in the reservoir with \(R\)
Proof ... that we indeed get a random sample

- If $N = n$, then $P[\text{row is in sample}] = 1$. Hence, reservoir contains all the rows in the table.

- Suppose for $N = t$, the reservoir is a simple random sample.
i.e., each row has $n/t$ chance of appearing in the sample.

- For $N = t+1$:
  - $(t+1)\text{st}$ row is included with probability $n/(t+1)$
  - Any other row:
    $P[\text{row is in reservoir}] = P[\text{row is in reservoir after } t \text{ steps}] * P[\text{row is not replaced in } (t+1)\text{th step}]$
    $= n/t * (1-1/(t+1)) = n/(t+1)$
Reservoir Sampling

Simple Algorithm

• Initialize reservoir to the first n rows.

- Construct the sample as you read the data. But we are still reading all the rows in the table to construct this sample!

  - If \( m \leq n \), then replace the \( m^{th} \) row in the reservoir with \( R \)
Reservoir Sampling

• Rather than looking at each element and deciding whether to include it or not,

Directly choose which element will be added to the reservoir next.
Suppose row $t$ from the original data was added to the reservoir.

Let $S(n,t) = s$ be the number of rows after row $t$ that were not included into the reservoir.
Skipping

• Idea: Sample $s$, and add row $(t+s+1)$ into the reservoir.

• CDF: $\Pr[S(n,t) \leq s]$
  
  $= 1 - \frac{t}{t+s+1}\frac{t-1}{t+s}\frac{t-2}{t+s-1} \ldots \frac{t-n+1}{t+s-n+2}$
Reservoir Sampling with Skipping

• Initialize reservoir with first n rows.

• After seeing t rows, randomly sample a skip s = S(n,t) from the CDF

• Pick a number m between 1 and n

• Replace the mth row in the reservoir with the (t+s+1)st row.

• Set t = t + s + 1
Reservoir Sampling with Skipping

- Initialize reservoir with first \( n \) rows.

- After seeing \( t \) rows, randomly sample a skip \( s = S(n, t) \) from the CDF.

\[
\text{Looks at exactly } O(n(1 + \ln(N/n))) \text{ rows}
\]

- Pick a number \( m \) between 1 and \( n \).

- Replace the \( m \)-th row in the reservoir with the \((t+s+1)\)-st row.

- Set \( t = t + s + 1 \).
Summary of Reservoir Sampling

• Helps create a simple random sample of \( n \) rows without looking at all the rows in the dataset

• Very useful when looking at all rows is expensive
  – E.g., rate limited data api like twitter.
Outline

• Sampling
  – Reservoir Sampling

• Filtering Sketch
  – Bloom Filter

• Distinct elements Sketch
  – FM Sketch
Problem

• Given a large set $S$, check whether a new element $i$ is in the set.

• Let $S$ have $m$ elements
• Let memory size $= n$ bits
  – In many cases $m > n$
Approximate Filter

• Returns TRUE with probability 1, when element is in S

• Returns FALSE with high probability \((1-\delta)\), when element is not in S
Primitive: Hash Function

- $h: S \rightarrow \{1,2,3,4,...,n\}$

- Elements of $S$ are hashed uniformly at random to one of the values in 1 to $n$.

- $\Pr[h(x) = y] = 1/n$
Bloom Filter

Consider a set of hash functions \{h_1, h_2, ..., h_k\}, h_i: S \rightarrow [1, n]

Initialization:
• Set all \( n \) bits in the memory to 0.

Insert a new element ‘a’:
• Compute \( h_1(a), h_2(a), ..., h_k(a) \). Set the corresponding bits to 1.

Check whether an element ‘a’ is in S:
• Compute \( h_1(a), h_2(a), ..., h_k(a) \). If all the bits are 1, return TRUE. Else, return FALSE.
Analysis … Part I

If $a$ is in $S$:

• If $h_1(a), h_2(a), \ldots, h_k(a)$ are all set to 1.
• Therefore, Bloom filter returns TRUE with probability 1.
Analysis … Part II

If a not in S:
• Bloom filter returns TRUE if each hi(a) is 1 due to some other element

\[
\Pr[\text{bit } j \text{ is } 1 \text{ after } m \text{ insertions}] = 1 - \Pr[\text{bit } j \text{ is } 0 \text{ after } m \text{ insertions}]
\]
\[
= 1 - \Pr[\text{bit } j \text{ was not set by } k \times m \text{ hash functions}]
\]
\[
= 1 - (1 - 1/n)^{km}
\]
If a not in S:

- Bloom filter returns TRUE if each hi(a) is 1 due to some other element

\[ \Pr[\text{Bloom filter returns TRUE}] = \Pr[\text{all bits } j \text{ are 1 after } m \text{ insertions}] = \{1 - (1 - 1/n)^{km}\}^k \]

\[ \approx (1 - e^{-km/n})^k \]
Example

- Suppose there are $m = 10^9$ elements in the set.
- Suppose memory size of 1 GB ($8 \times 10^9$ bits)

$k = 1$
- $Pr[\text{Bloom filter returns TRUE} \mid \text{a not in S}]$
  
  $= 1 - e^{-m/n}$
  
  $= 1 - e^{-1/8} \approx 0.1175$

$k = 2$
- $Pr[\text{Bloom filter returns TRUE} \mid \text{a not in S}]$
  
  $= (1 - e^{-2m/n})^2$
  
  $= (1 - e^{-1/4})^2 \approx 0.0493$
Example

- Suppose there are \( m = 10^9 \) elements in the set.
- Suppose memory size of 1 GB (8 x 10^9 bits)

As the number of hashes increases, it is more likely that all become 1 just by random chance.
Summary Bloom Filters

• Given a large set of elements $S$, efficiently check whether a new element is in the set.

• Bloom filters use hash functions to check membership
  – If $a$ is in $S$, return TRUE with probability 1
  – If $a$ is not in $S$, return FALSE with high probability
  – False positive error depends on $|S|$, number of bits in the memory and number of hash functions
Outline

• Sampling
  – Reservoir Sampling

• Filtering Sketch
  – Bloom Filter

• Distinct elements Sketch
  – FM Sketch
Distinct Elements

INPUT:
• A stream $S$ of elements from a domain $D$
  – A stream of logins to a website
  – A stream of URLs browsed by a user
• Memory with $n$ bits

OUTPUT
• An estimate of the number of distinct elements in the stream
  – Number of distinct users logging in to the website
  – Number of distinct URLs browsed by the user
FM Sketch

• Define: $\text{Tail}_0(h(x)) = \text{number of trailing consecutive 0's}$
  
  - $\text{Tail}_0(101001) = 0$
  - $\text{Tail}_0(101010) = 1$
  - $\text{Tail}_0(001100) = 2$
  - $\text{Tail}_0(101000) = 3$
  - $\text{Tail}_0(000000) = 6 \ (= L)$
FM Sketch

• For all $x \in S$,
  – Compute $k(x) = \text{Tail}_0(h(x))$

• Let $K = \max_{x \in S} k(x)$

• Return $F' = 2^K$
How good is the result?

• Let $F$ be the true number of distinct elements.

• For all $c > 3$,

$$\Pr\left[\frac{F}{c} \leq F' \leq cF\right] > 1 - \frac{3}{c}$$
Why?

- \( \Pr[\text{last } k \text{ bits are } 0] = 2^{-k}. \)

- \( 2^x \) and \( 2^y \) are powers of 2 closest to \( F/c, \ cF. \)
Why?

• \( \Pr[\text{at least one hash has } \geq y \text{ trailing 0s}] \leq \frac{F}{2^y} \leq \frac{1}{c} \)

• \( \Pr[\text{all hashes have } < x \text{ trailing 0s}] \leq \frac{2^x}{F} \leq \frac{2}{c} \)
Improving the result

• “Median of means”

• Compute a x b FM sketches:
  – F11, F12, …, F1a
  F21, F22, …, F2a
  …
  Fb1, Fb2, …, Fba
Improving the result

• “Median of means”

• Compute a x b FM sketches:

• Compute the mean of each set of a sketches
  – $G_i = \text{mean}(F_{i1}, F_{i2}, \ldots, F_{ia})$
Improving the result

• “Median of means”

• Compute a x b FM sketches:

• Compute the mean of each set of a sketches

• Return the median of these means as the answer
“Median of means”

- Compute $a \times b$ FM sketches
- Compute the mean of each set of $a$ sketches
- Return the median of these means as the answer

- The output now is very very close to the original count $F$
  - For sufficiently large $a$ and $b$. 
Summary of FM Sketch

• Computing the number of distinct exactly take time proportional to the size of the data, and needs maintaining a set of cardinality equal to the number of distinct elements.

• FM sketch allows approximate computation with much smaller space
  – Logarithmic in the number of distinct elements.