The Weighted Cardinality Estimation Problem

Aviv Yehezkel

Ph.D. Seminar

Supervisors: Prof. Reuven Cohen and Dr. Liran Katzir
Overview

• Cardinality Estimation Problem
• Weighted Cardinality Estimation Problem
• The Unified Scheme
Cardinality Estimation Problem
Motivation

Given a very long stream of elements with repetitions,

How many are distinct?
Cardinality of a Stream

• Let $M$ be a stream of elements with repetitions

  • $N$ is the number of elements called the size
  
  • $n$ the number of distinct elements called cardinality

• The problem:
  
  compute the cardinality $n$ in one pass and with small fixed memory

<table>
<thead>
<tr>
<th>Element</th>
<th>Multi</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>Z</td>
<td>1</td>
</tr>
</tbody>
</table>

$N = 8$

$n = 4$
Many Applications

Traffic analysis

Attacks detection

Genetics

Linguistic

and more…
Exact Solution

- Maintain distinct elements already seen

- One pass, but memory in order of $n$

- Lower bound: $\Omega(n)$ memory needed
Probabilistic Solution

• Main idea:
  • relax the constraint of exact value of the cardinality
  • An estimate with good precision is sufficient for the applications

• Several algorithms:
  • Probabilistic counting
  • HyperLogLog
  • Linear Counting
  • Min Count
  • …. 
Probabilistic Solution

- Elements of M are hashed to random variables in (0,1)

- Idea: use the maximum/minimum to estimate the cardinality
  - One pass
  - Constant memory
Probabilistic Solution

- Elements of M are hashed to random variables in (0,1)

- Intuition:
  - If there are 10 distinct elements,
  - Expect the hash values to be spaced about $\frac{1}{10}$th apart from each other

- $\mathbb{E}(\text{max}) = \frac{n}{n+1}$
Probabilistic Solution

\[ h(C) = 0.347 \]

\[ h^+ = 0.347 \]
Probabilistic Solution

$h(D) = 0.773$

$h^+ = 0.773$
$h(B) = 0.512$

$h^+ = 0.773$
Probabilistic Solution

\[ h(B) = 0.512 \]

\[ h^+ = 0.773 \]
Probabilistic Solution

\[ h(Z) = 0.139 \]

\[ h^+ = 0.773 \]
Probabilistic Solution

\[ h(D) = 0.773 \]

\[ h^+ = 0.773 \]
Probabilistic Solution

\[ h(B) = 0.512 \]

\[ h^+ = 0.773 \]
$h(D) = 0.773$

$h^+ = 0.773$
• $\mathbb{E}(\text{max}) = \frac{n}{n+1} = 0.773$

• Estimated cardinality = 3.405

• Actual cardinality = 4
Chassaing Algorithm

- **Simulate** $m$ different hash functions
  - $m$ maxima $h_1^+, h_2^+, ..., h_m^+$

- **Estimate**
  $$\text{Estimate} = \frac{m - 1}{\sum (1 - h_k^+)}$$
Chassaing Algorithm

• $h_k^+ \sim \frac{n}{n+1}$

• $\sum (1 - h_k^+) \sim \sum \frac{1}{n+1} = \frac{m}{n+1}$

• Therefore,
  • Estimate $= \frac{m - 1}{\sum (1 - h_k^+)} \sim n$
Chassaing Algorithm

- Relative error $\approx \frac{1}{\sqrt{m}}$ for a memory of $m$ words

- Minimal variance unbiased estimator (MVUE)
Formal Definition

Instance:
A stream of elements $x_1, x_2, \ldots, x_s$ with repetitions, and an integer $m$
Let $n$ be the number of different elements, denoted by $e_1, e_2, \ldots, e_n$

Objective:
Find an estimate $\hat{n}$ of $n$, using only $m$ storage units, where $m \ll n$
Min/Max Sketches

• Use $m$ different hash functions

• Hash every element $x_i$ to $m$ uniformly distributed hashed values $h_k(x_i)$

• Remember only the minimum/maximum value for each hash function $h_k$

• Use these $m$ values to estimate $n$
Generic Max Sketch Algorithm

Algorithm 1

1. Use $m$ different hash functions
2. For every $h_k$ and every input element $x_i$, compute $h_k(x_i)$
3. Let $h_k^+ = \max\{h_k(x_i)\}$ be the maximum observed value for $h_k$
4. Invoke $ProcEstimate(h_1^+, h_2^+, ..., h_m^+)$ to estimate $n$
Weighted Cardinality Estimation Problem
Weighted Sum of a Stream

- Each element is associated with a weight.
- The goal is to estimate the weighted sum $w$ of the distinct elements.

\[
w = \sum w_i = 0.5 + 0.25 + 1 + 1.25 = 3
\]

<table>
<thead>
<tr>
<th>Element</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.5</td>
</tr>
<tr>
<td>D</td>
<td>0.25</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>1.25</td>
</tr>
</tbody>
</table>
Application Example

- Stream of IP packets received by a server $x_1, x_2, ..., x_s$
- Each packet belongs to a flow (connection) $e_1, e_2, ..., e_n$
- Each flow $e_j$ imposes a load $w_j$ on the server
- The weighted sum $w = \sum w_j$ represents the total load imposed on the server
Formal Definition

**Instance:**
A stream of weighted elements \( x_1, x_2, \ldots, x_s \) with repetitions, and an integer \( m \)
Let \( n \) be the number of different elements, and let \( w_j \) be the weight of \( e_j \)

**Objective:**
Find an estimate \( \hat{w} \) of \( w = \sum w_j \), using only \( m \) storage units, where \( m \ll n \)
Our Contribution

• A unified scheme for generalizing any min/max estimator for the unweighted cardinality estimation problem to an estimator for the weighted cardinality estimation problem.
The Unified Scheme
Observation

• All min/max sketches can be viewed as a two step computation:
  1. Hash each element uniformly into (0, 1)
  2. Store only the minimum/maximum observed value
The Unified Scheme

• In the unified scheme we only change step (1) and hash each element into a Beta distribution.

• The parameters of the Beta distribution are derived from the weight of the element.
Lemma:

Let $z_1, z_2, \ldots, z_n$ be independent RVs, where $z_i \sim Beta(w_i, 1)$

Then,

$$\max\{z_i\} \sim Beta\left(\sum w_i, 1\right)$$
Corollary

• For every hash function,

\[ h_k^+ = \max\{h_k(x_i)\} \sim \max\{U(0,1)\} \sim \max\{Beta(1,1)\} \sim Beta(n, 1) \]

• Thus, estimating the value of \( n \) by Algorithm 1, is equivalent to estimating the value of \( \alpha \) in the \( Beta(\alpha, 1) \) distribution of \( h_k^+ \)
The Unified Scheme

For estimating the weighted sum:
• Instead of associating each element with a uniform hashed value
  • \( h_k(x_i) \sim U(0,1) \)
• We associate it with a RV taken from a Beta distribution
  • \( h_k(x_i) \sim Beta(w_j, 1) \)
  • \( w_j \) is the element’s weight
Algorithm 2

- Use $m$ different hash functions
- For every $h_k$ and every input element $x_i$:
  1. compute $h_k(x_i)$
  2. transform to $h_k^*(x_i) \sim Beta(w_j, 1)$
- Let $h_k^+ = \max\{h_k^*(x_i)\}$ be the maximum observed value for $h_k$
- Invoke $ProcEstimate(h_1^+, h_2^+, ..., h_m^+)$ to estimate the value of $w$
The Unified Scheme

• Practically, if

\[ h_k(x_i) \sim U(0,1) \]

• Then,

\[ h(x_i)^{1/w_j} \sim Beta(w_j,1) \]
## Distributions Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unweighted</td>
<td>$h_k^+ \sim \text{Beta}(n, 1)$</td>
</tr>
<tr>
<td>Weighted</td>
<td>$h_k^+ \sim \text{Beta}(w = \sum w_j, 1)$</td>
</tr>
</tbody>
</table>
The Unified Scheme

- The same algorithm that estimates $n$ in the unweighted case can estimate $w$ in the weighted case.

- $ProcEstimate()$ is exactly the same procedure used to estimate the unweighted cardinality in Algorithm 1.
The Unified Scheme Lemma

Estimating $w$ by Algorithm 2 is equivalent to estimating $n$ by Algorithm 1.

Thus, Algorithm 2 estimates $w$ with the same variance and bias as that of the underlying procedure used by Algorithm 1.
Weighted Generalization for Chassaing Algorithm

- Estimate: \( \frac{m - 1}{\sum (1 - h_k^+)} \)
- But now, \( h_k^+ = \max\{h_k(x_i)\} = \max\{h_k(x_i)^{1/w_j}\} \)
Stochastic Averaging

- Presented by Flajolet in 1985

- Use 2 hash functions instead of $m$

- Overcome the computational cost at the price of negligible statistical efficiency in the estimator’s variance
Stochastic Averaging

• Use 2 hash functions:
  1. $H_1(x_i) \sim \{1,2,\ldots,m\}$
  2. $H_2(x_i) \sim U(0,1)$

• Remember the maximum observed value of each bucket

• The generalization to weighted estimator is similar
Simulation

• We simulate a stream of weighted elements:
  • \( n \) elements from \( r \) weight classes
  • Each class is associated with a different weight, \( w_j \in [w_{\min}, w_{\max}] \)

• Weights distributions:
  • Uniform distribution: \( f_j = \frac{1}{r} \)
  • Normal distribution around \( \frac{1}{2}(w_{\min} + w_{\max}) \)
Method-1 (Benchmark)

- We simulate a new stream of \( w \) unweighted elements \( e_1, e_2, \ldots, e_w \)

- The cardinality of the new stream is equal to the weighted sum \( w \)

- We then run the unweighted algorithm, without weighted adaptation
Method-2 (Unified Scheme)

• We apply our unified scheme and generalize the unweighted algorithm into a weighted algorithm

• We then run it on the original weighted input stream
# Results – Chassaing Algorithm

The table below presents the results of the Chassaing Algorithm for various combinations of weight parameters, bias, and variance ratio. The number of runs was set at 10,000.

<table>
<thead>
<tr>
<th>distribution</th>
<th>#classes</th>
<th>Method-1</th>
<th>Method-2</th>
<th>variance ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform</td>
<td>16</td>
<td>0.00011</td>
<td>0.00046</td>
<td>0.986</td>
</tr>
<tr>
<td>uniform</td>
<td>64</td>
<td>0.00131</td>
<td>0.00056</td>
<td>1.024</td>
</tr>
<tr>
<td>uniform</td>
<td>512</td>
<td>0.00247</td>
<td>0.00134</td>
<td>0.979</td>
</tr>
<tr>
<td>normal</td>
<td>16</td>
<td>0.00252</td>
<td>0.00119</td>
<td>1.014</td>
</tr>
<tr>
<td>normal</td>
<td>64</td>
<td>0.00048</td>
<td>0.00432</td>
<td>1.006</td>
</tr>
<tr>
<td>normal</td>
<td>512</td>
<td>0.00274</td>
<td>0.00051</td>
<td>0.995</td>
</tr>
</tbody>
</table>

\[ n = 100,000; \ m = 32; \ 10,000 \text{ runs} \]
Conclusion

• We showed how to generalize every min/max sketch to a weighted version

• The proposed unified scheme uses the unweighted estimator as a black box, and manipulates the input using properties of the Beta distribution

• We proved that estimating the weighted sum by our unified scheme is statistically equivalent to estimating the unweighted cardinality
Questions?
Thank You