Approximation Techniques for Data Management Systems

“We are drowning in data but starved for knowledge”
John Naisbitt

CS 186
Fall 2005

Fast Approximate Answers

• Primarily for Aggregate queries
• Goal is to quickly report the leading digits of answers
  - In seconds instead of minutes or hours
  - Most useful if can provide error guarantees

E.g., Average salary

$59,000 +/- $500 (with 95% confidence) in 10 seconds
vs. $59,152.25
in 10 minutes

• Achieved by answering the query based on compact synopses of the data
• Speed-up obtained because synopses are orders of magnitude smaller than the original data

Sampling: Basics

• Idea: A small random sample S of the data often well-represents all the data
  - For a fast approx answer, apply the query to S & “scale” the result
  - E.g., R.a is (0,1), S is a 20% sample
    select count(*) from R where R.a = 0
    R.a
    0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
    Red = in S
select 5 * count(*) from S where S.a = 0
Est. count = 5 * 2 = 10. Exact count = 10

Unbiased: For expressions involving count, sum, avg: the estimator
is unbiased, i.e., the expected value of the estimator is the actual answer, even for (most) queries with predicates
• Leverage extensive literature on confidence intervals for sampling
  Actual answer is within the interval [a,b] with a given probability
  E.g., 54,000 ± 600 with prob ≥ 90%

Sampling: Confidence Intervals

<table>
<thead>
<tr>
<th>Method</th>
<th>95% Confidence Interval (a)</th>
<th>Guarantee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jackknife</td>
<td>1.64 * std / sqrt(n)</td>
<td>± 5%</td>
</tr>
<tr>
<td>Bootstrap (bias-corrected)</td>
<td>1.64 * std / sqrt(n)</td>
<td>always</td>
</tr>
<tr>
<td>Bootstrap (bias-uncorrected)</td>
<td>1.64 * std / sqrt(n)</td>
<td>always</td>
</tr>
</tbody>
</table>

Confidence intervals for Average: select avg(R.a) from R
(Con replace R.a with any arithmetic expression on the attributes in R)
std(R) = standard deviation of the values of R.a, avg(R) = AVERAGE for R.

• If predicates, S above is subset of sample that satisfies the predicate
• Quality of the estimate depends on the variance in R & |S| after the predicate: So 10% sample may suffice for 100 row relation!
  - Advantage of larger samples: can handle more selective predicates
Sampling from Databases

- Sampling disk-resident data is slow
  - Row level sampling has high 1/O cost
  - Must bring in entire disk block to get the row
  - Block level sampling: rows may be highly correlated
  - Random access pattern: possibly via an index
  - Need to account for the variable number of rows in a page, children in an index node, etc.

- Alternatives
  - Random physical clustering: destroys “natural” clustering
  - Precomputed samples: must incrementally maintain (at specified size)
  - Fast to use: packed in disk blocks, can sequentially scan, can store as relation and leverage full DBMS query support, can store in main memory

One-Pass Uniform Sampling

- Best choice for incremental maintenance
  - Low overheads, no random data access

- Reservoir Sampling [Vu85]: Maintains a sample S of a fixed-size M
  - Add each new item to S with probability M/N, where N is the current number of data items
  - If add an item, evict a random item from S
  - Instead of flipping a coin for each item, determine the number of items to skip before the next to be added to S

Histograms

- Partition attribute value(s) domain into a set of buckets

- Issues:
  - How to partition
  - What to store for each bucket
  - How to estimate an answer using the histogram

- Long history of use for selectivity estimation within a query optimizer

- Recently explored as a tool for fast approximate query processing

1-D Histograms

- Number of buckets B \ll domain size
- Each bucket just stores a total count
  - Distributed uniformly across values in the bucket

- Partition criteria
  - Equi-width: equal number of domain values per bucket (bad!!)
  - Equi-depth/height: equal count (“mass”) per bucket
  - V-Optimal: minimize total variance of value counts in buckets

Answering Queries Using Histograms

- Answering queries from 1-D histograms in general:
  - (Implicitly) map the histogram back to an approximate relation, & apply the query to the approximate relation

- Inside each bucket: Uniformity Assumption
  - Continuous value mapping
    - Count spread evenly among bucket values
  - Uniform spread mapping
    - Need number of distinct in each bucket

Haar Wavelet Synopses

- Wavelets: mathematical tool for hierarchical decomposition of functions/signals
  - Haar wavelets: simplest wavelet basis, easy to understand and implement
    - Recursive pairwise averaging and differencing at different resolutions

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Averages</th>
<th>Detail Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0 = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]</td>
<td>(0, -1, 0)</td>
</tr>
<tr>
<td>2</td>
<td>(2, 1, 4, 4)</td>
<td>(-1, 0)</td>
</tr>
<tr>
<td>1</td>
<td>(1, 5, 4)</td>
<td>(0.5, 0)</td>
</tr>
<tr>
<td>0</td>
<td>(2.75)</td>
<td>(-1.25)</td>
</tr>
</tbody>
</table>

Haar wavelet decomposition: [2.75, -1.25, 0.5, 0, 0, -1, -1, 0]
**Haar Wavelet Coefficients**

- Hierarchical decomposition structure (a.k.a. Error Tree)
- Conceptual tool to "visualize" coefficient supports & data reconstruction
- Reconstruct data valued D:
  - d(l) = \sum (x/l) * (coefficient on path)
- Range sum calculation d(h):
  - d(h) = simple linear combination of coefficients on path to l, h
- Only O(log N) terms

**Wavelet Data Synopses**

- Compute Haar wavelet decomposition of D
- Coefficient thresholding: only B pi D coefficients can be kept
  - B is determined by the available synopsis space
- Approximate query engine can do all its processing over such compact
coefficient synopses (joins, aggregates, selections, etc.)
- Conventional thresholding: Take B largest coefficients in absolute
normalized value
  - Normalized Haar basis: divide coefficients at resolution j by \sqrt{2^j}
  - All other coefficients are ignored (assumed to be zero)
  - Provably optimal in terms of the overall Sum-Squared (L2) Error

**Multi-dimensional Data Synopses**

- Problem: Approximate the joint data distribution of multiple attributes
  - Motivation
    - Selectivity estimation for queries with multiple predicates
    - Approximating general relations
  - Conventional approach: Attribute-Value Independence (AVI) assumption
    - sel(p(A1) & p(A2) & ... ) = sel(p(A1)) * sel(p(A2)) * ...
    - Simple -- one-dimensional marginals suffice
    - BUT: almost always inaccurate, gross errors in practice

**Multi-dimensional Histograms**

- Use small number of multi-dimensional buckets to directly approximate
  the joint data distribution
- Uniform spread & frequency approximation within buckets
  - n(i) = no. of distinct values along Ai, F = total bucket frequency
  - approximate data points on a n(1)x n(2) * ... uniform grid, each
    with frequency F / (n(1)x n(2)*...)

**Data Synopses in Commercial DBMSs**

- Sampling operators ans 1-D histograms are available in
  most commercial DBMSs
  - Oracle, DB2, SQL Server...
  - Used internally but also exposed to user (e.g., store
    "sample view")
  - SQL Server has support for 2-D histograms!
  - The next step: Synopses for XML?!
  - How do you effectively summarize a graph structure
    for queries like /a/b[d]/c/??

**Data-Stream Management**

- Traditional DBMS -- data stored in finite, persistent data sets
- Data Streams -- distributed, continuous, unbounded, rapid, time varying, noisy, ...
- Data-Stream Management -- variety of modern applications
  - Network monitoring and traffic engineering
  - Telecom call-detail records
  - Network security
  - Financial applications
  - Sensor networks
  - Web logs and clickstreams
SNMP/PMON/NetFlow data records arrive 24x7 from different parts of the network.
- Truly massive streams arriving at rapid rates
  - AT&T collects 600-800 GigaBytes of NetFlow data each day.
- Typically shipped to a back-end data warehouse (off site) for off-line analysis.

Data-Stream Processing Model

- Approximate answers often suffice, e.g., trend analysis, anomaly detection.
- Requirements for stream synopses:
  - Single Pass: Each record is examined at most once, in (fixed) arrival order.
  - Small Space: Log or polylog in data stream size.
  - Real-time: Per-record processing time (to maintain synopses) must be low.

Distinct Value Estimation

- Problem: Find the number of distinct values in a stream of values with domain \([0, \ldots, N-1]\).
  - Zeroth frequency moment \(F_0\): \(L_0\) (Hamming) stream norm.
  - Statistics: number of species or classes in a population.
  - Important for quality optimizers.
  - Network monitoring: distinct destination IP addresses, source/destination pairs, requested URLs, etc.

Example (N=64):
Data stream \[0 \quad 5 \quad 3 \quad 0 \quad 1 \quad 7 \quad 5 \quad 1 \quad 0 \quad 3 \quad 7\]
Number of distinct values: 5

Hash (aka FM) Sketches for Count Distinct

- Assume a hash function \(h(x)\) that maps incoming values \(x\) in \([0, \ldots, N-1]\)
  uniformly across \([0, \ldots, 2^L-1]\), where \(L = O(\log N)\).
- Let \(\text{lab}(y)\) denote the position of the least-significant 1 bit in the binary representation of \(y\).
  - A value \(x\) is mapped to \(\text{lab}(h(x))\).
- Maintain \(\text{Hash}\) sketch: \(\text{BITMAP}\) array of \(L\) bits, initialized to 0.
  - For each incoming value \(x\), set \(\text{BITMAP}(\text{lab}(h(x))) = 1\).

Example:
- \(x = 5 \Rightarrow h(x) = 101100 \Rightarrow \text{lab}(h(x)) = 2\)
- \(\text{BITMAP}\):

\[
\begin{array}{c}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

- Position: \(\log(d)\)
- Fringe of 0/1s around \(\log(d)\)

- Let \(R\) is position of rightmost zero in \(\text{BITMAP}\)
  - Use as indicator of \(\log(d)\).
- [FM85] prove that \(E[R] = \log(d)\) when \(d > 1/2\).
  - Estimate \(d = 2^R/\phi\)
  - Average several iid instances (different hash functions) to reduce estimator variance.
A Little Streaming Puzzle...

- **Input**: A stream of N numbers/elements
- **Output**: The stream majority element (if one exists)
  - e is a majority element if frequency(e) > N/2

- Q: How do you do this in small space??
  - Hint: Use just two memory locations
  - Hint++; Look at this as a "knockout tournament"

- Feeling adventurous?
  - How do you do the same majority query over a stream of insertions and deletions?
  - **Input**: Stream of <e, +> = insert e, <e, -> = delete e
  - Hint: Use a little more memory...

In Summary: Not your parents’ DBMS!

- Database/data-management research goes far beyond the basics!
- Extends from distributed systems to theory to approximation algorithms to probability/statistics to ...
  - Applications: data mining, sensors, p2p, ...
  - Just pick up a copy of recent SIGMOD/VLDB proceedings
- More and more relevant in dealing with the "data tsunami/"
  - Data is everywhere! And, it’s constantly growing in volume!
- Exciting, relevant research!

More details...

- Tutorial slides on approximate query processing and data streams
  - [http://www2.berkeley.intel-research.net/~minos/tutorials.html](http://www2.berkeley.intel-research.net/~minos/tutorials.html)