Drawing Samples from Databases

Chris Jermaine

Rice University

So Far...

We have talked about how to USE samples

- But what sorts of samples are there?
- And how about how to actually produce them?
- This is what we will discuss in this lecture
- We begin by discussing the different "flavors" of sampling

Simple Random Sampling With Replacement

This is the classic form of sampling

- Note: samples are i.i.d.
- So classical theory like the CLT applies
- To draw a sample of size n from a relation R of size $|R|$:

```
Repeat n times:
  Produce a random number r from 1 to |R|, inclusive
  Obtain the rth item from the dataset
  Add it to the sample
```

SRSWR drawbacks

- Without-replacement sampling has lower variance than with replacement sampling for large samples
- Standard single-table HT estimator is:

$$Y = \frac{|\mathrm{R}|}{n} \sum_j X_j t_j$$

- Variance of standard single-table SUM estimator is:

$$\sigma^2(Y) = \frac{|\mathrm{R}|^2 \sigma^2(\mathrm{R})}{n}$$

- Note this does not ever go to zero

Simple Random Sampling Without Replacement

This is the standard form of database sampling

- To draw a sample of size n from a relation R of size $|R|$:

```
Repeat n times:
  While we have not added a new item:
    Produce a random number r from 1 to |R|, inclusive
    If the rth item from the data not in the sample
      Obtain this item and add it to the sample
```

Simple Random Sampling Without Replacement

Main advantage over SRSWR is low variance at high sampling rate

- Standard HT estimator (single table) is identical:

$$Y = \frac{|\mathrm{R}|}{n} \sum_j X_j t_j$$

- However, variance is:

$$\sigma^2(Y) = \frac{|\mathrm{R}|(|\mathrm{R}| - n)\sigma^2(\mathrm{R})}{n}$$

- Note that this shrinks to zero with large sampling fraction

Bernoulli and Poisson Sampling

Idea is that we apply an independent test to each data item

Accept item j with probability p_j

In Bernoulli sampling

- p_j same over entire data set

In Poisson sampling

- p_j varies per tuple

To perform this type of sampling:

```
For j = 1 to |R|:
  Generate a random number r from 0 to 1
  If r is less than p_j, include tuple j in the sample
```

Bernoulli and Poisson Sampling

- Standard single-table HT SUM estimator takes the form:

$$Y = \sum_{i \in \text{sample}} \frac{t_i}{p_i}$$

- And variance is:

$$\sigma^2(Y) = \sum_i (\frac{1}{p_i} - 1) t_i^2$$

- With estimator:

$$\hat{\sigma}^2(Y) = \sum_{i \in \text{sample}} \frac{1}{p_i} (\frac{1}{p_i} - 1) t_i^2.$$

Bernoulli and Poisson Sampling

Relative to other schemes...

- Can make Poisson variance very small relative to sample size... choose each p_i...
- to minimize $\sigma^2(Y)$ subject to $\sum_i p_i =$ desired sample size
- Tends to choose large p_j for large t_j
- This leads to a "biased" sampling scheme, where large $|t_i|$ values chosen

Stratified Sampling

Basic idea

- All tuples first grouped into m classes
- Also called "strata"
- To obtain stratified sample, SRSWOR is performed on each strata
- Number of samples from the ith strata is n_i
- Where $\sum_{i=1}^{m} n_i = n$

Stratified Sampling

- To estimate SUM query result...
- Let $\mathtt{R}_i$ denote ith stratum
- Then define the HT estimator

$$Y_i = \frac{|\mathtt{R}_i|}{n_i} \sum_{t_j \in \mathtt{R}_i} X_j t_j$$

- and

$$Y = \sum_{i=1}^{m} Y_i.$$

Stratified Sampling

- Since sampling processes are independent, variance is sum of individual variances:

$$\sigma^2(Y) = \sum_{i=1}^{m} \sigma^2(Y_i)$$

Neyman Allocation

Is one of the most classic results in survey sampling theory

- Prescribes a set of n_i values to minimize $\sigma^2(Y)$
- Choose each n_i so that:

$$n_i = \frac{n|\mathrm{R}_i|\sigma^2(Y_i)}{\sum_{j=1}^{m}|\mathrm{R}_j|\sigma^2(Y_j)}$$

- In practice, a small pilot sample can be used to estimate the $\sigma^2(Y_j)$
- Can produce a DRAMATIC variance reduction
- Under what circumstances could it be used in DBMS sampling?

OK, so how to sample from a database?

In practice, there are many ways

- Most DBs stored on block-based devices
  - So it may make sense to sample blocks, not tuples
- And sometimes, data stored in indexes
  - So we may need to use the index to sample
- Or else, we may simply scan the entire DB and produce a sample in a pass

How to do each of these?

Block-based vs. tuple-based sampling

In practice, most data organized into blocks

Only have random access at block level

- If you want one tuple from a block...
- You get the whole block

Means that if ≥ 1 tuple sampled from each block...

- Sampling as expensive as scanning full data set

Alternative: randomly choose full BLOCKS to sample

- Sample a block? Use ALL its tuples

Block-based sampling in detail

If we sample blocks, can use most of what we've developed with little modification

- Rather than letting $t_{j_1,j_2,\ldots,j_m}$ refer to value after applying functions f, g, and h to concat of tuple j_1, tuple j_2, etc.
- $t_{j_1,j_2,\ldots,j_m}$ instead refers to the total aggregate value by applying f, g, and h to the cross product of all of the tuples in the j_1th block from the first relation, the j_2th block from the second relation, and so on

Block-based sampling in detail

- $B_j^{(i)}$ refers to the set of tuples in the jth block from the ith relation
- Let $f'(t) = f(t)$ if $g(t) = \texttt{true}$ and $h(t) = gid$; otherwise, $f'(t) = 0$
- Then let

$$t_{j_1,j_2,\ldots,j_m} = \sum_{t_1 \in B_{j_1}^{(1)}} \sum_{t_2 \in B_{j_2}^{(2)}} \cdots \sum_{t_m \in B_{j_m}^{(m)}} f'(t_1 \bullet t_2 \bullet \cdots \bullet t_m).$$

- Also redefine $X_{j_1,j_2,\ldots,j_m}$ to be the random variable that controls whether the sample contains block j_1 from relation 1, block j_2 from relation 2, and so on
- Our results of SPJGB queries hold without further modification!

So, should we just always sample blocks?

Makes a lot of sense in terms of I/O efficiency

- But are issues...
- You end up with a lot more data
- And in the extreme case all data on a block are correlated
- So you might burn a lot of CPU for little accuracy gain

Scan-based sampling

Fastest way to sampling data is to scan it

- Goal: for any $n \geq 1$
- A size n random sample of the file obtained
- After n tuples have been read
- Only the case if the file has been randomized
- Idea pioneered by Hellerstein, Haas, and Wang (SIGMOD 1997): "online aggregation"
- Attach a random bit string to each tuple, then sort (TPMMS)

SRSWOR Using a Reservoir

Classical problem:

- Unbounded stream of tuples
- Want to draw a SRSWOR of size n in one pass
- Classic algorithm for this is "reservoir sampling"

```
res = {}
i = 0
while still data:
  i++
  if |res| < n:
    add new tuple to res
  otherwise:
    add new tuple to res with probability n/i
    evicting a random existing tuple
```

SRSWOR From a B+-Tree

If the tree is "ranked" this is easy

- In a ranked tree, each link to a child node as a tuple count
- So I have a buch of (ptr, cnt) pairs
- If I want a sample of size n from a leaf node, just sample normally
- If I want a sample from an internal node:

$numsam[j]$ = 0 **for** j in 1 to $numKids$
for i = 0 to n:
  choose child j with prob $\frac{cnt_j}{\sum_{j'} cnt_{j'}}$
  cnt_j--
  $numsam[j]$++
for j in 1 to $numKids$
  recursively draw $numsam[j]$ samples from the jth subtree

SRSWR from a B+-Tree w/o rank info

What if tree is not ranked?

- Idea (Olken and Rotem, 1989): pretend each tree is 100
- That is, get UB on the number of children in internal, leaf node
- Call these values max_i, max_l
- Sampling procedure is again recursive
- Here's a one-at-a-time algorithm (can extend to batch)
- Note we add a new "reject and start over" which requires re-running from root

```
Leaf node:
Choose a value r from 1 to max_l
If < r items in leaf, reject and start over
Else, return rth item

Internal node:
Choose a value r from 1 to max_i
If < r items in IN, reject and start over
Else, descend into rth subtree and run appropriate algorithm
```

That's it!

Questions?