Approximate Counting Using Stratified Sampling

February 16, 2014

1 Introduction

In analytics databases, one of the most important queries that should return quickly is the count. One method of approximating counts in a single table is by sampling. Let the total number of rows in a table be N, and let's say p proportion of these examples satisfy some predicate. Since we assume we always have access to N, if we know with high probability$p \in [\hat{p} - \epsilon, \hat{p} + \epsilon]$, then we also know the number of rows matching the query, $Np \in [N\hat{p} - N\epsilon, N\hat{p} + N\epsilon]$ with high probability. From now on, we'll only talk about approximating p.

```
Sample until h_stop.
Three cases:
    crossover >> h_stop: We've reached the stopping point at h_stop!
    h_stop >> crossover: empirical term dominates from now on (due to $\frac{1}{\sqrt{t
    h_stop ~ crossover: this is difficult
```

2 Simulated Results

See simulated_results.txt for more detailed numbers used in this discussion. For even more details, see Online_Stratified_Sampling_Simulations.ipynb.

3 Dynamic vs. Static Allocation of ϵ to Strata

Recall the empirical bernstein bound: with probability $1 - \delta$, $\epsilon_t =_{def} |\hat{\mu}_t - \mu| \leq \frac{\frac{5}{6}C_\delta}{t - C_\delta} + \hat{\sigma}_t\sqrt{\frac{C_\delta}{t - C_\delta}}$ simultaneously for all $t > C_\delta$. Let's compare the two parts of the error bound: $h_t = \frac{\frac{5}{6}C_\delta}{t - C_\delta}$ and $e_t = \hat{\sigma}_t\sqrt{\frac{C_\delta}{t - C_\delta}}$. Since for $t \to \infty$ $\hat{\sigma}_t \to \sqrt{p(1-p)}$, for small p e_t can be smaller than h_t. However, $h_t \sim \frac{1}{t}$ and $e_t \sim \frac{1}{\sqrt{t}}$ for large t, so e_t eventually dominates.

In the following discussion we'll assume p is relatively large, so that the e_t term of the error bound dominates, and we can ignore h_t.

We'll assume from now on that $\epsilon_t \approx e_t \sim \sqrt{\frac{p(1-p)}{t}}$.

Let's say we have N rows total in a database table, and these are partitioned into k sets of rows $i = \{1, ...k\}$ with n_i rows per partition. In each partition, p_i proportion of the rows match a particular query. So $p_i n_i$ rows in partition i match the query.

We'll sample t_i rows with replacement from each partition: iid from the distribution defined by the n_i rows in that partition. Then based on our assumption that $\epsilon_t \approx \frac{\sqrt{p(1-p)}}{t}$, the total error in our estimation of $p = \sum_i \frac{n_i}{N} p_i$ is $\sum_i \frac{n_i}{N} \epsilon^i_{t_i} \approx \sum_i \frac{n_i}{N} \sqrt{\frac{p(1-p)}{t_i}}$. The total number of samples used is $\sum_i t_i$. Then we might want to minimize the cost $\sum_i t_i$ such that we obtain ϵ-accuracy, so $\sum_i \frac{n_i}{N} \sqrt{\frac{p(1-p)}{t_i}} \leq \epsilon$. In a distributed environment, we might not always be interested in the total cost $\sum_i t_i$, but in the most samples needed per partition, $max_i t_i$. An objective that generalizes these cases, and allows a balance between the two, is the L_α-norm of $\vec{t}$. This is defined as $\left\|\vec{t}\right\|_\alpha = (\sum_i t_i^\alpha)^{1/\alpha}$, $\alpha \geq 1$. For $\alpha \to \infty$ this gives the max norm, and for $\alpha = 1$ this is $\sum_i t_i$.

In the next sections we compare methods of allocating t_i to partitions under these assumptions and definitions of objectives/constraints. We show that the optimal allocation performs significantly better than locally allocating samples to partitions, for some choices of $\vec{n}$ and $\vec{p}$.

3.1 Optimal Dynamic Loss, for L_α-loss on <#samples per strata> vector

Here is the optimization problem formulated in the previous section:

$$\begin{array}{cc} min & \left\|\vec{t}\right\|_\alpha, \alpha \geq 1 \\ s.t. & \sum_i a_i \frac{1}{\sqrt{t_i}} \leq b \end{array} \sim \begin{array}{cc} min & \sum_i t_i^\alpha, \alpha \geq 1 \\ s.t. & \sum_i a_i \frac{1}{\sqrt{t_i}} \leq b \end{array}$$

Note that $a_i =_{def} n_i \sqrt{p_i(1-p_i)}$, and $b_i = N\epsilon$.

Using a variable substitition:

$$\sim_{x=\frac{1}{\sqrt{t_i}}} \begin{array}{cc} min & \sum_i \frac{1}{x_i^{2\alpha}}, \alpha \geq 1 \\ s.t. & \sum_i a_i x_i \leq b \end{array}$$

We can directly solve for the KKT conditions of the above program, to get x_i.

$$\frac{2\alpha}{x_i^{2\alpha+1}} = \lambda a_i \sim x_i^{2\alpha+1} = \frac{2\alpha}{\lambda a_i} \sim x_i = \sqrt[2\alpha+1]{\frac{2\alpha}{\lambda a_i}}$$

$$\sum_i a_i x_i = b \sim \sum_i a_i \left(\sqrt[2\alpha+1]{\frac{2\alpha}{\lambda a_i}} \right) = b \sim \sqrt[2\alpha+1]{\frac{1}{\lambda}} \sqrt[2\alpha+1]{2\alpha} \sum_i a_i^{1-\frac{1}{2\alpha+1}} = b$$

$$\lambda = 2\alpha \left(\frac{1}{b}\right)^{2\alpha+1} \left(\sum_i a_i^{\frac{2\alpha}{2\alpha+1}}\right)^{2\alpha+1}$$

$$x_i^* = \frac{b}{\left(\sum_j a_j^{\frac{2\alpha}{2\alpha+1}}\right)^{\ 2\alpha+1\sqrt{a_i}}}$$

Finally, plugging x^* back into the objective $f(x^*)$, we get the optimal sample-cost.

$$f(x^*) = \left\|\vec{t}\right\|_\alpha = \left(\sum_i \frac{1}{x_i^{2\alpha}}\right)^{1/\alpha} = \left(\left(\sum_i a_i^{\frac{2\alpha}{2\alpha+1}}\right)^{2\alpha} \sum_i \frac{a_i^{\frac{2\alpha}{2\alpha+1}}}{b^{2\alpha}}\right)^{1/\alpha} = \left(\frac{1}{b^{2\alpha}} \left(\sum_i a_i^{\frac{2\alpha}{2\alpha+1}}\right)^{2\alpha+1}\right)^{1/\alpha}$$

$$\frac{\left(\sum_i \left(a_i^2\right)^{\frac{\alpha}{2\alpha+1}}\right)^{\frac{2\alpha+1}{\alpha}}}{b^2} = \frac{\left\|\vec{a}^2\right\|_{\alpha/(2\alpha+1)}}{b^2}$$

3.2 Loss of Static Sample Allocation, for L_α-loss on <#samples per partition> vector

In the optimal allocation of samples to partitions above, we chose $\vec{t}$, knowing $\vec{p}$ ahead of time. Usually we don't know this, and so allocate samples based solely on the proportion of rows in that partition. An additional benefit of this method: this allows for communication-free parallel sampling in each partition. Each partition (let's say each corresponds to a compute node) is given an ϵ_i. It then samples until $\epsilon_t \leq \epsilon_i$, then stops and returns. We choose each ϵ_i such that the overall error is bounded by ϵ. This gives us:

$\sum_i \frac{n_i}{N}\epsilon_i = \epsilon$, $\epsilon_i = \frac{N}{kn_i}\epsilon = \sqrt{\frac{p_i(1-p_i)}{t_i}} = \sqrt{p_i(1-p_i)}x_i$. Then $x_i = \frac{N\epsilon}{kn_i\sqrt{p_i(1-p_i)}} = \frac{b}{ka_i}$

And plugging the solution into the objective yields:

Then $f(x^{local}) = \left\|\left(\vec{t}\right)\right\|_\alpha = \left\|\begin{pmatrix} \frac{1}{x_1^2} \\ \vdots \\ \frac{1}{x_k^2} \end{pmatrix}\right\|_\alpha = \left(\sum_i \frac{1}{x_i^{2\alpha}}\right)^{1/\alpha} = \left(\sum_i \frac{k^{2\alpha}a_i^{2\alpha}}{b^{2\alpha}}\right)^{1/\alpha} = \frac{k^2}{b^2}\left(\sum_i \left(a_i^2\right)^\alpha\right)^{1/\alpha} = \frac{k^2}{b^2}\left\|\vec{a}^2\right\|_\alpha$

3.3 Improvement Ratio

For different $\vec{n}$ and $\vec{p}$, we compare the sampling performance of the optimal and local methods above. We do this by analyzing the ratio of the sampling performance:

$$g_k(\vec{a}) = \frac{f\left(x^{local}\right)}{f\left(x^*\right)} = \frac{\frac{k^2}{b^2}\left\|\vec{a}^2\right\|_\alpha}{\frac{\left\|\vec{a}^2\right\|_{\alpha/(2\alpha+1)}}{b^2}} = \frac{k^2\left\|\vec{a}^2\right\|_\alpha}{\left\|\vec{a}^2\right\|_{\alpha/(2\alpha+1)}}$$

We know that for $\gamma > \beta > 0$, $\left\|\vec{y}^2\right\|_\gamma \leq \left\|\vec{y}^2\right\|_\beta$. Equality is obtained when $\vec{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ since $(1^\gamma)^{1/\gamma} = \left(1^\beta\right)^{1/\beta} = 1$, and $\frac{\left\|\vec{y}^2\right\|_\gamma}{\left\|\vec{y}^2\right\|_\beta}$ is minimized when $\vec{y} = \vec{1}$.

In this case, $\left\|\vec{y}^2\right\|_\gamma = k^{1/\gamma}$ and $\left\|\vec{y}^2\right\|_\beta = k^{1/\beta}$, so

$$\frac{\left\|\vec{y}^2\right\|_\gamma}{\left\|\vec{y}^2\right\|_\beta}(min) = k^{\frac{1}{\gamma}-\frac{1}{\beta}}$$

Using these facts above ($\gamma = \alpha$ and $\beta = \frac{\alpha}{2\alpha+1}$), we have:

$$g_k^{min} = k^2 k^{\frac{1}{\alpha}-\frac{2\alpha+1}{\alpha}} = k^{2+\frac{1}{\alpha}-2-\frac{1}{\alpha}} = 1$$

and

$$g_k^{max} = \frac{k^2 1}{1} = k^2$$

3.4 Intuition for Max Norm ($\alpha = \infty$)

One might guess that the

The local allocation for the max norm doesn't perform as well. The optimal allocation is uniform *time*, not uniform ϵ_i.