CS 368 (Stanford, Spring 2022) Lecture 1

Adapted from CS 369G, Spring 2016. Scribes: Tommy Li, Alfred Xue, Paris Syminelakis.

Counting Distinct Elements

1 Overview

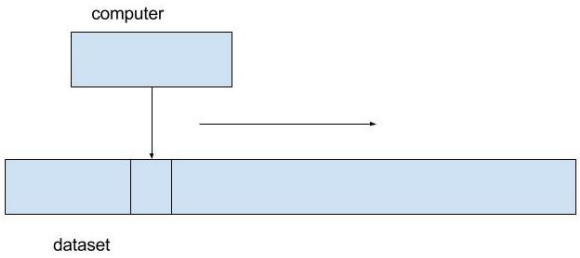
This lecture introduces the Data Stream Model and shows how hash functions and a popular technique called median-of-means may be used to develop a good estimator for the number of distinct elements in a large data set.

2 The Data Stream Model

Imagine you are working with Google's log analysis team. You want to make a query about a data set that is

- so big that it doesn't fit on a single computer, and
- so big that a polynomial running time isn't good enough.

What can you do? To satisfy these constraints, we think about algorithms that follow the Data Stream Model [Mor78]. A computer implementing such an algorithm can only access data in a single pass, and only one piece at a time. Therefore at any given moment it only has access to its own memory and a single piece of the input data.



Here's an example question: Find a counter that uses less than $\log n$ space. This clearly can't be done perfectly, so cutting corners in some way must be necessary. We could cut corners by allowing approximation in the sense that we require that, given the true number k of elements, our counter returns a value between $k(1-\epsilon)$ and $k(1+\epsilon)$. We could also cut corners by allowing some small possibility of failure, by only requiring that our algorithm succeeds with probability $1-\delta$, where δ is some small number (e.g $\delta < 10^{-6}$). In this case, $O(\log \log n)$ bits suffice [FM85; AMS99].

3 Counting Unique Entries

Let's return to the story of being in the Google log-analysis team. Say you want to count the number of distinct queries in a list of queries. Now what do you do? One naive approach would be to use a hash table to store all each query as you receive it. However, this would require a very large hash table! The memory required would be at least linear in the number of distinct entries, which would already be too much. Another approach might be to attempt to use some subset of the queries as a representative of the entire data set, e.g. by counting the number k_m of distinct elements in some random selection of m elements of the n entries and estimating the actual number k of distinct elements as $\frac{n}{m}k_m$. It turns out this type of approach fails as well, roughly because you can't reliably gain information about things that appear extremely infrequently reliably. For example, one cannot reliably distinguish between $\underbrace{000\ldots\ldots000}_{n \text{ times}}$ and $\underbrace{000\ldots\ldots000}_{n-m \text{ times}}123\ldots m$ using a random selection of m elements. So a different approach is needed.

4 The Estimator

One idea that does work is to use a hash function $h : U \to [0, 1]$. What we do is to hash each entry x_i of the data as we see it, and keep track of the minimum seen hash value in our memory.

Claim: If there are k distinct elements $x_1, x_2, \ldots, x_k$ then $E\left[\min_{i=1,\ldots,k} h(x_i)\right] = \frac{1}{k+1}$.

Proof. Let $Y = \min(h(x_1), \ldots, h(x_k))$. We will assume for now that the values $h(x_1), \ldots, h(x_k)$ are independently distributed uniform random variables over the interval $[0, 1]$. It follows that $Pr[h(x_i) \leq t] = t$ and by independence $Pr[Y > t] = (1-t)^k$. Taking the complement of the event we get $Pr[Y \leq t] = 1 - (1-k)^k$. We differentiate this cumulative distribution function with respect to t to get $Pr[Y \in [t, t+dt]] = k(1-k)^{k-1}dt$. We can now use this to compute the expectation of Y, getting:

$$\begin{aligned}
E[Y] &= \int_0^1 tk(1-t)^{k-1}dt \\
&= k\int_0^1 (1-(1-t))(1-t)^{k-1}dt \\
&= k\left(\int_0^1 (1-t)^{k-1}dt - \int_0^1 (1-t)^k dt\right) \\
&= k\left(\frac{1}{k} - \frac{1}{k+1}\right) = \frac{1}{k+1}
\end{aligned}$$

□

This is useful because it means that Y may be used as a tool to estimate k.

5 Median of Means

While we have an algorithm that has an expected value of k, we have done no analysis on the probability the algorithm returns a value close to k. One naive approach is to simply take the mean of multiple instantiations of this algorithm. To do this, we can build multiple hash functions, and keep track of the minimum of all of the hash functions, and then take the average of the minimums of these hash functions.

Let Z be the average of the minimums of these hash functions. That is, $Z = \frac{y_1 + y_2 + \ldots y_t}{t}$. We can say that $E[Z] = E[Y_i]$, $Var[Z] = \frac{Var[Y]}{t}$. To compute the variance of Y, we note that $Var[Y] = E[Y^2] - (E[Y])^2$. To increase accuracy, we can build multiple hash functions, and keep track of the minimum of all of the hash functions, and then take the average of them. To compute the variance of Y, we note that

$$\begin{aligned} E[Y^2] &= \int_0^1 t^2 k(1-k)^{k-1} dt \\ &= k[\frac{1}{k} - \frac{2}{k+1} + \frac{1}{k+2}] \end{aligned}$$

Thus

$$\begin{aligned} &Var[Y] = E[Y^2] - (E[Y])^2 \\ &Var[Z] = E[Z^2] - (E[Z])^2 = \frac{2}{(k+1)(k+2)} - \frac{1}{k+1}^2 \leq \frac{1}{(k+1)^2} \\ &.Var[Y] \leq (E[Y])^2 \\ &Var[Z] \leq \frac{(E[Z])^2}{t} \end{aligned}$$

Now we can bound the probability we are within δ of k.

$$\Pr\left(|Z - E[Z]| \geq \epsilon \cdot E[Z]\right) \leq \frac{Var(Z)}{(\epsilon \cdot E[Z])^2} \leq \frac{1}{\epsilon^2 t}$$

This isn't sufficiently small in relation to t, so we use a different technique to get high probabilities, by applying Chernoff bounds in a specific way. The method we are going to use is called *median of means*.

1. Lets say we have several groups of hash functions z_i, each group has $\frac{4}{\epsilon^2}$ estimators, and has probability $1/4$ to report a value outside its estimated range.
2. We now return median$(z_1, z_2, \ldots, z_s)$.

There are two ways the median can be bad, it can be too low, or it can be too high. In either case, at least half of these elements are bad. We can use indicator variables for each z_i, that return 1 if it is bad, and 0 otherwise. The expected sum of these indicator variables is $\frac{s}{4}$. The probability that this value is more than $\frac{s}{2}$ can be expressed as follows.

$$\Pr[\text{at least half of } z_1, z_2, \ldots z_i \text{ are bad}] \leq e^{-(2\ln 2 - 2 + 1)\cdot s/4} \leq e^{-s/11}$$

which when set to be equal to δ allows us to solve $s = 11 \ln \frac{1}{\delta}$.

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating the frequency moments". In: *Journal of Computer and system sciences* 58.1 (1999), pp. 137–147.

[FM85] Philippe Flajolet and G Nigel Martin. "Probabilistic counting algorithms for data base applications". In: *Journal of computer and system sciences* 31.2 (1985), pp. 182–209.

[Mor78] Robert Morris. "Counting large numbers of events in small registers". In: *Communications of the ACM* 21.10 (1978), pp. 840–842.