CS787: Advanced Algorithms	**Scribe:** Muni Sreenivas Pydi
Lecture 22: Streaming Algorithms / Dimension Reduction	**Date:** April 18, 2019

22.1 Counting the number of distinct elements in a stream

Given a stream of m data points $i_1, \ldots, i_m$ where $i_t \in U$ and $|U| = n$, we would like to estimate the count, $|\{x \in U : \exists t \text{ with } i_t = x\}|$, which is the number of distinct elements in the stream.

We will use the idea of a hash table to keep track of the distinct elements in the stream. Every time we see a new element in the stream, we mark the location corresponding to the element in the hash table. Since multiple occurrences of the same element gets mapped to the same location in the table, we can count the number of distinct elements without keeping track of their frequencies of occurrence.

If the hash table is too small, there will be many collisions which affects the accuracy of our estimate. If it is too big (i.e. of the order of n), then we defeat the purpose of using a hash table in the first place. Ideally, we want to use a large hash table without having to store the entire table.

We begin by presenting a sketch of the algorithm in an ideal world where we map each element of the universe U to a real number between 0 and 1. Note that there are no hash collisions in this ideal world, and the hash table is of infinite size. Let $d := |\{x \in U : \exists t \; with \; i_t = x\}|$ represent the number of distinct elements in the stream. To estimate d, we look at the spread of the entries in our ideal hash table. In particular, we observe that for large d, the ith order statistic of the hash table entries is concentrated around $\frac{i}{d+1}$. For example, the least entry Z in the table is concentrated around $\frac{1}{d+1}$. From this, we can estimate d with $\frac{1}{Z} - 1$.

Sketch of the algorithm

- Pick a hash function h that maps every $x \in U$ to a $U[0, 1]$ (i.e. a uniformly distributed real number between 0 and 1) independently.
- Initialize $Z = 1$.
- At time t, $Z = \min(Z, h(i_t))$.
- Return $\left(\frac{1}{Z} - 1\right)$.

We will prove that Z is an unbiased estimator for $\frac{1}{d+1}$ with small variance.

$$\mathbf{E}[Z] = \int_0^1 \mathbf{Pr}[Z > t]dt = \int_0^1 (1-t)^d dt = \left.\frac{-(1-t)^{d+1}}{d+1}\right|_0^1 = \frac{1}{d+1}.$$

$$\mathbf{E}[Z]^2 = \int_0^1 \mathbf{Pr}\left[Z^2 > t\right]dt = \int_0^1 \mathbf{Pr}\left[Z > \sqrt{t}\right]dt = \int_0^1 (1-\sqrt{t})^d dt = \frac{2}{(d+1)(d+2)} < 2\left(\mathbf{E}[Z]\right)^2.$$

$$\mathrm{Var}[Z] = \mathbf{E}[Z]^2 - (\mathbf{E}[Z])^2 < 2\left(\mathbf{E}[Z]\right)^2 - (\mathbf{E}[Z])^2 = \frac{1}{(d+1)^2}.$$

Note that we have made two unrealistic assumptions for this sketch: (i) It is possible to store real numbers with infinite precision. (ii) h maps every element $x \in U$ to a number between 0 and 1 *independently.*

Now, we consider a more realistic algorithm which uses a hash table of size $M = n^3$. Consider a hash function $h : U \to [M]$. The key idea is to store ℓ order statistics of the stream. Since the d distinct elements are mapped uniformly to a table of size m, the ℓ^{th} order statistic, y_ℓ will approximately be at the location $\frac{\ell M}{d}$. Hence, we can estimate d by $\frac{\ell M}{y_\ell}$. Here is the sketch of the revised algorithm.

Sketch of the algorithm

- Pick a hash function $h : U \to [M]$, whose mapping is pair-wise independent. ($M = n^3$)
- At time t, maintain ℓ smallest distinct hash values $y_1 < \ldots < y_\ell$.
- Return $\frac{\ell M}{y_\ell}$.

Note that we do not store the entire hash table of size $M = n^3$. We only store ℓ entries of the table, each of which would take $3 \log n$ bits for encoding.

Now, we prove that the algorithm gives a good estimate with high probability. Let W_x be an indicator random variable for the event $h(x) \in \left[1, \frac{\ell M}{(1+\epsilon)d}\right]$. Let W be the sum of W_x over all the elements that arrive in the stream. It is easy to see that $\mathbf{E}[W] = \frac{\ell}{1+\epsilon}$. Now, we bound the variance of W.

$$\begin{aligned}\mathrm{Var}(W) &= \mathbf{E}\left[\sum_x W_x^2 + 2\sum_{x,x'} W_x W_{x'}\right] - (\mathbf{E}[W])^2 \\ &= \frac{\ell}{1+\epsilon} + 2\binom{d}{2}\frac{\ell^2}{(1+\epsilon)^2 d^2} - \frac{\ell^2}{(1+\epsilon)^2} \\ &\leq \frac{\ell}{1+\epsilon}.\end{aligned}$$

Here, we made use of the fact that $W_x^2 = W_x$. Now, we use Chebyshev's inequality to get a tail bound on W.

$$\mathbf{Pr}[W \geq \ell] \leq \mathbf{Pr}\big[(W - \mathbf{E}[W])^2 \leq \epsilon^2\ell^2\big] \leq \frac{\ell}{(1+\epsilon)\epsilon^2\ell^2} = \frac{1}{(1+\epsilon)\epsilon^2\ell}.$$

Choosing $\ell := \frac{10}{\epsilon^2}$, we get that $\mathbf{Pr}[W \geq \ell] < \frac{1}{10}$. Similarly, we can define $W' = \sum_x W'_x$ with W'_x being the indicator random variable for the event $h(x) \in \left[1, \frac{\ell M}{(1-\epsilon)d}\right]$ and show that $\mathbf{Pr}[W' \leq \ell] < \frac{1}{10}$, which implies $\mathbf{Pr}\left[Z = \frac{\ell M}{y_\ell} \in (1 \pm \epsilon)d\right] > \frac{4}{5}$.

Note that we have only assumed pairwise independence of the hash function instead of full independence like in our sketch in the ideal world.

To get even tighter concentration, we can apply the median of means strategy from the previous lecture. We repeat the algorithm $\log \frac{1}{\delta}$ times concurrently, with different independent hash functions. In effect, the space complexity of the algorithm is $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n\right)$.

22.2 Dimension reduction

The goal of dimension reduction is to compress the data while preserving certain properties of the data that we want to study. A common property that we like to preserve is the pairwise distance between data points. This has applications in data clustering, nearest neighbor search etc.

In the following analysis, we think of each data point as a vector of features in $\mathbb{R}^n$. We focus on the Euclidean distance between data points, although there is literature on other kinds of distance metrics.

Given n data points $x_1, \dots, x_n \in \mathbb{R}^n$, we want to map (potentially randomly) them to $y_1, \dots, y_n \in \mathbb{R}^k$ such that with probability at least $1 - \delta$, for all $i, j \in [n]$, we have the following.

$$(1-\epsilon)\|x_i - x_j\| \leq \|y_i - y_j\| \leq (1+\epsilon)\|x_i - x_j\|.$$

Here, ϵ indicates the distortion of the mapping.

Now, we provide some intuition on how such a mapping can be constructed using ideas from the tug-of-war estimate for estimating frequencies. Let M_{ij} be a uniform random variable that takes values $\{+1, -1\}$ independently for all $i \in [n]$ and $j \in [k]$. Let $y_j = \frac{1}{\sqrt{k}} \sum_i x_i M_{ij}$.

$$\mathbf{E}\big[y_j^2\big] = \frac{1}{k}\left(\sum_i x_i^2 + 2\sum_{i \neq i'} x_i x'_i \underbrace{\mathbf{E}\big[M_{ij} M_{i'j}\big]}_{=0}\right) = \frac{1}{k}\|x\|^2.$$

Hence, every coordinate of y is an unbiased estimate of $\frac{1}{k}\|x\|^2$. Summing k of these, we get a good estimate for $\|x\|^2$ with low variance. In the next lecture, we will show how to pick the entries M_{ij} so that $O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$ coordinates are sufficient to preseve the distances upto a distortion of ϵ with probability at least $1 - \delta$.