Machine Learning for Large Scale Data Sets

Dr. Dimakis and Dr. Caramanis

NEIL VYAS
EE 381V
The University of Texas at Austin
January 25, 2016

Contents

1 Hashing

1.1 Universal Hashing

We can't ask the question "is this hash function h good?" because for any given $h : [n] \to [m]$, an adversary can find all values in the domain that collide. So we talk about the quality of *hash families*, denoted by $\mathcal{H}$, which are sets of hash functions.

A hash family $\mathcal{H}$ is *2-universal* iff

$$P_h[h(x) = x' \text{ and } h(y) = y'] = \frac{1}{m}^2,$$

for $h \in \mathcal{H}, x, y \in [n], x \neq y$ We say it is k-universal iff the above definition holds for k terms with probability $\frac{1}{m}^k$. A uniform hash function, i.e. a unform hash family, is not a uniform hash function since each member hash function h could hash into a single bucket, where the bucket is chosen uniformly at random. Note that there are m^n hash functions $h : [n] \to [m]$. A *fully random* hash satisfies k-universality for all appropriate k, but needs $\min(\log(m^n), n)$ space.

Keep in mind that for any k-universal hash family, we can recover that it is j-universal for $j < k$ by marginalizing over $k - j$ variables.

A canonical hash family is

$$h(x) = x + b \pmod m,$$

for $b \sim U[0..m)$ Note that this is not a universal hash family, since we can pick x, y to be m apart and they will always hash to the same number, i.e., the chance of collision is 1. So we can slightly modify this to be

$$h(x) = ax + b \pmod m,$$

now for $a, b \sim U[0..m)$. This definition is now 2-universal.

Note that the *only* random thing in our above definition is the choice of hash function $h \in \mathcal{H}$. Even though it's tempting to think of h as random, h is really a deterministic map that we pick at random from $\mathcal{H}$.

1.2 Hashing in Streaming

Let's consider IP addresses. Each IP address has 32 bits, and so there are 2^{32} possible IP addresses (note that there are many less, since IP addresses have some structure). There are a few questions we can ask here:

1. Which IP addresses appear often? *(Heavy Hitters)*
2. How many unique IP addresses appear? *(HyperLogLog)*

Let's consider the last question. We will show that it is impossible to obtain an exact count of unique items with less than $\Theta(m)$ bits of memory using the pigeonhole principle. However, we can estimate this with high probability for any stream with $O(\log m)$ space. For IP addresses, we have $m = 2^{32}$.

Proposition 1.1. *It takes $\Theta(m)$ bits of memory to count exactly the number of unique items in a stream when $n > m$.*

Proof. Suppose we want to use only $m-1$ bits of memory, and say we have seen m items, not necessarily all distinct. How many possible subsets of distinct items could you have seen? It's just the power set of m items, so we should have $2^m - 1$ such subsets, since we've seen at least one item. With $m-1$ bits of memory, we have 2^{m-1} possible states for that memory. Thus, since $2^{m-1} < 2^m - 1$, there must be at least two different subsets of distinct items, say, s_1 and s_2, which map to the same memory configuration, by the pigeonhole principle.

If $|s_1| \neq |s_2|$, then you cannot output the correct number of distinct items after m items, so we stop; let's figure out what happens if $|s_1| = |s_2|$. For the next item we provide, we choose any element from $s_1 - s_2$; this guarantees that $|s_1| \neq |s_2|$. ■

> **Aside:** Defeating Randomization
> Note that in this proof, we depended on the fact that the program was deterministic - between inputs, s_1 and s_2 remained constant, so picking the next element from $s_1 - s_2$ was guaranteed to work. How can we change this proof when the program assigns distinct subsets to memory configurations randomly between inputs?

Let's get some intution for the randomized algorithm. Suppose we have a random stream of n numbers, drawn uniformly at random from $[1..m]$. Can we estimate the number of unique numbers in $\log(M)$ space? If we just consider the minimum, we can find that in $\log(M)$. Let X be the random stream, indexed like X_i. With a little thinking, we find that

$$E[\min(X_i, \ldots, X_s)] \approx \frac{m}{|s|+1} \Rightarrow |s| = \frac{m}{\min X_i} - 1.$$

But this estimator is easily defeated by a stream with a small minimum and a large number of numbers, but a small number of unique values, like $\{1, 5, 5, \ldots\}$. So, we have to first hash the stream. By just remembering the minimum of the hashes, we can get a good estimate in $\log(M)$.

Theorem 1.1. *With probability at least $\frac{2}{3} - \frac{d}{M}$, we have*

$$\frac{d}{6} \leq \frac{M}{\min} \leq 6d.$$

Proof. We have to prove both the upper and lower bounds; let's start with the lower bound. ■

Lemma 1.2.

$$P\left[\frac{M}{\min} > 6d\right] < \frac{1}{6} + \frac{d}{M}.$$

Proof.

$$\begin{aligned} P\left[\frac{M}{\min} > 6d\right] &= P\left[h(b_1) < \frac{M}{6d} \vee \ldots \vee h(b_d) < \frac{M}{6d}\right] \\ &\leq dP\left[h(b_1) < \left\lceil\frac{M}{6d}\right\rceil\right] \qquad \text{(by union bound)} \\ &\leq dP\left[h(b_1) < \frac{M}{6d} + 1\right] \\ &\leq d\left(\frac{M/6d+1}{M}\right) \\ &= \frac{1}{6} + \frac{d}{M} \end{aligned}$$

■

Lemma 1.3.

$$P\left[\frac{M}{\min} < \frac{d}{6}\right] < \frac{1}{6}.$$

Proof.

$$P\left[\min > \frac{6M}{d}\right] = P\left[h(x_1) > \frac{6M}{d} \wedge \ldots \wedge h(x_d) > \frac{6M}{d}\right]$$

So now we've got a bit of a roadblock; we can't union bound here, so we're going to use the second moment method. Let

$$y_i = \begin{cases} 0 & \text{if } h(b_i) > \frac{6M}{d} \\ 1 & \text{otherwise} \end{cases}$$

And let

$$Y = \sum_{i=1} dy_i.$$

So now we want to make sure $Y > 0$ with some large probability. Thus, we find that

$$\begin{aligned} E[Y] &= \sum_{i=1}^{d} E[y_i] \\ &= dP[y_i = 1] \\ &\geq d\left[\frac{6M/d}{M}\right] \\ &= 6. \end{aligned}$$

But we're not done yet (see the aside)! We're going to use *Shepp's second moment method* to show the result, which is as follows:

$$P[Y > 0] \geq \frac{(E[Y])^2}{E[Y^2]}.$$

So we know that

$$E\left[Y^2\right] = \text{Var}[Y] + E[Y]^2,$$

and that the variance of a sum of pairwise-independent random variables (which the y_i are!) is the sum of the variances. We also know that for a $0-1$ random variable, the variance is always less than the expectation.

So let's go for it!

$$\begin{aligned} P[Y > 0] &\geq \frac{E[Y]^2}{\text{Var}[Y] + E[Y]^2} \\ &\geq \frac{E[Y]^2}{E[Y] + E[Y]^2} \\ &= \frac{E[Y]}{1 + E[Y]} \\ &\geq \frac{6}{7}. \qquad \text{(since } \frac{1}{1+x} \text{ is increasing)} \end{aligned}$$

And this is tighter than the result desired. ■

Aside: Pathological Variables
Is showing that $E[y]$ big enough to know that Y is big? No! Consider the following variable:

$$Y = \begin{cases} 0 & \text{w.p. } 1 - \frac{1}{n} \\ 2^{2^n} & \text{w.p. } \frac{1}{n} \end{cases}$$

1.3 Locality Sensitive Hashing (LSH)