CS 7800 / 4810 — Advanced Data Structures

Lecture Notes: Mash & Dashing (HyperLogLog)

---

1 The Hat Problem (Cardinality Estimation)

Setup

Take cards labeled $1, 2, \ldots, 1000$ and choose a *random* subset of size n to hide in a hat. You may see **one representative** card drawn from the hat. The goal is to **estimate** n.

Which representative to pick?

Options: median, minimum, maximum.

- What if the minimum is 500? 10? 4?
- The estimate should *grow* as the minimum *shrinks*.

Minimum-based estimator

If n elements are chosen uniformly at random from $\{1, \ldots, N\}$ (with $N = 1000$), the expected minimum is

$$\mathbb{E}[\min] \approx \frac{N}{n+1}.$$

Inverting this relationship gives the estimator

$$\hat{n} = \frac{N}{\min} - 1.$$

Example 1.1. Suppose $\min = 40$ and $N = 1000$:

$$40 \approx \frac{1000}{n+1} \implies n \approx \frac{1000}{40} - 1 = 24.$$

This estimate is easy to compute and fits in **10 bits**.

2 The Two-Hat Problem (Functions on Sets)

Now there are *two* hats, each holding a random subset of $\{0, \ldots, N\}$ (call them A and B).

- The space of coincidences is large.
- A single representative per hat is insufficient.

Bottom-k sketch

Instead of taking just the minimum, consider the **bottom-k** elements (the k smallest hash values).

- Using bottom-k sketches we can estimate $|A \cap B|$ and $|A \cup B|$.

Partition-based alternative

Instead of bottom-k, take the *minimum in each of k partitions*. This accomplishes something similar to bottom-k with lower merge cost.

3 HyperLogLog (Cardinality Estimation)

Naive solution: use a hash table — requires $\Omega(N)$ space. HyperLogLog uses randomness to achieve sub-linear space.

Hash function and the coin-game intuition

Hash each element to a *uniformly random* binary string:

$$\text{Bob} \xrightarrow{\text{hash}} \underbrace{0101110011}_{\text{random bits}}.$$

Each bit is independently 0 or 1 with probability $\frac{1}{2}$.

Coin game: flip a coin repeatedly; stop on the first tail (T).

- $\Pr(\text{exactly } \ell \text{ heads before a tail}) = 2^{-(\ell+1)}$.
- Every $2^{\ell+1}$ attempts, one sequence will have exactly ℓ heads.

  Equivalently: *if we observe a run of ℓ leading heads, there are probably $2^{\ell+1}$ items in the stream.*

Leading-zeros estimator (LogLog)

Count leading **zeros** in each hash value.

- If the longest run of leading zeros across all hash values is L, the cardinality estimate is 2^{L+1}.
- **Example:** leading zeros $= 2 \;\Rightarrow\; \hat{n} = 2^3 = 8$.

Bit-length of the counter

Let M be the maximum number of unique elements. Since $2^{L+1} \le M \Rightarrow L \le \lg M$, the counter L requires $\lg \lg M$ bits. This is the celebrated **A.C.** (Alon–Matias–Szegedy) space result.

Problems with a single counter

1. Estimates only *powers of 2*.
2. Too much luck involved: if the true count lies between 2^i and 2^{i+1}, a single lucky (or unlucky) hash value dominates.

Solution: multiple approximate counters

Maintain K independent approximate counters $L_1, L_2, \ldots, L_K$ and average their results:

$$\hat{n} = 2^{\frac{L_1+L_2+\cdots+L_K}{K}}.$$

Warning

Averaging the exponents directly can introduce **bias**.

4 HyperLogLog (Full Algorithm)

HyperLogLog improves upon LogLog by replacing the arithmetic mean with the **harmonic mean**, which is less sensitive to large outliers.

Harmonic mean

Definition 4.1 (Harmonic Mean)**.** The *harmonic mean* of n positive numbers $x_1, \ldots, x_n$ is

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

Example 4.1. For the values $1, 4, 4$:

$$H = \left(\frac{1^{-1} + 4^{-1} + 4^{-1}}{3} \right)^{-1} = \frac{3}{1 + \frac{1}{4} + \frac{1}{4}} = \frac{3}{1.5} = 2.$$

Dashing: HyperLogLog pipeline

The four steps of HyperLogLog (as implemented in Dashing) are:

1. K**-partition:** split the hash space into K buckets.
2. $\lfloor \lg_2 N \rfloor$**:** compute leading-zero count (LZC) — a 6-bit value for a 64-bit hash.
3. **Re-exponentiation:** recover the count estimate from the stored value.
4. **Averaging with bias correction:** use the harmonic mean and apply the standard HyperLogLog correction constant.

Pipeline sketch:

$$k\text{-mer} \xrightarrow{\text{hash}} \underbrace{\qquad\qquad}_{\text{64-bit hash}} \xrightarrow{\text{LZC}} \underbrace{\quad}_{\text{6-bit}} \longrightarrow \text{register array (8 bits / entry, 25\% wasted)}$$

Dashing improvement: truncated logarithm

Instead of the 6-bit LZC (leaving 2 bits wasted per register), Dashing stores

$$\lfloor \log_{1.19} \cdot \rfloor \longrightarrow \text{8 bits (full register).}$$

This yields a **better cardinality estimate** by using all available bits.

End of lecture notes.

5 Mash: Bottom-k MinHash Sketch

Mash is a similarity-search tool over genomic sequences. It operates on different types of genomic data:

- Genomes
- Metagenomes
- Amino acids

Definition 5.1 (K-mer)**.** A **k-mer** is a length-k subsequence of a genomic string.

Jaccard similarity via MinHash

Let $S(A)$ denote the bottom-k sketch of set A. The **Jaccard similarity** between sets A and B is

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{\left|S(A \cup B) \cap S(A) \cap S(B)\right|}{|S(A \cup B)|}.$$

Because $S(A \cup B)$ is a random sample of $A \cup B$, the fraction of elements in $S(A \cup B)$ that are shared by both $S(A)$ and $S(B)$ is an *unbiased estimate* of $J(A,B)$.

Example 5.1 (Cardinality walk-through)**.** Let

$$A = \{3,7,8,11,15,17,22,233\}, \quad B = \{2,3,6,7,9,11,17,233\}.$$

With sketch size $k = 8$ and assuming no hash collisions,

$$S(A \cup B) = \{2,3,6,7,8,9,11,15\}.$$

The elements in *both* $S(A)$ and $S(B)$ are $\{3,7,11\}$, so

$$\hat{J}(A,B) = \frac{3}{8} = 0.375.$$

Choosing the right k

The probability that a given k-mer κ appears in a random genome X of size n over alphabet Σ is

$$\Pr[\kappa \in X] = 1 - \left(1 - |\Sigma|^{-k}\right)^n.$$

To ensure a desired probability q_v of a k-mer being present in the genome, set

$$k^* = \left\lceil \log_{|\Sigma|}\left(\frac{n(1-q_v)}{q_v}\right)\right\rceil.$$

Error bound on the Jaccard estimate

The error on the MinHash Jaccard estimate is

$$\varepsilon = O\left(\frac{1}{\sqrt{s}}\right),$$

where s is the sketch size.

Mash p-value

Question: How many k-mers are expected to match between the sketches of two *unrelated* genomes?

Relevant parameters:

- Sketch size s.
- $\Pr[\kappa \in \text{genome}]$, the probability of a k-mer appearing in a genome.

The expected Jaccard index λ between two independent genomes X, Y satisfies

$$\lambda = \frac{P(\kappa \in X)\, P(\kappa \in Y)}{P(\kappa \in X) + P(\kappa \in Y) - P(\kappa \in X)\, P(\kappa \in Y)}.$$

6 Proofs and Analyses:

7 HyperLogLog: Analysis and Proofs

7.1 Setup

We hash each element x to an infinite binary string:

$$h(x) \in \{0,1\}^\infty$$

Define:

$$\rho(x) = \text{position of the first 1 in } h(x)$$

Example:

$$h(x) = 000100\cdots \Rightarrow \rho(x) = 4$$

7.2 Key Observation

For a uniformly random hash:

$$\mathbb{P}[\rho(x) = k] = 2^{-k}$$

Proof:

- First $k-1$ bits are 0: probability $2^{-(k-1)}$
- k-th bit is 1: probability $1/2$

Thus:

$$\mathbb{P}[\rho(x) = k] = 2^{-(k-1)} \cdot \frac{1}{2} = 2^{-k}$$

7.3 Single Register Estimator

Let:

$$R = \max_{x \in S} \rho(x)$$

7.4 Distribution of R

For $n = |S|$:

$$\mathbb{P}[R < k] = \mathbb{P}[\rho(x) < k \ \forall x]$$

Since elements are independent:

$$\mathbb{P}[R < k] = (1 - 2^{-k})^n$$

Thus:

$$\mathbb{P}[R \geq k] = 1 - (1 - 2^{-k})^n$$

7.5 Intuition for Estimator

We expect:

$$2^R \approx n$$

Heuristic:

$$\mathbb{P}[\rho(x) \geq k] = 2^{-(k-1)}$$

Expected number of elements with $\rho(x) \geq k$:

$$n \cdot 2^{-(k-1)}$$

Set this to ≈ 1:

$$n \cdot 2^{-(k-1)} \approx 1 \Rightarrow k \approx \log_2 n$$

Thus:

$$R \approx \log_2 n \Rightarrow n \approx 2^R$$

7.6 Bias of 2^R

The estimator 2^R is biased due to high variance:

- Maximum is unstable
- Dominated by rare events

This motivates averaging across multiple registers.

7.7 HyperLogLog Algorithm

- Partition hash space into $m = 2^b$ buckets using first b bits
- For each bucket j, compute:

$$R_j = \max \rho(x) \text{ over elements mapped to bucket } j$$

Final estimator:

$$\hat{n} = \alpha_m \cdot m^2 \left(\sum_{j=1}^{m} 2^{-R_j} \right)^{-1}$$

where α_m is a bias-correction constant.

7.8 Why the Harmonic Mean?

We use:

$$Z = \sum_{j=1}^{m} 2^{-R_j}$$

Then:

$$\hat{n} \propto \frac{m^2}{Z}$$

Key idea:

- 2^{R_j} is a noisy estimator of bucket size
- Using 2^{-R_j} and averaging gives stability
- This corresponds to a harmonic mean

7.9 Expectation of 2^{-R_j}

Let bucket j contain n_j elements.

One can show:

$$\mathbb{E}[2^{-R_j}] \approx \frac{1}{n_j}$$

Thus:

$$\mathbb{E}[Z] \approx \sum_{j=1}^{m} \frac{1}{n_j}$$

If elements are evenly distributed:

$$n_j \approx \frac{n}{m} \Rightarrow Z \approx \frac{m^2}{n}$$

Rearranging:

$$n \approx \frac{m^2}{Z}$$

7.10 Variance

One can show:

$$\mathrm{Var}(\hat{n}) \approx \frac{1.04^2}{m} \cdot n^2$$

Thus relative error:

$$\frac{\sqrt{\mathrm{Var}(\hat{n})}}{n} \approx \frac{1.04}{\sqrt{m}}$$

Key takeaway:

$$\text{Error} = \Theta\left(\frac{1}{\sqrt{m}}\right)$$

7.11 Bias Correction

The constant:

$$\alpha_m = \begin{cases} 0.673 & m = 16 \\ 0.697 & m = 32 \\ 0.709 & m = 64 \\ 1.03896/(1 + 1.079/m) & m \geq 128 \end{cases}$$

corrects systematic bias from approximation.

7.12 Small and Large Range Corrections

- **Small n:** many empty buckets $\Rightarrow$ use linear counting
- **Large n:** hash collisions $\Rightarrow$ apply correction

7.13 Summary

- Uses probabilistic counting via leading zeros
- Converts max-statistic into stable harmonic mean
- Achieves:

$$O(\log \log n) \text{ space}, \quad O(1/\sqrt{m}) \text{ error}$$

8 Analysis of Bottom-k Sketches

We assume hash values are drawn i.i.d. from Uniform$(0, 1)$.

Let S be a set of size n, and let R_k denote the k-th smallest hash value.

8.1 Distribution of R_k

Let $U_1, \ldots, U_n \sim \text{Uniform}(0, 1)$ i.i.d., and let

$$R_k = k\text{-th order statistic of } \{U_1, \ldots, U_n\}.$$

Then R_k has a Beta distribution:

$$R_k \sim \text{Beta}(k, n - k + 1).$$

In particular:

$$\mathbb{E}[R_k] = \frac{k}{n+1}.$$

8.2 Cardinality Estimator

We use:

$$\hat{n} = \frac{k}{R_k}.$$

8.3 Bias Analysis

We compute:

$$\mathbb{E}\left[\frac{1}{R_k}\right].$$

Using properties of the Beta distribution:

$$\mathbb{E}\left[\frac{1}{R_k}\right] = \frac{n}{k-1}, \quad \text{for } k \geq 2.$$

Thus:

$$\mathbb{E}[\hat{n}] = k \cdot \frac{n}{k-1} = \frac{k}{k-1} n.$$

Conclusion: $\hat{n}$ is slightly biased upward.

8.4 Unbiased Estimator

Define:

$$\tilde{n} = \frac{k-1}{R_k}.$$

Then:

$$\mathbb{E}[\tilde{n}] = n.$$

8.5 Variance

Using known moments of the Beta distribution:

$$\mathrm{Var}(R_k) = \frac{k(n-k+1)}{(n+1)^2(n+2)}.$$

One can show (via Delta method or direct calculation):

$$\mathrm{Var}(\tilde{n}) = \Theta\left(\frac{n^2}{k}\right).$$

Thus the relative error is:

$$\frac{\sqrt{\mathrm{Var}(\tilde{n})}}{n} = \Theta\left(\frac{1}{\sqrt{k}}\right).$$

Interpretation: Increasing k reduces error as $1/\sqrt{k}$.

8.6 Jaccard Similarity Estimator

Let:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Let $S(A \cup B)$ be the bottom-k sketch of the union.
Define indicator variables:

$$X_i = \begin{cases} 1 & \text{if the } i\text{-th sketch element belongs to both } A \text{ and } B \\ 0 & \text{otherwise} \end{cases}$$

Estimator:

$$\hat{J} = \frac{1}{k} \sum_{i=1}^{k} X_i.$$

8.7 Unbiasedness

Each of the k sampled elements is a uniform random sample from $A \cup B$.

Thus:

$$\mathbb{P}[X_i = 1] = \frac{|A \cap B|}{|A \cup B|} = J(A, B).$$

Hence:

$$\mathbb{E}[\hat{J}] = J(A, B).$$

8.8 Variance

Since X_i are (approximately) independent Bernoulli variables:

$$\mathrm{Var}(\hat{J}) = \frac{J(1-J)}{k}.$$

Thus:

$$\text{Std. dev.} = O\left(\frac{1}{\sqrt{k}}\right).$$

8.9 Intersection Estimation

We estimate:

$$|A \cap B| = J(A, B) \cdot |A \cup B|.$$

Let $\hat{U}$ be the estimate of $|A \cup B|$.

Define:

$$\widehat{|A \cap B|} = \hat{J} \cdot \hat{U}.$$

8.10 Remarks on Independence

- The sketch elements are not fully independent, but are negatively correlated
- This improves concentration bounds in practice
- Analysis often assumes independence for simplicity

8.11 Concentration Bounds (Sketch)

Using Chernoff bounds:

$$\Pr\left[|\hat{J} - J| > \varepsilon\right] \leq 2\exp(-\Theta(k\varepsilon^2)).$$

Thus, to achieve error ε with probability $1 - \delta$:

$$k = O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right).$$