COMP 480/580 — Probabilistic Algorithms and Data Structure Aug 29, 2023

Lecture 3: Estimation and Hashing

Lecturer: Anshumali Shrivastava *Scribe By: Wesley Yang*

1 Mark and Recapture Estimation

Problem: How do we estimate the number of turtles in a pond?

- Let n be the total number of turtles. We can capture k_1 turtles, mark them, and release them back into the pond. Assuming they mix evenly, we can then recapture k_2 turtles, M of which are marked, and set up the following equations:

$$\frac{k_1}{n} \simeq \frac{M}{k_2}$$

$$\hat{n} = \frac{k_1 k_2}{M}$$

Is there a more disciplined approach to show that $\hat{n}$ is a good estimator?

- Set up n indicator random variables $I_1, I_2, ..., I_n$ where I_j is 0 or 1 depending on whether turtle j is marked. Then:

$$I_j = \begin{cases} 1 \text{ with probability } k_1/n \\ 0 \text{ with probability } 1 - (k_1/n) \end{cases}$$

$$P(I_j = 1) = E[I_j] = \frac{k_1}{n}$$

- We can write the random variable of interest, M, as a summation:

$$M = \sum_{j=1}^{k_2} I_j$$

- By Linearity of Expectation:

$$E[M] = \sum_{j=1}^{k_2} E[I_j] = \frac{k_1 k_2}{n}$$

$$n = \frac{k_1 k_2}{E[M]}$$

- So the $\hat{n}$ we derived earlier through intuition is not an unbiased estimator:

$$E[\hat{n}] = E\left[\frac{1}{M}\right] k_1 k_2$$

2 Families of Hash Functions

- A hash function maps objects to a discrete range from 0 to R: $h(O) \in [0, 1, ..., R]$.
- A perfect hash function guarantees that if $O_1 \neq O_2$, then $h(O_1) \neq h(O_2)$. However, unless the number of possible objects is very small, no feasible function exists. We could store every single object, but that would defeat the purpose of using a hash table in the first place. We need to relax the constraint and allow for some collisions.
- An n-universal family of hash functions H has the following property for all $h \sim H$:

$$P(h(O_1) = h(O_2) = ... = h(O_n) \mid O_1 \neq O_2 \neq ... \neq O_n) \leq \frac{1}{R^{(n-1)}}$$

- Consider a 2-universal hash function. What's the probability of rolling 2 dice and getting the same number on both? Assuming the output of the hash function h is truly random (pseudorandom) and uniformly distributed across the range R, then:

$$P(h(O_1) = h(O_2) \mid O_1 \neq O_2) = \frac{1}{R}$$

- Examples of 2- and 3-universal hash functions:

$$h(x) = ((ax + b) \bmod P) \bmod R$$

$$h(x) = ((ax^2 + bx + c) \bmod P) \bmod R$$

- where P is some large prime chosen such that $P > R$; R is the final range (desired number of buckets to map to); a, b, c, ... are parameters that define a specific hash function within the given family H and are chosen randomly and uniformly from the range $[1, P - 1]$.
- Higher-degree polynomials may be used for larger values of n and thus stronger independence guarantees, with the trade-off being higher computational complexity.
- How is h sampled from H? Since a given hash function is defined by its parameters P, a, b, ..., simply generate them in advance and store them.
- Recall the Principle of Deferred Decision: There is no difference between pre-generating a sequence of values and generating them on-demand.