Hashing, Episode 3:

Approximate counting and searching via hashing

Rasmus Pagh
IT University of Copenhagen

MADALGO SUMMER SCHOOL ON DATA STRUCTURES 2013

Today: Hash and forget

- Approximate counting by consistent sampling.
- Min-wise hashing.
- Teaser: Locality-sensitive hashing.

Approximate counting

- Suppose the human race decided to count its number of members.
- Each human throws 6 dice. With probability ~10^{-7} a person throws 6x⚅.
- If k people report they threw only ⚅s, we estimate the population as $10^7\,k$.

Counting with a hash function

- Now suppose we want to count first names - each name only once!
- Idea: "Consistent sampling"
  - Hash name x to a value $h(x)$ in [0;1].
  - Report any x with $h(x) \leq p$.
  - Estimate is p^{-1} times #distinct reports.

Variance of consistent sampling

- Suppose we want to estimate $|S| = c$, and include x in the sample with probability p.
- Let X denote the size of the sample.
- X/p is an unbiased estimator: $E[X/p] = c$.
- **Lemma**:
  If h is 2-wise independent, $\mathrm{Var}(X/p) \leq c/p$.

Error bound (example)

- $\Pr[X/p > 2c] < \mathrm{Var}(X)/c^2 \leq (cp)^{-1}$.
- So: Good bound whp. if cp is big enough.
- But how do we choose a suitable p without knowing c!?

Min-wise hashing

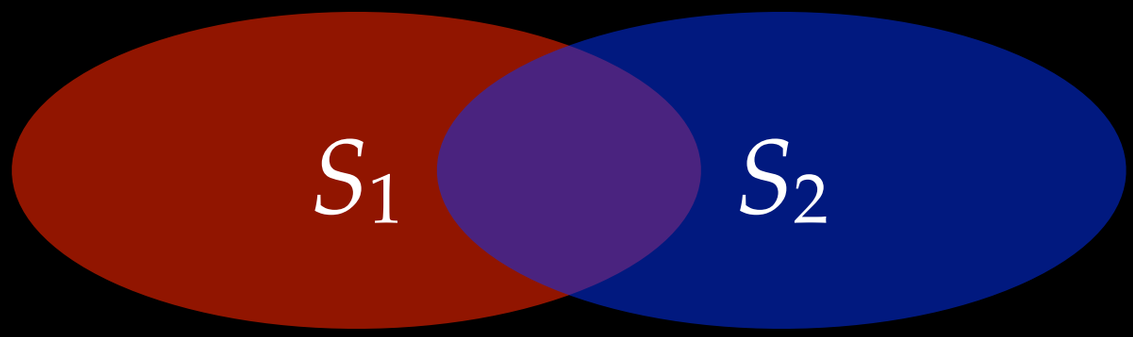
- Idea: Adjust sampling rate p to keep a fixed sample size k.
- Possible implementations:
  - Store keys with the k smallest hash values.
  - Store x with smallest hash value for $h_1,\ldots,h_k$.
- If kth smallest hash value is p, estimate that there are k/p distinct keys.

Min-wise hashing analysis

- Suppose the exact number of keys is c.
- Let $L_\alpha = |\{x \in S \mid h(x) < \alpha\}|$
- **Lemma**. If $L_{0.9k/c} < k$ and $L_{1.1k/c} \geq k$ then the estimate k/p is between $0.9c$ and $1.1c$.
- **Proof**: $\mathrm{E}[L_{0.9k/c}]=0.9k$ & tail bound using that h is pairwise independent implies $L_{0.9k/c} < k$ holds whp. Same for $L_{1.1k/c}$.

Combining min-wise hashes I

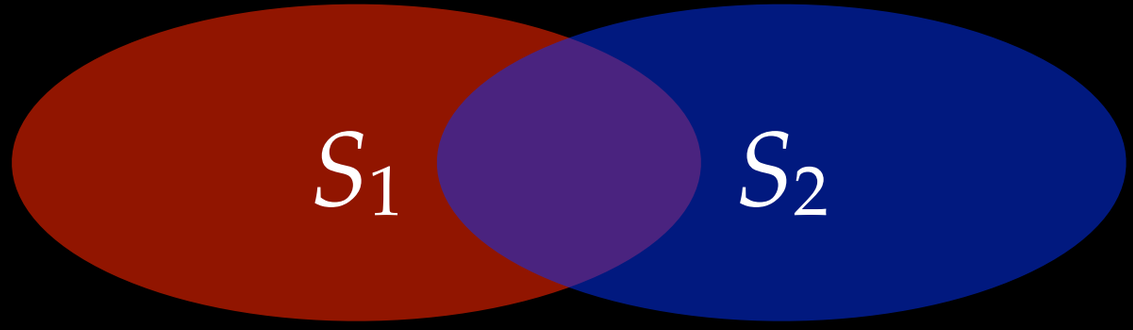
- Let A_1=minhash(S_1), A_2=minhash(S_2).
- What do A_1 and A_2 tell us about S_1 and S_2?



- Can be used to form minhash($S_1 \cup S_2$), and thus estimate $|S_1 \cup S_2|$.

Combining min-wise hashes II

- Let A_1=minhash(S_1), A_2=minhash(S_2).
- What do A_1 and A_2 tell us about S_1 and S_2?



- E[| minhash($S_1 \cup S_2$) $\cap$ minhash(S_1) $\cap$ minhash(S_2) |] $= k \, | S_1 \cap S_2 | \, / \, | S_1 \cup S_2 |$.

Thorup, STOC '13: 2-wise independence enough for concentration

Locality-sensitive hashing

- Hashing: Map keys as randomly as possible.
- LSH: Map "similar" keys to similar values, but avoid collisions of "not so similar" keys.
- Example LSH:
  - Hash a bit string x by sampling b of its bits.
  - Repeat many times to get collision for similar keys.

High-dimensional similarity search

- Typically allow approximation factor *c*: Looking for a point at distance *d* from *x*, we accept points at distance *cd* being reported.

- State-of-the-art solutions either:
  - Use space around n^{1+1/c^2}, or
  - Use query time ~ $n^{2/c}$, only sublinear for large enough *c*.

High-dimensional similarity search

- Commercial break:
  I'm looking for X PhD students and X post-docs starting in 2014+ for a project on scalable similarity search at IT University of Copenhagen.
- Here E[X]≈1.5, Var[X]>2.

Some references

- Broder: On the resemblance and containment of documents
http://www.cs.princeton.edu/courses/archive/spring05/cos598E/bib/broder97resemblance.pdf

- Bar-Yossef et al.: Counting Distinct Elements in a Data Stream
http://www.cs.umd.edu/~samir/498/distinct.ps

- König and Li: Theory and Applications of b-Bit Minwise Hashing
http://research.microsoft.com/pubs/152334/CACM_hashing.pdf

- Cohen: Size-estimation framework with applications to transitive closure and reachability.
http://www.cs.washington.edu/education/courses/cse521/05wi/papers/cohen-size-estimation.ps

- Thorup: Bottom-k and Priority Sampling, Set Similarity and Subset Sums with Minimal Independence
http://arxiv.org/pdf/1303.5479v2.pdf

- Backstrom et al.: Four degrees of separation
http://people.cam.cornell.edu/~jugander/papers/websci12-fourdegrees.pdf

- Andoni and Indyk: Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions
http://people.csail.mit.edu/indyk/p117-andoni.pdf

Nice stuff I did not cover

(Incomplete list, obviously.)

- Tabulation hashing (papers by Patrascu and Thorup).
- Dictionaries where each operation is O(1) time whp. (Arbitman et al.)
- Simulating full independence (Pagh[2]).
- Dynamic approximate membership: Upper and lower bounds (Lovett and Porat; Pagh et al.).
- Simple perfect hashing (Botelho et al.)