Distributed Private Heavy Hitters

Justin Hsu, Sanjeev Khanna, Aaron Roth

University of Pennsylvania

July 11, 2012

A motivating problem: Website referrals

A popular website wants to know who the top referrer is.

A motivating problem: Website referrals

A popular website wants to know who the top referrer is.

Each user knows where he arrived from, but he doesn't want to make this information public (may be embarrassing)

How to protect privacy?

Differential Privacy

- Rigorous, well-studied notion of privacy, first proposed by Dwork, McSherry, Nissim, Smith (2006)
- Provides guarantees of how a *single* record influences the output of a mechanism
- Laplace mechanism: add noise to protect privacy

How to protect privacy?

Differential Privacy

- Rigorous, well-studied notion of privacy, first proposed by Dwork, McSherry, Nissim, Smith (2006)
- Provides guarantees of how a *single* record influences the output of a mechanism
- Laplace mechanism: add noise to protect privacy

Definition

A mechanism M is ϵ-*differentially private* if for databases $\mathcal{D}, \mathcal{D}'$ which differ in a single record, and for r any output,

$$\frac{\Pr[M(\mathcal{D}) = r]}{\Pr[M(\mathcal{D}') = r]} \leq e^{\epsilon}$$

Database Location

Centralized vs. Distributed

- Usually, unprotected database located with a central party
- What if there is no trusted party?
- What algorithms can we give for the fully distributed setting?

Database Location

Centralized vs. Distributed

- Usually, unprotected database located with a central party
- What if there is no trusted party?
- What algorithms can we give for the fully distributed setting?

Prior work

- Kasiviswanathan, Lee, Naor, et al. (2008) studied the fully distributed model in the context of learning
- McGregor, et al. (2008), studied the two database case
- Dwork, Naor, Pitassi, et al. (2009) studied heavy hitters in pan-private setting

The Heavy Hitters problem

Problem Statement

- Collection of users, each with a private universe element
- Goal: release the most popular element (the *heavy hitter*)

The Heavy Hitters problem

Problem Statement

- Collection of users, each with a private universe element
- Goal: release the most popular element (the *heavy hitter*)

Local Privacy Model

- No central authority has access to all the clean data
- Mechanism must query each user *individually* and return a universe element
- Each query must be differentially private

The Heavy Hitters problem

Problem Statement

- Collection of users, each with a private universe element
- Goal: release the most popular element (the *heavy hitter*)

Local Privacy Model

- No central authority has access to all the clean data
- Mechanism must query each user *individually* and return a universe element
- Each query must be differentially private

Questions:

- What kind of accuracy is possible?
- Efficient algorithms?

Accuracy and Efficiency

α-Accuracy

- If mechanism M returns an element whose frequency differs from the heavy hitter's frequency by at most additive α, we say M is *α-accurate*

Accuracy and Efficiency

α-Accuracy

- If mechanism M returns an element whose frequency differs from the heavy hitter's frequency by at most additive α, we say M is *α-accurate*

Efficiency

- Notation: m number of users, N size of universe
- Consider N to be very large (number of websites on internet)
- Consider algorithm to be *efficient* if running time is $\text{poly}(m, \log N)$

Information theoretic results

Theorem (Lower bound)

There is no differentially private mechanism that achieves $\sqrt{m}$-accuracy for the heavy hitters problem with high probability, in the local model.

Theorem (Upper bound)

There is a differentially private algorithm that achieves $O(\sqrt{m \log N})$-accuracy for the heavy hitters problem with high probability, in the local model.

Lower bound on error

Theorem (Lower bound)

There is no differentially private mechanism that achieves $\sqrt{m}$-accuracy for the heavy hitters problem with high probability on the heavy hitters problem, in the local model.

Proof sketch

- Universe size $N = 2$, with users' data drawn from a uniform distribution
- By differential privacy, belief about private data is approximately uniform given query answers
- By anti-concentration, mechanism can't do better than $\sqrt{m}$ error with high probability

Lower bound on error

Comparison with centralized setting

- In centralized setting, can get $O(\log N)$-accuracy (exponential mechanism)
- $\Omega(\sqrt{m})$ error is *unavoidable* cost of moving to fully distributed setting

Near-optimal accuracy algorithm: JL-HH

Near-optimal accuracy algorithm: JL-HH

Lemma (Johnson-Lindenstrauss)

For any set S of p points in $\mathbb{R}^w$, there is a linear map $A : \mathbb{R}^w \to \mathbb{R}^z$, where $z = O(\log(p)/\alpha^2)$, such that inner products are approximately preserved: For any two points $u, v \in S$,

$$|\langle u, v\rangle - \langle Au, Av\rangle| \leq \alpha(\|u\|^2 + \|v\|^2)$$

Near-optimal accuracy algorithm: JL-HH

Lemma (Johnson-Lindenstrauss)

For any set S of p points in $\mathbb{R}^w$, there is a linear map $A : \mathbb{R}^w \to \mathbb{R}^z$, where $z = O(\log(p)/\alpha^2)$, such that inner products are approximately preserved: For any two points $u, v \in S$,

$$|\langle u, v\rangle - \langle Au, Av\rangle| \leq \alpha(\|u\|^2 + \|v\|^2)$$

Notation

- Private histogram $v \in \mathbb{N}^N$, each i'th index contains count of element i
- Each user has histogram $u^i \in \mathbb{N}^N$, and $v = \sum_i u^i$
- Goal: return $\text{argmax}_i v_i$

Near-optimal accuracy algorithm: JL-HH

JL-HH sketch

- *Count* of j'th element is $\langle v, e_j \rangle$, with e_j standard basis vector
- Estimate this by $\langle Av, Ae_j \rangle$
- Estimate Av by summing $Au^i + \eta^i$ over all *users* i
- $\eta = \sum_i \eta^i$ noise to protect differential privacy
- For each universe element j, compute $\langle Av + \eta, Ae_j \rangle$
- Return element with largest estimated count

Near-optimal accuracy algorithm: JL-HH

JL-HH sketch

- *Count* of j'th element is $\langle v, e_j \rangle$, with e_j standard basis vector
- Estimate this by $\langle Av, Ae_j \rangle$
- Estimate Av by summing $Au^i + \eta^i$ over all *users* i
- $\eta = \sum_i \eta^i$ noise to protect differential privacy
- For each universe element j, compute $\langle Av + \eta, Ae_j \rangle$
- Return element with largest estimated count

Accuracy, efficiency, and privacy

- Each user in JL-HH interacts in a differentially private way with the algorithm.
- $O(\sqrt{m \log N})$-accurate for heavy hitters problem
- Requires iterating over all N universe elements, *not efficient*

Two incomparable, efficient algorithms

Theorem (GLPS-HH Algorithm)

There is a differentially private, efficient algorithm that achieves $O(m^{5/6})$-accuracy for the heavy hitters problem.

Theorem (Bucket Algorithm)

There is a differentially private, efficient algorithm that calculates the true heavy hitter with high probability, as long as the count of the heavy hitter dominates the l_2 norm of the other elements.

First efficient algorithm: GLPS-HH

GLPS Algorithm

- Gilbert, et al. (2009) give a sophisticated compressed sensing algorithm
- Similar idea as JL-HH: linear projection to lower dimensional space, add noise, then reconstruct the original histogram
- More technical decoding step to estimate histogram efficiently
- Runs in time $O(m \log^c N)$

First efficient algorithm: GLPS-HH

GLPS Algorithm

- Gilbert, et al. (2009) give a sophisticated compressed sensing algorithm
- Similar idea as JL-HH: linear projection to lower dimensional space, add noise, then reconstruct the original histogram
- More technical decoding step to estimate histogram efficiently
- Runs in time $O(m \log^c N)$

Theorem (Accuracy of GLPS-HH)

GLPS-HH is α-accurate for $\alpha = O(m^{5/6} \log^2 N)$ with probability at least 3/4. The failure probability can be driven down by iteration.

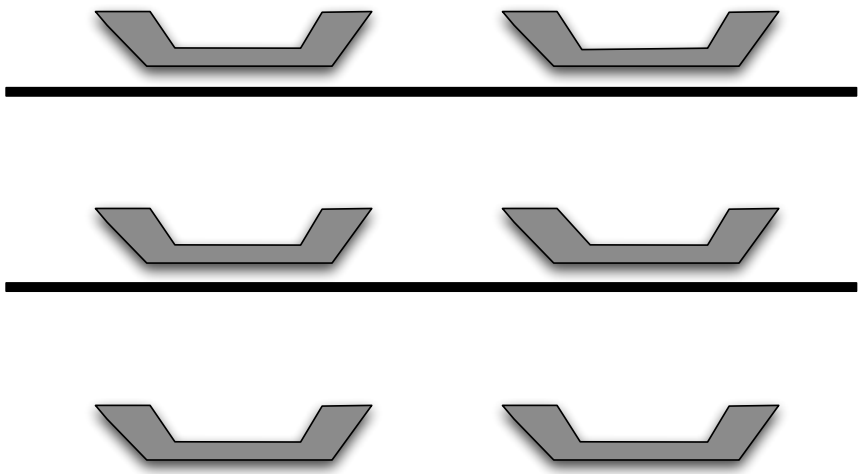
Second efficient algorithm: Bucket algorithm

Sketch of Bucket algorithm

- Take $\log N$ random hash functions, and hash each user's data into one of two buckets.
- Total up noisy counts in each bucket, select the unique element that is hashed into the larger bucket by each hash function, if it exists.
- Run this procedure $\log N$ rounds, and take a majority vote to find the heavy hitter

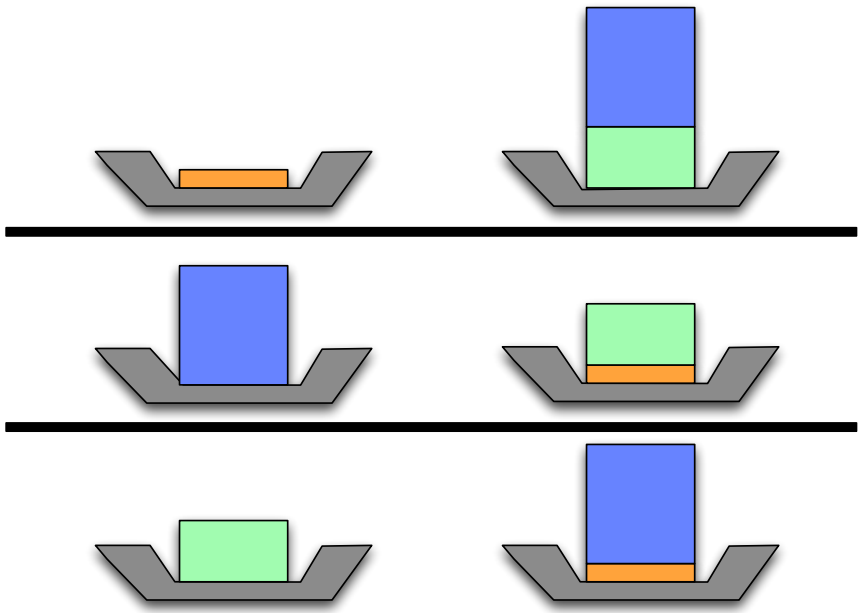
Bucket algorithm, in pictures

- Step 1: Select $\log N$ random 0/1 hash functions



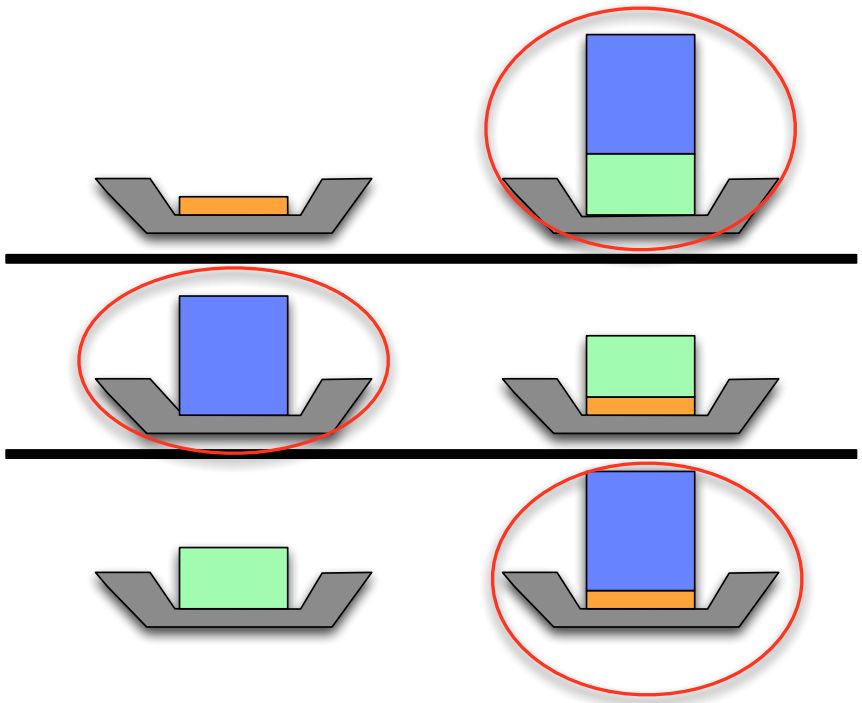
Bucket algorithm, in pictures

- Step 1: Select $\log N$ random 0/1 hash functions
- Step 2: Hash user data into the buckets for each trial



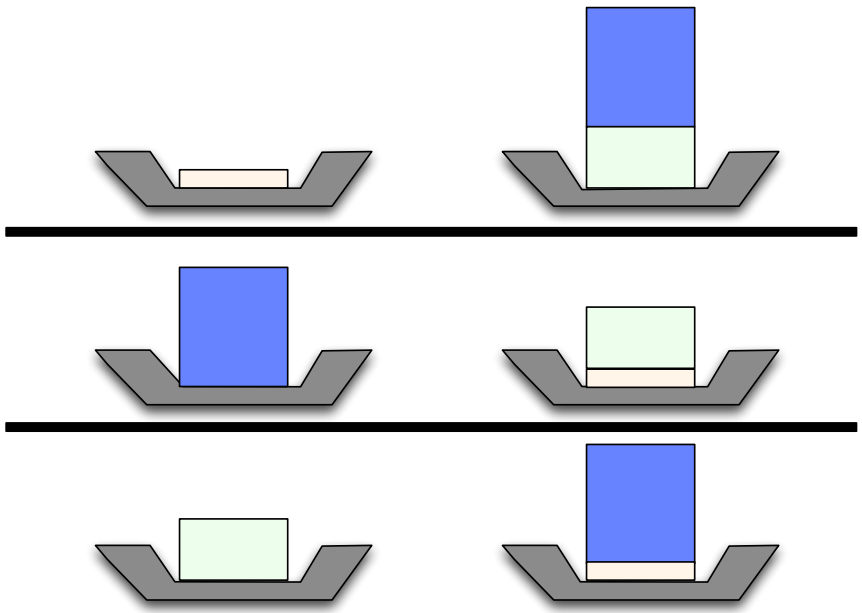
Bucket algorithm, in pictures

- Step 1: Select $\log N$ random 0/1 hash functions
- Step 2: Hash user data into the buckets for each trial
- Step 3: Total up noisy counts to find majority bucket



Bucket algorithm, in pictures

- Step 1: Select $\log N$ random 0/1 hash functions
- Step 2: Hash user data into the buckets for each trial
- Step 3: Total up noisy counts to find majority bucket
- Step 4: Select element that hashes into majority bucket for each trial



Bucket algorithm: performance and runtime

Bucket algorithm: performance and runtime

Accuracy

- Guarantee: if heavy hitter has count greater than the l_2-norm of rest of histogram, algorithm will return true heavy hitter
- No guarantee if condition is not met

Bucket algorithm: performance and runtime

Accuracy

- Guarantee: if heavy hitter has count greater than the l_2-norm of rest of histogram, algorithm will return true heavy hitter
- No guarantee if condition is not met

Privacy and running time

- Bucket algorithm is differentially private
- Pairwise independent hash functions suffice, linear hash functions
- Finding element that hashes into all the larger buckets is fast: system of $O(\log N)$ linear equations
- Run time $O(m \log^3 N)$, efficient

Wrapping up

Open problems

- Are there algorithms that achieve optimal accuracy?
- What is the best that can be done efficiently (poly($m, \log N$) time)?
- What other problems in the distributed setting can be tackled with this approach?

Link to paper

Available on arXiv, `http://arxiv.org/abs/1202.4910`

Distributed Private Heavy Hitters

Justin Hsu, Sanjeev Khanna, Aaron Roth

University of Pennsylvania

July 11, 2012