A Distributed Algorithm For Differentially Private Heavy Hitters

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper we present a simple *distributed* algorithm for learning the heavy hitters in a stream of data in the local model of differential privacy. Our algorithm is efficient in the sense that its running time and communication complexity is polynomial in the input size. Furthermore, our algorithm achieves the optimal error guarantee for the frequency estimation.

1 Introduction

One of the most widely performed learning tasks on telemetry data by software companies is to understand *heavy hitters or frequent items*. Typical applications of such a data analysis include understanding the most frequently visited websites (URLs), popular apps on an app-store, trending news, etc. In order to meet users' privacy expectations and in view of tightening privacy regulations (European GDPR law) the ability to collect telemetry data privately is paramount. In this paper we are interested in algorithms that preserve user's privacy in the face of such data collection from users.

Over the past decade, Differential privacy, introduced in the seminal work of Dwork *et al* [12] (DP), has emerged as the standard definition of privacy for data analysis and machine learning, with wide acceptance both in academia and industry. In the context of telemetry collection one typically considers algorithms that exhibit a stronger notion of differential privacy called *local differential privacy* [14, 18, 16, 10, 5], also known as randomized response model [21], γ-amplification [15], or FRAPP [1]. These are randomized algorithms that are invoked on *user's device* to turn user's private value into a response that is communicated to data collector and have the property that the likelihood of any specific algorithm's output varies little with the input, thus providing users with plausible deniability. Formally, *given a constant $\epsilon > 0$, a randomized algorithm $\mathcal{A} : \mathcal{V} \to \mathcal{Z}$ is ϵ-locally differentially private (ϵ-LDP) if for any pair of values $v, v' \in \mathcal{V}$ and any subset of output $S \subseteq \mathcal{Z}$, we have that* $\mathbf{Pr}[\mathcal{A}(v) \in S] \leq e^{\epsilon} \cdot \mathbf{Pr}[\mathcal{A}(v') \in S]$.

In the heavy hitters problem we are given a set of n users; each user holds an item v_i, which we assume without loss of generality is an element in $\{1, 2, \ldots d\}$. A histogram of the data lists for each item $v \in [d]$ the fraction of population that holds item v. However, when d is large compared to n, as is the case in our example of understanding most frequent URLs, an explicit representation of histogram is neither possible nor necessary. Often it is sufficient to have an implicit representation of the histogram. Two commonly used representations are: a) *Frequency oracle*, which is an algorithm that answers the frequency of a queried item $v \in [d]$. The frequency of an item v is defined as $f(v) = \frac{1}{n} \cdot |\{i | v_i = v\}|$. b) *Heavy hitters*, which returns a list of k most frequent items. An algorithm for heavy items returns a list of k items $\hat{v}_1, \hat{v}_2, \ldots \hat{v}_k$ together with the estimated frequencies $\hat{f}(\hat{v}_1), \hat{f}(\hat{v}_2), \ldots \hat{f}(\hat{v}_k)$. *The frequencies of items not in the list are assumed to be zero.* The error of a heavy hitter algorithm for item v is defined to be $|\hat{f}(v) - f(v)|$. The error of heavy hitter algorithm is the maximum error over all the items $v \in [d]$. That is, Error $= \max_{v \in [d]} |\hat{f}(v) - f(v)|$.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

The heavy hitters problem in the context of local differential privacy was made popular in the influential paper of Erlingsson, Pihur, and Korolova [14]. They gave differentially private heuristics, without any rigorous error analysis, for the heavy hitters problem that formed the basis of its implementation in Chrome browser. Later, Apple used differentially private heavy hitters algorithms to understand most frequently used emojis and to discover new words in text data [20].

The theoretical study of the problem, however, dates back to [17, 19], who gave private algorithms for heavy hitters with error $O(\frac{1}{n^{1/6}})$. The algorithms of [17] only satisfied (ϵ, δ)-differential privacy for $\delta > 0$, which is a strictly weaker notion of differential privacy. In a breakthrough work Bassily and Smith [4] constructed *optimal* algorithms for differentially private heavy hitters problem. They showed that there exist efficient mechanisms for both frequency oracle and heavy hitters with an error $O\left(\sqrt{\log d/n}\right)$.

Despite its theoretical optimality on the error, the algorithm in [4] is not very suitable for use in practice. The client side running time of the algorithm is $O(n^{1.5})$, which is prohibitively expensive in applications where the number of users is in the of order of hundreds of millions. This limitation was addressed in a very recent paper [2], who gave two elegant algorithms, ***TreeHist*** and ***Bitstogram***, which dramatically reduced the running time on the client side to $O(\max\{\log n, \log d\}^2)$. Both these algorithms rely on first solving the frequency oracle problem, and then using it as a subroutine to solve the heavy hitters problem. The Bitstogram algorithm achieves the optimal guarantees on the error, whereas the TreeHist algorithm is sub-optimal by a factor of $\sqrt{\log n}$ for the heavy hitters problem.

All the known algorithms for the heavy hitters [4, 2] problem rely on users having access to *shared random bits*. In the TreeHist algorithm every user needs access to $O(\log n)$ number of shared random hash functions from a universal family of hash functions. On the other hand, Bitstogram algorithm requires that users have access to a shared random $\{-1, 1\}$-matrix of size $O(\sqrt{n} \times n)$. The most secure way (and sometimes the only way) to implement shared randomness is that the *server generates random bits and sends it to each client*. For Bitstogram algorithm, this would require transmitting $\Omega(\sqrt{n})$ number of bits for each round of data collection. (Note that a user does not need the entire matrix but only a single column.) This is too expensive as telemetry data is collected for many applications, and downloading $\Omega(\sqrt{n})$ number of bits when n is in order of hundreds of millions is just not feasible. On the other hand, TreeHist protocol needs access to $O(\log n)$ bits, and hence is more amenable in practice. Yet, both these schemes require that *all the users have same shared random bits*.

Implementing shared randomness, *however small it may be*, poses several engineering challenges for large companies with billions of users that have huge telemetry clients, which log thousands of telemetry events. To make matters worse, telemetry data is collected periodically (say every day or every 6 hours [9]), and most implementations also use *memoization* tables to reduce privacy leakage across different rounds [14, 9]. This means that the server needs to ensure that every client is *synchronized*, in the sense that they have the access to fresh random bits for every round of telemetry collection or when the memoization tables are flushed. This entails the server and clients maintaining a state for every single telemetry event and every round, and guaranteeing that every client is synchronized with the server. For many large scale telemetry systems this extra overhead can lead to several tricky implementation issues. Hence, based on our conversations with practitioners, the following question arose.

"Is there a distributed algorithm for learning heavy hitters that eleminates the need for shared randomness across users?"

This intriguing question is the motivation behind this paper. First we formalize what we mean by a distributed algorithm.

Definition 1. *An algorithm for heavy hitters problem is distributed if there is no shared random bits across users. Each user knows only the privacy parameter ϵ and size of the universe d.*

Note that the size of universe is often implicitly known in most applications. For example, in the case of URLs it is determined by the maximum size a URL can take. Furthermore, the privacy parameter is often configured in the client software. All known algorithms [4, 2] for heavy hitters do not satisfy the above definition.

Our results.

The main result of the paper is the following.

Theorem 1. *There exists an efficient and distributed algorithm, which we name as AoN, for the frequency oracle and the heavy hitters problems in the local model of differential privacy. AoN gives following guarantees: 1) AoN is ϵ-locally differentially private for any $\epsilon > 0$; 2) AoN achieves the optimal worst case error guarantee for the frequency oracle problem. The worst case error guarantee of the frequency oracle is at most*

$$\frac{e^{\frac{\epsilon}{2}}(1+e^{\frac{\epsilon}{2}})}{e^{\frac{\epsilon}{2}}-1} \cdot \sqrt{\frac{1}{2n}\log\left(\frac{2d}{\delta}\right)},$$

which holds with probability $1-\delta$; 3) The worst case error guarantee of the heavy hitters algorithm is at most $\frac{e^{\frac{\epsilon}{4}}(1+e^{\frac{\epsilon}{4}})}{e^{\frac{\epsilon}{4}}-1} \cdot \sqrt{\frac{1}{2n}\log\left(\frac{2d}{\delta}\right)\log n}$, which holds with probability $1-\delta$. 4) The client side running time of AoN is $O(\max\{\log n, \log d)^2\})$; 5) The communication complexity of AoN is $O(\max\{\log n, \log d\})$ bits.

The error guarantees obtained by AoN matches that of TreeHist. We note that the number of bits transmitted from a client to the server is more in AoN compared to TreeHist ($O(\log n)$ bits compared $O(1)$). However, the number bits sent from the server to a client (or alternatively, the number of bits a client downloads from the server) is zero where as in TreeHist it $O(\log n)$. Moreover, AoN eliminates the need for the clients to be synchronized, which we believe is the novel aspect of our scheme compared to every existing LDP mechanism. We give a complete proof this theorem in the full version of the paper.

Given an optimal frequency oracle, we also show that heavy hitters problem can also be solved in a distributed manner by some modifications to the schemes in [2].

Apart from the properties mentioned in Theorem (1), our algorithm has several other desirable characteristics: 1) AoN is simple to specify and memoize compared to the existing mechanisms. 2) The constants in the error guarantee given by AoN are considerably smaller compared to the TreeHist algorithm. However, we would like to emphasize that the main technical contribution of the paper is a novel distributed and optimal algorithm for the frequency oracle problem with a theoretical analysis. To the best of our knowledge this is the first such known algorithm for the heavy hitters problem.

2 All-or-Nothing (AoN) Mechanism

Now we present AoN mechanism, which is a computation and communication efficient algorithm for computing heavy hitters. *An algorithm is computation and communication efficient if a) its running time is polynomial in n and $\log d$, b) the number bits transmitted by the algorithm is polynomial in n and $\log d$.* Observe that precisely because of this reason we can not simply use a histogram algorithm to solve the frequency estimation problem for heavy hitters problem. Such an approach would require m bits of communication. AoN mechanism consists of two parts: 1) A client side protocol that is invoked on each user device to report differentially private version of the item $v_i \in \{1, 2 \dots d\}$ that user i holds. 2) A frequency oracle on the server side that collects user reports and produces an estimated frequency of an item $v_i \in \{1, 2 \dots d\}$. Now we describe these two algorithms and prove their properties.

2.1 AoN Client Protocol

During the data collection, each user $i \in [n]$ samples uniformly at random a hash function $h_i \in H$ from a universal family of hash functions H. (It can be shown that sampling a hash function from an universal family is equivalent to selecting two integers $a, b \in [p]$ uniformly at random.) Next, the user i evaluates $h_i(v_i)$. If $h_i(v_i) = 1$, then the user reports the hash function h_i to the server. If $h_i(v_i) \neq 1$, then user tosses a coin with bias $\frac{1}{e^\epsilon}$, where ϵ is the privacy parameter; if it lands heads, then the user sends h_i to the server. Otherwise, the user transmits an empty string ϕ. The formal description of the algorithm is given in Algorithm 1.

The below lemma follows immediately from the definition.

Lemma 1. *Each user $i \in [n]$ transmits $O(\max\{\log n, \log d\})$ bits to the server, and the running time of Algorithm 1 is $O(\max\{\log n, \log d)^2\}$.*

Algorithm 1 AoN Client Protocol

AoN **Mechanism for Data Collection on Client**

Input: an element $v_i \in [d]$ from user i, privacy parameter $\epsilon \geq 1$;

Output: a hash function $h_i \in H$ or ϕ, where each $h_i : [d] \to [B]$, and $B = \lceil e^{\epsilon/2} + 1 \rceil$;

1: Randomly draw a hash function h_i from a universal family $H = \{h : [d] \to [B]\}$;
2: If $h_i(v_i) = 1$: output h_i;
3: Else: output h_i with probability $1/e^{\epsilon}$, and ϕ otherwise.

Algorithm 2 AoN Frequency Oracle

Processing Frequency Query on Server

Input: an element $v \in [d]$;

Output: an estimation $\hat{f}(v)$ of the frequency of v ($f(v) = |\{i \in [n] \mid v_i = v\}|/n$);

1: For each user $i \in [n]$:
2: Collect $A(v_i)$ using the client-side mechanism;
3: If $A(v_i) \neq \phi$:
4: If $h_i(v) = 1$: $\theta \leftarrow \theta + 1$;
5: Output the estimated frequency of v as

$$\hat{f}(v) = \left(\frac{\theta}{n} - \frac{1}{e^{\frac{\epsilon}{2}}(1 + e^{\frac{\epsilon}{2}})}\right) \cdot \frac{e^{\frac{\epsilon}{2}}(1 + e^{\frac{\epsilon}{2}})}{e^{\frac{\epsilon}{2}} - 1}$$

Next, we show that AoN is ϵ-differentially private. We defer the proof to a full version of the paper.

Lemma 2. *For any $\epsilon > 0$, Algorithm 1 is ϵ-differentially private.*

2.2 AoN Frequency Oracle

Our server side frequency oracle algorithm is quite simple. Suppose we want to estimate the frequency of an item $v \in [d]$. The algorithm scans through hash functions received from the users, and counts the number of hash functions that evaluate to 1 on the item v. Let θ be this quantity. Our algorithm outputs the frequency of item v as

$$\hat{f}(v) = \left(\frac{\theta}{n} - \frac{1}{e^{\frac{\epsilon}{2}}(1 + e^{\frac{\epsilon}{2}})}\right) \cdot \frac{e^{\frac{\epsilon}{2}}(1 + e^{\frac{\epsilon}{2}})}{e^{\frac{\epsilon}{2}} - 1} \tag{1}$$

A formal description of our algorithm is given in Algorithm 2.

It remains to analyze the error guarantee of our algorithm. As a first step towards that we show that the expected value of $\hat{f}(v)$ is equal to the true frequency of item v.

Lemma 3. *For every item $v \in [d]$, $\hat{f}(v)$ is an unbiased estimator of $f(v)$; that is $E[\hat{f}(v)] = f(v)$.*

We do a concentration analysis to prove the error guarantee.

Lemma 4. *For every item $v \in [d]$ and $\delta > 0$, with probability $1 - \delta$,*

$$|\hat{f}(v) - f(v)| \leq \frac{e^{\frac{\epsilon}{2}}(1 + e^{\frac{\epsilon}{2}})}{e^{\frac{\epsilon}{2}} - 1} \cdot \sqrt{\frac{1}{2n} \log\left(\frac{2d}{\delta}\right)}$$

The proof of the main theorem (1) now follows from Lemmas (1,2,3,4). The omitted proofs will be made available in the full version of the paper.

References

[1] S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *ICDE*, pages 193–204, 2005.

[2] R. Bassily, K. Nissim, U. Stemmer, and A. Thakurta. Practical locally private heavy hitters. In *NIPS*, 2017.

[3] R. Bassily, K. Nissim, U. Stemmer, and A. G. Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2285–2293, 2017.

[4] R. Bassily and A. D. Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, pages 127–135, 2015.

[5] R. Bassily, A. D. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473, 2014.

[6] S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[7] J. L. Carter and M. N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, pages 106–112, New York, NY, USA, 1977. ACM.

[8] M. Dietzfelbinger, J. Gil, Y. Matias, and N. Pippenger. Polynomial hash functions are reliable (extended abstract). In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming*, ICALP '92, pages 235–246, London, UK, UK, 1992. Springer-Verlag.

[9] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3574–3583, 2017.

[10] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438, 2013.

[11] J. C. Duchi, M. J. Wainwright, and M. I. Jordan. Local privacy and minimax bounds: Sharp rates for probability estimation. In *NIPS*, pages 1529–1537, 2013.

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[13] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[14] Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

[15] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.

[16] G. C. Fanti, V. Pihur, and Ú. Erlingsson. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *PoPETs*, 2016(3):41–61, 2016.

[17] J. Hsu, S. Khanna, and A. Roth. Distributed private heavy hitters. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 461–472, 2012.

[18] X. Hu, M. Yuan, J. Yao, Y. Deng, L. Chen, Q. Yang, H. Guan, and J. Zeng. Differential privacy in telco big data platform. *PVLDB*, 8(12):1692–1703, 2015.

[19] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, pages 143–152, New York, NY, USA, 2006. ACM.

[20] A. Thakurta, A. Vyrros, U. Vaishampayan, A. Kapoor, J. Freudiger, V. Sridhar, and D. Davidson. Learning new words. In *US Patent 9594741*, 2017.

[21] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[22] L. Wasserman and S. Zhou. A statistical framework for differential privacy. *Journal of the American Statistical Association*, 105(489):375–389, 2010.