RAPPOR-A LDP model for frequency estimation

Arnab Banerjee, Kun Jin

November 2018

1 Introduction

Local Differential Privacy is a methodology to learn about client statistics without explicitly learning about individual clients in the client population. Local differential privacy ensures that the sensitive data is privatized right at the start before the data leaves the client's device. The privatized data is post-processed at the server side to gain insights into population statistics.

Randomized Aggregatable Privacy-Preserving Ordinal Response(RAPPOR)[2] is a locally differentially private model used for collecting client-side information in a differentially private fashion while ensuring efficient and accurate analysis of the collected data. Some of the key features introduced by this model is this model allows collection of statistics on random set of strings by applying randomized response on Bloom filter. Another important feature is that it provides privacy guarantees even in the case when reports are collected from the same client more than once thus providing longitudinal privacy.

RAPPOR is based on the idea of randomized response[2],a technique to conduct surveys on sensitive topics while providing privacy guarantee to the respondents.The traditional randomized response surveying technique does not provide any longitudinal privacy and thus RAPPOR offers significant advantage in this aspect by providing longitudinal privacy. Since RAPPOR is a locally differentially private model it avoids addition of privacy externalities, such as those that might be created by maintaining a database of contributing respondents.

Large enterprises often want to understand population statistics of its user base. They want to understand how frequently certain software features are used, and measuring a software's performance through telemetry. RAPPOR provides a differentially private methodology to achieve the above purpose without compromising significantly on accuracy. RAPPOR has been implemented and deployed in the Chrome Web browser to collect data about Chrome clients. More specifically, Google used RAPPOR to learn the distribution of Chrome homepages across its twelve million customer base[2].

2 Data Structures& Algorithms used

2.1 Bloom Filter

A Bloom Filter is a probabilistic bit array [7],that is used to test whether an element is a member of a set. If the element is actually a member then Bloom Filter would detect it while some non-member elements can also be predicted to be a member of the set due to collisions.

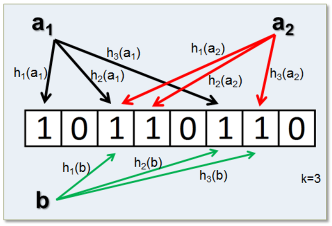

Figure 1: Illustration of Bloom Filter

2.2 Lasso Regression

Lasso is a L1-regularization technique which is primarily used to do feature selection. It is generally used when we have more number of features, because it automatically does feature selection. Lasso regression tries to fit given data while minimizing the summation of MSE of the training data and the feature coefficients. It finds the optimal solution to the following equation:

$$\min\left(||Y - X\theta||^2 + \lambda||\theta||_1\right) \tag{1}$$

Here λ is the hyperparameter. The lasso regression ensures that the coefficients of many features result in zero thus making the model less complex and less susceptible to noise in the data.

2.3 Ordinary Least Square Regression:

Least square regression tries to fit the training data by minimizing the MSE of the entire training set. This model helps us to find the parameters of the features presented to it. In Ordinary least square regression the dependent variable is assumed to have a linear relationship with the independent variables. There is a closed form solution to the ordinary least square regression problem.

3 Rappor Algorithm

3.1 Client side

3.1.1 Inputs:

1. True client value-v
2. Cohort of client-m
3. Bloom Filter size-k
4. Number of hash functions-h
5. Hyperparameters for adding noise- f

3.1.2 Output:

Noisy encoded representation of v

3.1.3 Algorithm:

1. **Signal:** Hash client's value v concatenated with cohort m onto the Bloom filter B of size k using h hash functions.
2. **Permanent randomized response:** For each bit i, $0 \leq$ i $<$ k in B, create a binary bit array $B^{'}$ which equals to

$$B_i^{'} = \begin{cases} 1 \quad \textit{with probability } f/2 \\ 0 \quad \textit{with probability } f/2 \\ B_i \; \textit{with probability } 1 - f \end{cases} \tag{2}$$

3. **Report:** Send the bit array $B^{'}$ to server.

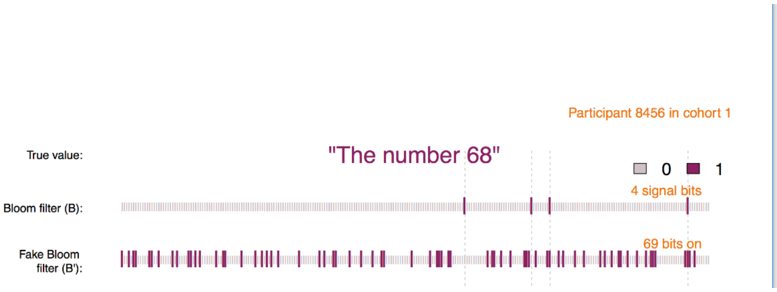

Figure 2: Working of client side algorithm

3.2 Server side

3.2.1 Inputs:

1. Noisy representation of client values
2. Cohort of all clients
3. Bloom Filter size
4. Number of hash functions
5. Hyperparameters- f

3.2.2 Output:

Estimated distribution of client values

3.2.3 Algorithm:

1.Calculate the number of times each bit i(0 ≤ i < k) in cohort j(1 ≤ j ≤ m) , c_{ij} was set in a set of N_j reports.
2. Estimate the number of times each bit i within cohort j, t_{ij}, is truly set in B for each cohort. Given c_{ij}, the estimate is given by

$$t_{ij} = \frac{c_{ij} - 0.5f * N_j}{(1 - f)} \tag{3}$$

3. Create a design matrix X of size km * M where M is the number of candidate strings under consideration. Each column of X contains h*m 1's at positions where a particular candidate string was mapped to by the Bloom filters in all m cohorts.

4. Use Lasso regression to fit a model $Y \sim X$ and select candidate strings corresponding to non-zero coefficients.

5. Fit a regular least-squares regression using the selected variables to estimate counts, their standard errors and p-values.

4 Properties of RAPPOR

1. Longitudinal Privacy: Longitudinal privacy is ensured with the help of two distinct randomization steps as specified in the step 2 of the given client side algorithm and also an additional "Instantaneous Randomized Response" step described in the original paper. The "Permanent Randomized Response" step is used to create a “noisy” answer which is memoized by the client permanently and reused later when the client sends the same actual value. The "Instantaneous" randomization step will perform randomization over the permanent response. If an attacker has all the instantaneous reports then those reports

can completely reveal the permanent response but it will never reveal the actual client value. Thus long term, longitudinal privacy is ensured by the use of the Permanent randomized response, while the Instantaneous response makes it difficult to identify a client based on the permanent randomized response. This step also provides stronger short-term privacy guarantees.

2. Privacy Guarantees for Permanent Randomized Response:
The Permanent randomized response (Steps 1 and 2 of RAPPOR) satisfies ϵ_∞ differential privacy where

$$\epsilon_\infty = 2h * ln(\frac{1 - 0.5f}{0.5f}) \tag{4}$$

5 Types of RAPPOR

1. One-time RAPPOR: Client submits response only once and thus longitudinal privacy is not required. The Instantaneous randomized response step is not required in this case.

2. Basic RAPPOR: If the set of candidate strings are small then each string can be mapped deterministically to a single bit in the bit array and thus probabilistic Bloom filter need not be used. This modification would affect step 1, where a probabilistic Bloom filter would be replaced by a deterministic mapping of each candidate string to a single bit in the bit array. In this case, the effective number of hash functions, h, would be 1.

3. Basic One-time RAPPOR: This is the simplest configuration of the RAPPOR mechanism, combining the first two modifications at the same time.

6 Experiments

6.1 Experiment Settings

The differential-privacy $\epsilon_\infty = 2ln(3)$ is guaranteed in our experiments. Specifically, $f = 0.73$, $h = 2$ and based on Equation 4, we can compute the ϵ_∞.

6.2 Evaluation Metrics

In the following experiments, to evaluate the results, we use two metrics, namely,

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

where TP means True Positive (which should be Positive), FP means False Positive (which should be Negative), and FN means False Negative (which should be Positive).

6.3 Influence Factors

At the beginning, we used the md5 hashing technique to get the bloom filter indices but it is not mapping uniformly due to the hashed string being hexadecimal. Instead, we harness murmur3 hashing technique to compute the bloom filter indices, and in this case we can arrive at a better performance and more reasonable results.

6.3.1 Bloom filter size

In this experiment, the rappor works best when the bloom filter size is 48 from Figure 3. Another two parameters are fixed, hash function number for one cohort is $h = 2$ and cohort number is set as $m = 8$. We can see that the precision is always as high as above 90%. From the beginning, when the bloom filter size is 16 and 32, the average recall ratio is around 50%. When the bloom filter size is 48, the average recall ratio can be improved to be 56%. When the size is enlarged, the recall ratio is a bit lower, but still keep 50% or so.

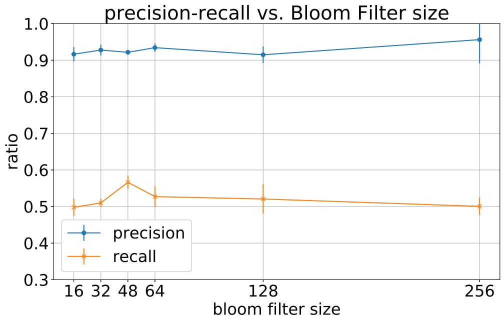

Figure 3: Bloom filter size factor

6.3.2 Number of Hash functions

This experiment is to see how the number of hash functions can impact the precision and recall of the decoding utility. The result is shown in figure 4. Also, here we fix other two parameters: bloom filter size is $k = 48$, cohort number is $m = 8$. From the figure, the precision is also above 90%. And when the hash number is 2, we can arrive at a relatively higher recall ratio (56%) although slightly than other settings.

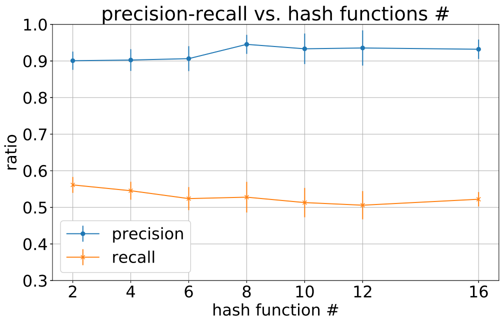

Figure 4: Hash functions number factor

6.3.3 Number of cohorts

This experiment is to see how the number of cohorts impact the performance. Here, we also fix other two parameters: bloom filter size is $k = 48$, hash functions number is $h = 2$. From figure 5, we can see that the precision can always be above 90%. And when cohort number is 4, the recall ratio is about 52%, and when it is 8, we can arrive at a relatively higher recall ratio 56%, and when it is larger, the recall ratio is a bit dragged down, maybe this is because True Positive number is larger since the distribution of each client string is wider and the lasso may have strict effects on selecting significant columns (candidate strings).

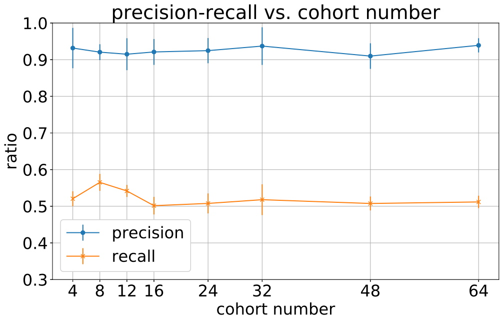

Figure 5: Cohort number factor

6.4 Estimation on String Names

Table 6.4 lists the top-10 most frequent string names with their estimated frequency. Also in our experiment whose result is in Figure 6, rappor cannot make accurate results on strings with few frequency (rare strings). This is in fact

string	true frequency	estimated frequency
of	37436	35874
and	29366	25143
to	26603	21019
a	23654	30465
in	21928	18008
that	10687	14046
he	9598	11596
for	9592	10458
with	7561	8939
as	7394	5216

incurred by over-dispersion problem, which means that the standard deviation is larger than the mean.

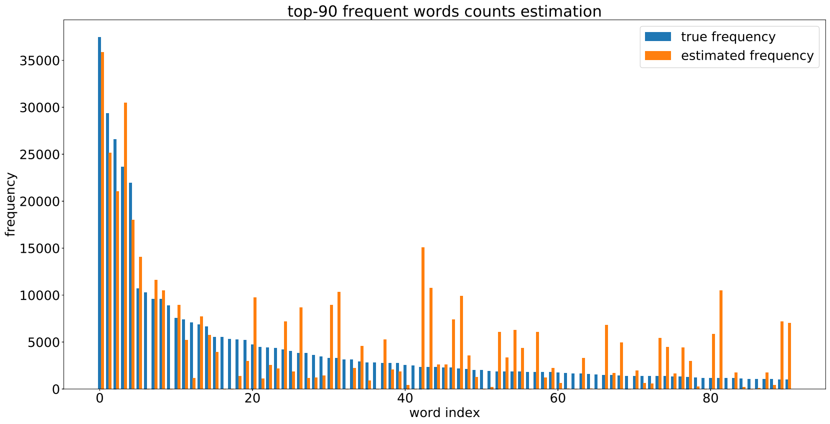

Figure 6: top-90 frequent words counts estimation

7 Discussion

In this work, on the server side, we find the experiment parameter should be carefully tuned to get better performance. Also the computation cost is large when the candidate strings are large since lasso needs long time to select columns (candidate strings) based on $L1$ regularization. On the client side, communication overload is a bit high since each time the client should send size k bits of bloom filter size, which is a cost when communication is frequent.

8 Related Work

To protect the users' security and privacy has been an important and hot research topic. Recently, differential privacy has become the well-studied and widely accepted definition of privacy. And in this area, local differential privacy is proposed to preserve the privacy of the data. Heavy-hitters [1, 5, 4, 2] are mostly harnessed in recent studies. For the heavy-hitters related work, Chan *etal.* [1] preserves the privacy while untrusted aggregator wants to estimate accuratly the heavy-hitters across a set of distributed streams which may contain sensitive data. Hsu *etal.* [4] presents efficient algorithms and lower bounds for solving the heavy hitters problem while preserving differential privacy in a fully distributed local model, in which the problem is to find the identity of the most common element shared among n parties. Kilian *etal.* [5] considers the problems of computing the Euclidean norm of the difference of two vectors and the large different components which are modelled as heavy hitters, then they implement approximate but private protocols in the semi-honest model. Bloom filters [2] are used as the same function as Heavy-hitters, which are adopted by Rappor. Rappor provides a privacy-preserving way to learn the behavior of users' software statistics while guaranteeing the software privacy, *i.e.*, without leaking the individual user's personal information by means of ramdomized response. The concept of randomized response method originates from [6] to eliminate evasive answer bias allowing the interviewee to maintain privacy through the device of randomizing his response. Further, following the randomized response mechanisms, Fanti *etal.* [3] proposes to decode joint distributions from noisy reports, and to learn distributions when the aggregator does not know the dictionary of strings beforehand.

9 Conclusion

RAPPOR can be used to detect heavy hitter strings while maintaining privacy. RAPPOR provides privacy guarantees using rigorous mathematical framework while ensuring the method is accurate enough to be used in practice. The key strength of RAPPOR lies in the fact that it provides longitudinal privacy by randomizing the client response using two separate randomization steps. Though the parameters f,p,q are usually fixed depending on the level of privacy required the other parameters are tunable and thus parameter selection plays a significant role in the final accuracy of the algorithm.

References

[1] T-H Hubert Chan, Mingfei Li, Elaine Shi, and Wenchang Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 140–159. Springer, 2012.

[2] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.

[3] Giulia Fanti, Vasyl Pihur, and Úlfar Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 2016(3):41–61, 2016.

[4] Justin Hsu, Sanjeev Khanna, and Aaron Roth. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2012.

[5] Joe Kilian, André Madeira, Martin J Strauss, and Xuan Zheng. Fast private norm estimation and heavy hitters. In *Theory of Cryptography Conference*, pages 176–193. Springer, 2008.

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

[7] Wikipedia. Bloom filter — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Bloom%20filteroldid=868330445, 2018. [Online; accessed 29-November-2018].