A model-based filter to improve local differential privacy

Juan M. Gutierrez [1] **Valérie Gauthier-Umaña** [2] **Juan F. Pérez** [3]

Abstract

Local differential privacy has been gaining popularity in both academic and industrial settings as an effective mechanism to enable the computation of summary statistics by service providers while providing privacy guarantees to end users. In the proposed mechanisms of local differential privacy, the introduction of noise in the user data is key to preserve privacy, but can greatly limit the estimation power on the provider side. In this paper we propose to include pre-filters based on machine learning models to help discard observations that may be too noisy to add value to the estimation process. We test our approach on both synthetic and real datasets, and identify average reductions of up to 31% in the mean squared error.

1. Introduction

User data collection is a very powerful tool for service providers to improve the relevance and timeliness of their services. While service providers typically aim to perform summary statistical and machine learning analyses, these may require collecting individual information. However, as these data may be sensitive, people may not be willing to share it. To resolve this conundrum, the field of differential privacy (Dwork & Roth, 2014) has focused on developing mechanisms that enable a potential provider to extract valuable summary information from the data without being able to recover individual user information. We are particularly interested in Local Differential Privacy (LDP), where the raw data is never submitted to the provider, which only receives a noisy version of it. The addition of noise allows the user to deny any response and protect his privacy.

[1]Quantil, Colombia [2]Systems and Computing Engineering Department, Universidad de los Andes, Colombia [3] Centro para la Optimización y Probabilidad Aplicada (COPA), Department of Industrial Engineering, Universidad de los Andes, Colombia. Correspondence to: Juan M. Gutiérrez <miguel.gutierrez@quantil.com.co>, Valérie Gauthier-Umaña <ve.gauthier@uniandes.edu.co>, Juan F. Pérez <jf.perez33@uniandes.edu.co>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

Related work. A number of LDP protocols have been proposed, as described and benchmarked in (Wang et al., 2017) and (Cormode et al., 2021), including some used by companies such as Apple (Apple, 2017) and Google, which proposed and implemented RAPPOR (Erlingsson et al., 2014). Different protocols vary in their encoding, using local hashing (Wang et al., 2017) or Hadamard transformations (Apple, 2017), the domain size reduction using sketches or Bloom filters (Erlingsson et al., 2014), and in methods to identify heavy hitters or post-process the data to improve the estimation via rounding and projections (Wang et al., 2017; Cormode et al., 2021).

Filtering for improved estimation. While the post-processing techniques mentioned above have focused on adjusting the counts after the estimation, in this paper we propose to *add a filtering step* to improve the estimation by removing excessively noisy observations at the server. While this technique can be combined with any of the existing methods, we choose RAPPOR (Erlingsson et al., 2014) as a baseline, as it has been recently shown to provide competitive results (Cormode et al., 2021) and displays a number of variations relevant for analysis. Tests conducted on both synthetic and real datasets show that the pre-filter approach is effective to improve the estimation, reducing the mean squared error by up to 31%.

The next section introduces background material, and Section 3 describes the proposed model-based filter approach, for which we present experimental results in Section 4. We conclude with a discussion of the results and future work.

2. Background

In this section we provide necessary definitions regarding local differential privacy, the RAPPOR method (Erlingsson et al., 2014), on which we build upon, and some recent proposals on Bloom filter detection.

Local Differential Privacy. We consider the same setup as in RAPPOR (Erlingsson et al., 2014), where a set of users submit a value of interest (e.g., the URL of a website visited by the user) to a central server that accumulates this information to generate summary statistics. Before submission, the user value v is passed through an algorithm A that guarantees its (local differential) privacy, defined as

follows (Wang et al., 2017).

Definition 2.1. (Local Differential Privacy) Given a privacy budget $\epsilon \geq 0$, an algorithm A satisfies ϵ-local differential privacy if and only if for any input values v_1 and v_2,

$$Pr(A(v_1) = y) \leq e^{\epsilon} Pr(A(v_2) = y),\ \forall y \in \text{Range}(A),$$

i.e., for all possible outputs of algorithm A.

Following this definition, a few algorithms have been proposed and implemented. Here we focus on RAPPOR (Erlingsson et al., 2014), which has received much attention and has been implemented by Google.

RAPPOR. Given a user value v, the Rappor algorithm executed at the client works as follows: (i) a Bloom filter is applied to the value v to obtain a bit array of size k. To this end h hash functions $H = \{H_1, ..., H_h\}$ are used, each of which outputs an integer in $[k] = \{0, 1, ..., k-1\}$. The result is a size-k bit array B_0 with h positions equal to 1 and the rest set to 0. (ii) a perturbation is applied to each bit $i \in B_0$ to obtain a randomized version B_1 as

$$Pr(B_1[i] = 1) = \begin{cases} 1 - \frac{1}{2}f, & B_0[i] = 1, \\ \frac{1}{2}f, & B_0[i] = 0. \end{cases}$$

The resulting vector B_1 is a noisy representation of the user value v that is then submitted to the server. This mechanism has been proven to be $2h \ln\left(\frac{1-\frac{1}{2}f}{\frac{1}{2}f}\right)$ locally differentially private (Erlingsson et al., 2014).

Once the results from all users are received at the server, they are processed to obtain frequency estimates for the values observed. Here the server goes through three main steps: (i) noise removal, via a Bayesian estimation of the actual counts observed; (ii) detection of the values observed, which is performed via a Lasso regression to estimate the actual non-zero counts; (iii) estimation of the counts, via linear regression. These steps, explained in detail in (Erlingsson et al., 2014), can also be followed by additional post-processing as discussed in (Cormode et al., 2021) to improve the frequency estimation.

Bloom filter detection. Recently, there has been a proposal to improve indexes, such as Bloom filters, through the use of deep-learning models (Kraska et al., 2018; Mitzenmacher, 2018). The objective is to employ learning models to determine the structure of lookup keys and use this signal to effectively predict the position or existence of records. The deep learning model is trained with keys that are known to either exist or not in the index, such that it can learn to predict whether a new key is part of the index or not (Kraska et al., 2018; Mitzenmacher, 2018). We employ this idea in building our pre-filter model, but in this case, the inputs used in the model are the outputs of the RAPPOR method executed at the client's local machine, which includes both the Bloom filter and the randomized response steps. The next section explains this in more detail.

3. A Filter for Local Differential Privacy

We propose to incorporate a pre-filter model in RAPPOR to improve the frequency estimation in a local differential privacy setting. The pre-filter is executed at the server whenever a new observation B_1 is received from a client. Based on a pre-trained deep-learning model, the pre-filter determines whether the observation corresponds to any of the values to be considered in the estimation step or not. Since B_1 is a randomized version of B_0, the noise introduced in the randomization step can be so much that the observation becomes of little value for the estimation step, and can instead introduce undesired noise. The pre-filter thus helps the estimation step by preventing excessively noisy observations from being included as part of the estimation set. We now describe the model and its training.

3.1. The Learning Model for the Pre-Filter

The objective of the pre-filter is to detect excessively noisy observations received from a client. Thus, it must be able to determine whether a given observation, which is a size-k binary vector, is sufficiently similar to the vectors produced by the Bloom filter when it receives as input a value that is expected to be in the database. Here we assume that the set of possible values has been determined previously, with approaches such as those described in (Wang et al., 2017; Cormode et al., 2021). Since the pre-filter receives data from the client, the value v generated by the user has already gone through the Bloom filter and the randomization described in Section 2. We refer to this version of the value v as *encoded* v.

The pre-filter thus faces a binary classification task. That is, we want to learn a model f that can predict if an encoded value v is similar to any of the encoded versions of the expected values, or if instead it can be considered as a noisy observation. To this end we build two sets: the first set is $\mathcal{K}$, which is made of all the size-k vectors that the Bloom filter generates for each of the values expected to be observed. Note that these vectors are encoded in that they have passed through the Bloom filter, but no randomization noise has been added. The second set is $\mathcal{U}$, which we build by generating random size-k vectors, where each bit is set to one with probability γ. These vectors are the reference of what can be considered a random vector that adds little information for the estimation step.

With these sets we can then train a neural network model as in (Mitzenmacher, 2018), using as training dataset $\mathcal{D} = \{(B_i, y_i = 1) | B_i \in \mathcal{K}\} \cup \{(B_i, y_i = 0) | B_i \in \mathcal{U}\}$, such that category 1 is associated with encoded vectors sufficiently

similar with those expected to be observed, while category 0 refers to random or noisy vectors. Since this is a binary classification problem, the neural network can have sigmoid or hyperbolic tangent activation functions to produce a probability. The model is trained to minimize the log loss function $L = \sum_{(B,y)\in\mathcal{D}} y \log f(B) + (1-y)\log(1-f(B))$.

The output of $f(B)$ can be interpreted as the probability that B is an encoded vector similar to those expected, and can thus be passed onto the estimation step. We include in the estimation step all vectors B such that $f(B) > \tau$, where τ is a threshold parameter in $[0.5, 1]$.

3.2. The Model Architecture

The model consists of a six-layer network, where the first three layers are 1D convolutional layers using ReLU activation functions with 128 (kernel size 7), 64 (kernel size 3), and 16 (kernel size 2) neurons, respectively. The last of these convolutional layers is processed by max polling (kernel size 2). The last three layers are dense with 64, 32, and 1 neurons, respectively, where the first two layers use a ReLU activation function, and the last layer uses a hyperbolic tangent activation function. Also, we use a dropout of 0.5 between layers.

3.3. A Second Pre-Filter

The results in Section 4 illustrate the gains in accuracy obtained with the model-based pre-filter described above. The model is able to identify patterns in the randomized data that can thus be used to discard exceedingly noisy data points. In the same vein, we could consider capturing other patterns additional to those already incorporated by the pre-filter built on the set $\mathcal{D}$. To explore this possibility we build a second pre-filter, where the aim is to capture a second-order pattern, i.e., a pattern that involves two or more bits in the encoded vector.

For the second pre-filter we compute a new set $\hat{\mathcal{D}} = \{(\hat{B}_i, y_i = 1) | \hat{B}_i \in \hat{\mathcal{K}}\} \cup \{(\hat{B}_i, y_i = 0) | \hat{B}_i \in \hat{\mathcal{U}}\}$, where the vectors $\hat{B}_i$ are $k/2$ bits long and are built as follows. Each element $\hat{B}_i \in \hat{\mathcal{K}}$ corresponds to one element $B_i \in \mathcal{K}$, with entries $\hat{B}_i[j] = B_i[j] \vee B_i[j + k/2]$ for $i \in \{1, \ldots, k/2\}$. That is, we perform a logical OR operation among two entries of the vector B_i, without repetition. To obtain the elements in $\hat{\mathcal{D}}$ we perform a similar OR operation on the random vectors in $\mathcal{D}$ to generate binary vectors of size $k/2$.

This procedure aims to obtain a reduced version of the original vectors in $\mathcal{K}$, combining two bits at a time, and aiming to keep a pattern of zeroes that can be recognized by the detection model when contrasted with the random vectors in $\hat{\mathcal{U}}$. We then train a similar network model on these data to obtain the second pre-filter. In production, when a vector is *discarded* by the first pre-filter for being too noisy, we apply the second pre-filter, looking to prevent the early dismissal of a vector that may have appeared noisy to the first pre-filter but that kept relevant information for the estimation step. The second pre-filter thus acts as a second chance for a vector to be considered in the estimation step.

3.4. Ensuring differential privacy with pre-filters

Applying a post-processing step such as the introduction of the pre-filters proposed above may raise the question of its impact on the privacy. As the pre-filter can be seen as a deterministic mapping that assigns a weight of zero to those observations considered too noisy, we prove the following result that shows that applying such a mapping does not impact the privacy. We prove the result for the case of *local* differential privacy, following a similar result in (Dwork & Roth, 2014) for the non-local (centralized) case.

Proposition 3.1. *Let A be an algorithm that satisfies ϵ-local differential privacy with range R and let $f : R \to R'$ be a deterministic mapping with range R'. Then the composition $f \circ A$ is ϵ-local differentially private.*

Proof. For any element $y' \in R'$ there is a set of preimages $T = \{y \in R | f(y) = y'\}$. Consider v_1 and v_2 two different inputs of A, we have

$$\begin{aligned} Pr(f \circ A(v_1) = y') &= Pr(A(v_1) \in T) \\ &= \sum_{y\in T} Pr(A(v_1) = y) \\ &\leq \sum_{y\in T} e^{\epsilon} Pr(A(v_2) = y) \\ &= e^{\epsilon} Pr(f \circ A(v_2) = y'), \end{aligned}$$

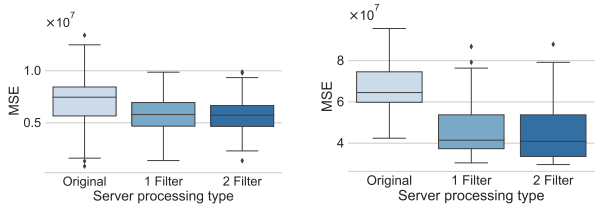
which proves that $f \circ A$ is ϵ-local differentially private. □

4. Results

To assess the effectiveness of our approach, we evaluate first it on synthetic data generated from a Zipf distribution with a domain of size $d = 100$ and a system composed of $n = 100000$ users or observations. Each observation goes through the Bloom filter and randomized response steps of Rappor before being processed by the pre-filter.

In Figure 1 we consider an initial setup with a privacy budget of $\epsilon = 3$, a Bloom filter with $h = 2, k = 128$, and a threshold in the learning model of $\tau = 0.8$. With this privacy budget, the associated value of f is 0.64. We consider four different setups: the original setup from Rappor (Erlingsson et al., 2014), the setup adding one filter, and adding a second filter. For the second filter we consider the description in Section 3.3 and a second version where we invert the indexes when applying the OR operator. For each setup we execute

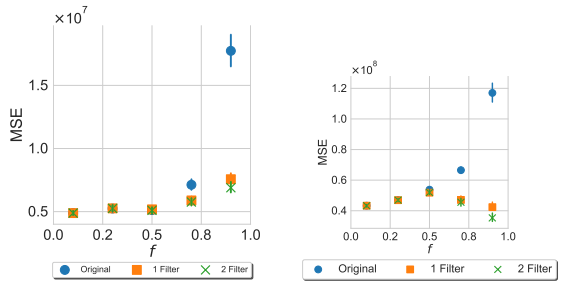
100 repetitions with different datasets and compare the mean squared error (MSE) of the frequency estimation.



(a) Zipf distribution (b) Diabetes data

Figure 1. MSE comparison of the baseline Rappor with 1 and 2 pre-filters. $f = 0.7, h = 2, k = 128, \tau = 0.8$

Figure 1(a) shows that adding the pre-filter model results in a better estimation, reflected in a smaller MSE mean and standard deviation. The introduction of a single filter reduces the mean MSE by about 24%, whereas including a second filter produces an additional, albeit smaller, reduction of 3%, for a total reduction in mean MSE of 27%.



(a) Zipf Distribution (b) Diabetes data

Figure 2. Impact of noise probability f. $h = 2, k = 128, \tau = 0.8$

We now evaluate the impact of the probability f, which is fixed by the privacy budget (a tighter budget requires a larger f) and determines how much noise is added to the output of the Bloom filter. We thus fix the other parameters as before, i.e., $h = 2, k = 128, \tau = 0.8$, and vary the probability $f \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The results in Figure 2(a) show that as more noise is added through the probability f the gap between the baseline RAPPOR and pre-filter model increases. This means the model-based pre-filter is able to drop those observations which are excessively noisy and improves the count estimation. Note that the value of f is larger when the privacy budget is tighter, which is precisely the case when the pre-filter is more beneficial.

Now we consider the Diabetes Health Indicators Dataset from the Behavioral Risk Factor Surveillance System, available in (CDC & Teboul). This dataset contains information from 253680 Americans surveyed about risk factors related to diabetes. We estimate the count frequency on the Body Mass Index variable, which has domain $[1, 2, ..., 100]$.

Figure 1(b) shows that in this case the introduction of a single filter reduces the mean MSE by about 29%, and the second filter further reduces it by 31%. As before, we evaluate the impact of the probability f in Figure 2(b), which shows the added benefits of the pre-filter as the f increases for this real dataset.

5. Discussion, limitations and future work

The results in the previous section illustrate the benefits of incorporating a model-based pre-filter in an LDP mechanism. The benefits are extracted mostly with a single pre-filter, although the addition of a second pre-filter provides further improvements, leading to a reduction of up to 31% in MSE. Also, the benefits are larger when the noise f is larger, which is related to scenarios with tighter privacy budgets.

While we have showcased that the pre-filter approach provides benefits for the estimation step, many open avenues of investigation remain to be explored in future work. In particular, it is necessary to study the impact of the model parameters, the Bloom filter parameters h, k and the threshold τ. Further, the proposed method works well for moderate values of the domain size, but there are use cases where this size can be very large, in the order of hundreds of thousands. Alternative methods are necessary in these cases.

References

Apple. Learning with privacy at scale. Technical report, Apple, 2017.

CDC and Teboul, A. Diabetes health indicators dataset. URL www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset.

Cormode, G., Maddock, S., and Maple, C. Frequency estimation under local differential privacy. In *VLDB*, 2021.

Dwork, C. and Roth, A. The algorithmic foundationsof differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9:211–407, 2014.

Erlingsson, U., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-presernving ordinal response. In *CSS*, 2014.

Kraska, T., Beutel, A., Chi, E., Dean, J., and Polyzotis, N. The case for learned index structures. In *SIGMOD*, 2018.

Mitzenmacher, M. A model for learned bloom filters,and optimizing by sandwiching. In *NeurIPS*, 2018.

Wang, T., Blocki, J., Li, N., and Jha, S. Locally differentially private protocolsfor frequency estimation. In *26th USENIX Security Symposium*, 2017.