Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy

Thông T. Nguyễn[1]   Xiaokui Xiao[1]   Yin Yang[2]   Siu Cheung Hui[1]   Hyejin Shin[3]   Junbum Shin[3]

[1]Nanyang Technological University   [2]Hamad Bin Khalifa University
{s140046, xkxiao, asschui}@ntu.edu.sg   yyang@qf.org.qa

[3]Samsung Electronics
{hyejin1.shin, junbum.shin}@samsung.com

ABSTRACT

Organizations with a large user base, such as Samsung and Google,   However, doing so raises   and risks accidental privacy breaches with serious consequences.   techniques address this problem by   with guarantees of plausible deniability; meanwhile,

So far, existing LDP solutions either have severely restricted functionality, or focus mainly on theoretical aspects such as asymptotical bounds rather than practical usability and performance. Motivated by this, we propose Harmony, a practical, accurate and efficient system for collecting and analyzing data from smart device users, while satisfying LDP. Harmony applies to multi-dimensional data containing both numerical and categorical attributes, and supports both basic statistics (e.g., mean and frequency estimates), and complex machine learning tasks (e.g., linear regression, logistic regression and SVM classification). Experiments using real data confirm Harmony's effectiveness.

1. INTRODUCTION

Smart devices connected to the Internet, including mobile phones, wearables, home appliances, sensors and vehicles, have become a part of everyday life in many parts of the world. The data collected by these devices could be an invaluable asset to hardware designers and application developers. For instance, a smartphone maker with its own customized UI such as Samsung TouchWiz could learn about the usage patterns of the various UI features such as Multi Window and One-Handed Mode, and focus on improving the popular ones. However, privacy concerns remain a major hurdle in collecting users' data. For example, the user may not want others to know her web browsing history, apps installed and locations visited. Even if the user (reluctantly) allows trusted organizations to collect her data, the possession of large amounts of sensitive personal data poses a major security risk.   which happened to AOL[1], Netflix[2] and Ashley Madison[3],   . So, organizations with a large user base commonly face a dilemma: either they collect users' personal data and become exposed to the risk of privacy breaches, or they do not collect such data and lose the opportunity of mining them.

Local differential privacy   , which has been used in Google's Chrome browser [11], addresses the above dilemma.   Unlike other models of differential privacy [8,9], which publish randomized aggregates but still collect the exact sensitive data,   Meanwhile, LDP satisfies the strong and rigorous privacy guarantees of differential privacy, i.e.,   .   in its Chrome browser, namely   [11],   A classic example is to collect statistics about a sensitive group (e.g., communists in the US), in which the aggregator asks each individual: "Are you a communist?" To answer this question, each individual tosses a coin, gives the true answer if it is a head, and a random yes/or answer otherwise. Clearly, this randomized approach provides plausible deniability to the individuals.

Based on the collected randomized answers, the aggregator estimates the percentage of users whose true answer is "yes" (resp. "no"). Besides simple counting, a follow-up paper [12] shows that Rappor can also compute other types of statistics such as joint-distribution estimation and association testing. However, three major limitations remain:

---

[1]https://en.wikipedia.org/wiki/AOL_search_data_leak
[2]http://www.wired.com/2009/12/netflix-privacy-lawsuit/
[3]http://edition.cnn.com/2015/08/27/opinions/yang-ashley-madison-hack/

e.g., the average running time of an app.

as we show later in Section 3.2, to compute aggregates on d independent attributes, the error of Rappor grows linearly with d, which is sub-optimal [5, 7]. Third, such as logistic regression and SVM classification.

LDP has also drawn considerable interest from the theory community. However, as we review in Section 6, their focus lies mostly in analyzing the asymptotical performance of basic building blocks of LDP, rather than practical systems and performance. In other words, the problem they study is "what is the best that LDP can do", rather than "how to design a practical LDP solution". Perhaps due to this reason, as we show in Section 3.1, the state-of-the-art solution (to our knowledge) for estimating mean values of multiple attributes under LDP is buggy and requires a fix. Meanwhile, we are not aware of any systematic approach that can perform common machine learning tasks under LDP on a large data domain containing both numeric and categorical attributes.

Motivated by this, we have been building Harmony, an advanced data analytics tool that conforms to LDP requirements. Harmony supports a multitude of common data analysis tasks over an arbitrary number of numerical or categorical attributes. Further, Harmony achieves both non-trivial asymptotical error bounds and improved accuracy in practice, compared to the current state of the art. As a case study, we show how Harmony can improve diagnostic information reporting in Samsung smartphones with strong privacy guarantees, as follows.

1.1 Potential Use Case

Samsung currently collects mobile phone usage information from users through a diagnostic tool bundled with the Samsung Android OS[4]. The information collected include the mobile phone's settings (e.g., display settings, whether or not the location functionality is turned on), memory and battery usage, as well as other log data. The tool transmits the data in an unperturbed format to Samsung, and the transmission explicitly requires users' consent. However, as the collected information could potentially reveal sensitive information, it is beneficial to enhance the tool with privacy protection mechanisms, so as to provide rigorous privacy assurance to users.

Towards this end, local differential privacy is an attractive approach due to its strong privacy guarantee, and its practicability that has been demonstrated in Rappor [11]. However, Rappor's LDP mechanism focuses on collecting a single categorical attribute, whereas the data collected by the Samsung diagnostic tool include both numeric ones (e.g., battery usage) and categorical ones (e.g., display settings). Furthermore, Rappor does not support complex learning tasks (e.g., regressions, SVM), whereas such tasks are important for Samsung to build analytical models from the data, e.g., building an early symptom model to predict system errors based on other measurements such as battery usage, memory usage, active applications, etc. This necessitates the development of new data collection technique based on local differential privacy.

---

[4]The tool is available in the system under "Settings > About device > Report diagnostic info".

Table 1: Notations

$pdf(x)$	Probability density at x
$\langle a, b \rangle$	A tuple of a and b
$a\|\|b$	A concatenation of a and b
$\text{Bern}(p)$	A Bernoulli distribution with parameter p
$\log(x)$	Natural logarithm of x
$\text{Dom}(f)$	Domain of function f
$[d]$	Set of natural numbers $\{1, 2, \ldots, d\}$

In the following, Section 2 provides the necessary background on LDP. Sections 3 presents the fundamental LDP mechanisms in Harmony. Section 4 applies Harmony to common data analytics tasks based on empirical risk minimization, including linear regression, logistic regression and SVM classification. Section 5 contains an extensive set of experiments. Section 6 reviews related work. Finally, Section 7 concludes with directions for future work.

2. PRELIMINARIES

In our setting, an *aggregator* (e.g., Samsung) collects data from a set of *users* (e.g., smart device owners), and computes statistical models of the collected data. The goal is to maximize the accuracy of these statistical models, while preserving the privacy of the users. Following the local differential privacy model [2, 7, 11], we assume that the aggregator already knows the identities (e.g., IP addresses) of the users, but not their private data. Formally, let n be the total number of users, and u_i ($1 \le i \le n$) denote the i-th user. Each user u_i's private data is represented by a tuple t_i, which contains d attributes $A_1, A_2, \ldots, A_d$. These attributes can be either numerical or categorical. Without loss of generality, we assume that each numeric attribute has a domain $[-1, 1]$, and each categorical attribute with k distinct values has a discrete domain $\{1, 2, \ldots, k\}$.

To protect privacy, each user u_i first perturbs her tuple t_i using a randomized *perturbation function* f. Then, she sends the perturbed data $f(t_i)$ to the aggregator instead of her true data record t_i. The perturbation function determines the privacy / utility tradeoff. As an extreme case, if $f(t_i) = t_i$ (i.e., no perturbation), the aggregator obtains perfect utility since it computes the statistical models based on the exact data; however, the privacy of the users is completely lost as the aggregator receives their sensitive data. The other extreme is that f simply outputs a random tuple regardless of t_i, which leads to the highest level of privacy and zero utility, i.e., the aggregator learns nothing about the users' data.

Given a privacy parameter $\epsilon > 0$ that controls the privacy-utility tradeoff, we require that f satisfies ϵ-*local differential privacy* (ϵ-LDP) [11], defined as follows:

DEFINITION 1 (ϵ-LOCAL DIFFERENTIAL PRIVACY). *A randomized function f satisfies ϵ-local differential privacy if and only if for any two input tuples $t, t' \in Dom(f)$ and for any possible output t^* of f, we have:*

$$\Pr[f(t) = t^*] \le e^\epsilon \times \Pr[f(t') = t^*].$$

Basically, local differential privacy is a special case of differential privacy [9] where the random perturbation is performed by the users, not by the aggregator. In other words, the aggregator never possesses the exact private data of any

user. According to the above definition, the aggregator, who receives the perturbed tuple t^*, cannot distinguish whether the true tuple is t or another tuple t' with high confidence (controlled by parameter ϵ), regardless of the background information of the aggregator. This provides plausible deniability to the user. Note that in ϵ-LDP, since random perturbation is done at each user, it is possible to achieve *personalized privacy protection* by using different values of the privacy parameter ϵ at different users, depending on their respective privacy requirements. In this paper, we assume a universal ϵ for the ease of presentation and analysis.

1. ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ These are two basic types of statistics. For each numeric attribute A_j, we aim to estimate the mean value of A_j over all n users, $\frac{1}{n}\sum_{i=1}^{n} t_i[A_j]$. For each categorical attribute A_j', we aim to estimate the frequency of each possible value of A_j', i.e., a histogram. Note that it is also possible to build such a histogram for a numeric attribute with a finite number of possible values.

2. ⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ These are advanced statistics commonly used in machine learning. Examples include linear regression, logistic regression, and support vector machines (SVM) [3].

Unless otherwise specified, all expectations in this paper are taken over the random choices made by the algorithms considered.

Remark. In practice, the tuple t_i of a user may change overtime (e.g., the phone usage information of a user would change day by day); accordingly, the aggregator may want to re-collect information from users after a certain time period (e.g., one week). In that case, we aim to ensure that the collection of each individual snapshot of t_i satisfies ϵ-differential privacy. One may argue that it is more desirable to ensure that all collected snapshots jointly achieve ϵ-differential privacy, but to our knowledge, this is an open problem when the number of snapshots to be collected can be arbitrarily large.

3. ESTIMATING MEANS AND FREQUENCIES

This section investigates the design of the perturbation function f to support accurate estimation of mean values (resp. frequencies) of numeric (resp. categorical) attributes. For ease of exposition, Section 3.1 considers the case when all attributes $A_1, A_2, \ldots, A_d$ in the users' data have a numeric domain $[-1, 1]$; after that, Section 3.2 extends our discussions to the case when both numeric and categorical attributes are present.

3.1 Estimating Mean Values for Numeric Attributes

Given a tuple t_i containing d numeric attributes, a naive design of the perturbation function f is to apply the *Laplace Mechanism* [8]. In particular, let $t_i^* = f(t_i)$ be the perturbed tuple of user u_i, we have:

$$\forall j \in [d], t_i^*[A_j] = t_i[A_j] + \text{Lap}\left(\frac{2d}{\epsilon}\right),$$

where $\text{Lap}(\lambda)$ denotes a random variable that follows a Laplace distribution of scale λ, with the following probability density function:

$$pdf(x) = \frac{1}{2\lambda}\exp\left(-\frac{|x|}{\lambda}\right).$$

Once the aggregator receives all perturbed tuples, it simply computes their ⬚⬚⬚⬚ $\frac{1}{n}\sum_{i=1}^{n} t_i^*[A_j]$ as an estimate of the mean of A_i. ⬚⬚⬚⬚⬚⬚ Meanwhile, it is easy to calculate that ⬚⬚⬚⬚⬚

⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚⬚ Note that this is a fundamental problem that also exists in the traditional differential privacy setting, when publishing statistics for multiple independent attributes. Perhaps rather surprisingly, this problem has not received much attention in the differential privacy literature; the first and only solution we are aware of is proposed by Duchi *et al.* [6, 7] under the local differential privacy setting, presented below.

Duchi *et al.*'s method. Algorithm 1 shows the pseudo-code of Duchi *et al.*'s method. The authors claim that this method satisfies ϵ-local differential privacy, yields unbiased estimates for the mean value of each attribute, and incurs $O\left(\sqrt{d\log d}/(\epsilon\sqrt{n})\right)$ expected error for each attribute, which is proven to be asymptotically optimal. As we explain later, all three claims are incorrect, i.e., their method can violate differential privacy, lead to a biased estimate and incur a much higher amount of error. It takes as input the exact tuple $t_i \in [-1, 1]^d$ of user u_i and a privacy parameter ϵ, and outputs a perturbed vector $t_i^* \in \{-B, B\}^d$, where B is a constant decided by d and ϵ. Note that the output is binary for each attribute, i.e., it is either B or $-B$. Therefore, it suffices for each user to transmit only one bit for each attribute to the aggregator.

Upon receiving the perturbed tuples, the aggregator simply computes the average value for each attribute over all users, and outputs these averages as the estimates of the mean values for their corresponding attributes. Next we focus on the calculation of B, which is rather complicated. Essentially, B is a scaling factor to ensure that the expected value of a perturbed attribute is the same as that of the exact attribute value. First, we calculate:

$$C_d = \begin{cases} 2^{d-1}, & \text{if } d \text{ is odd} \\ 2^{d-1} - \frac{1}{2}\binom{d}{d/2}, & \text{otherwise} \end{cases} \tag{1}$$

Then, B is calculated by:

$$B = \begin{cases} \dfrac{2^d + C_d \cdot (e^\epsilon - 1)}{\binom{d-1}{(d-1)/2} \cdot (e^\epsilon - 1)}, & \text{if } d \text{ is odd} \\[3mm] \dfrac{2^d + C_d \cdot (e^\epsilon - 1)}{\binom{d-1}{d/2} \cdot (e^\epsilon - 1)}, & \text{otherwise} \end{cases} \tag{2}$$

Duchi *et al.* show that $\frac{1}{n}\sum_{i=1}^{n} t_i^*[A_j]$ is an unbiased estimator of the mean of A_j, and

$$\mathbb{E}\left[\max_{j\in[d]}\left|\frac{1}{n}\sum_{i=1}^{n} t_i^*[A_j] - \frac{1}{n}\sum_{i=1}^{n} t_i[A_j]\right|\right] = O\left(\frac{\sqrt{d\log d}}{\epsilon\sqrt{n}}\right), \tag{3}$$

Algorithm 1: Duchi *et al.*'s Method [6,7]

input : tuple $t_i \in [-1,1]^d$ and privacy parameter ϵ.
output: tuple $t_i^* \in \{-B, B\}^d$.

1 Generate a random tuple $v \in \{-1,1\}^d$ by sampling each $v[A_j]$ independently from the following distribution:

$$\Pr[v[A_j] = x] = \begin{cases} \frac{1}{2} + \frac{1}{2} t_i[A_j], & \text{if } x = 1 \\ \frac{1}{2} - \frac{1}{2} t_i[A_j], & \text{if } x = -1 \end{cases} \quad ;$$

2 Let T^+ (resp. T^-) be the set of all tuples $t^* \in \{-B, B\}^d$ such that $t^* \cdot v > 0$ (resp. $t^* \cdot v \le 0$);
3 Sample a Bernoulli variable u that equals 1 with $e^\epsilon/(e^\epsilon + 1)$ probability;
4 **if** $u = 1$ **then**
5   **return** a tuple uniformly at random from T^+;
6 **else**
7   **return** a tuple uniformly at random from T^-;

---

which is asymptotically optimal [7].

Problems in Duchi *et al.*'s method and a possible fix. We implemented and evaluated Duchi *et al.*'s method, but found that whenever the number d of attribute is even, the method yields a biased estimation of the mean of each attribute and incurs significant error. Then, we also found that it violates differential privacy when d is even. To illustrate, consider that $d = 2$ and we have an input tuple $t_i = \langle 1, 1 \rangle$, i.e., $t_i[A_1] = t_i[A_2] = 1$. Then, Line 1 in Algorithm 1 would generates a tuple $v = \langle 1, 1 \rangle$. Let B be as defined in Equation (2), and T^+ and T^- be as defined in Line 2 in Algorithm 1. It can be verified that T^+ and T^- contain 1 and 3 tuples, respectively, with

$$T^+ = \{ \langle B, B \rangle \}, \text{ and}$$
$$T^- = \{ \langle -B, -B \rangle, \langle -B, B \rangle, \langle B, -B \rangle \}.$$

Then, by Lines 3-8 in Algorithm 1, the method outputs $\langle B, B \rangle$ with $\frac{e^\epsilon}{e^\epsilon + 1}$ probability. In contrast, each tuple in T^- has only $\frac{1}{3e^\epsilon + 3}$ probability to be output.

Now consider another input tuple $t_i' = \langle -1, -1 \rangle$. It follows that, for t_i', the algorithm outputs $\langle B, B \rangle$ with only $\frac{1}{3e^\epsilon + 3}$ probability. As a consequence,

$$\Pr[f(t_i) = \langle B, B \rangle] = 3e^\epsilon \cdot \Pr\left[f(t_i') = \langle B, B \rangle\right]$$
$$> e^\epsilon \cdot \Pr\left[f(t_i') = \langle B, B \rangle\right],$$

which indicates that the algorithm does not satisfy ϵ-differential privacy.

We find that the above problem is caused by Line 3 in Algorithm 1, in that the Bernoulli variable u is incorrectly defined for the case for d is even. To address the problem, one possible fix we found is to re-define u as a Bernoulli variable such that

$$\Pr[u = 1] = \frac{e^\epsilon \cdot C_d}{(e^\epsilon - 1)C_d + 2^d}.$$

It can be shown that, with this revised choice of u, Algorithm 1 achieves ϵ-differential privacy and ensures the error bound in Equation 3. We omit the proofs for brevity.

Proposed method. In what follows, we present an algorithm used in Harmony for perturbing a tuple that is conceptually simpler than Duchi *et al.*'s method, but achieves the

---

Algorithm 2: Proposed Method for Handling Numeric Attributes

input : tuple $t_i \in [-1,1]^d$ and privacy parameter ϵ.
output: tuple $t_i^* \in \left\{ -\frac{e^\epsilon+1}{e^\epsilon-1}d, \ 0, \ \frac{e^\epsilon+1}{e^\epsilon-1}d \right\}^d$.

1 Let $t_i^* = \langle 0, 0, \ldots, 0 \rangle$;
2 Sample j uniformly at random from $[d]$;
3 Sample a Bernoulli variable u such that

$$\Pr[u = 1] = \frac{t_i[A_j] \cdot (e^\epsilon - 1) + e^\epsilon + 1}{2e^\epsilon + 2} \quad ;$$

4 **if** $u = 1$ **then**
5    $t^*[A_j] = \frac{e^\epsilon+1}{e^\epsilon-1} \cdot d$;
6 **else**
7    $t^*[A_j] = -\frac{e^\epsilon+1}{e^\epsilon-1} \cdot d$;
8 **return** t_i^*

---

same privacy assurance and asymptotic error bound. Furthermore, our experiments (in Section 5) show that the algorithm slightly outperforms Duchi *et al.*'s method in terms of the empirical accuracy of the estimated means of numeric attributes. Additionally, our method is more efficient: in particular, each user only needs to transmit one bit to the aggregator, which is clearly optimal. Our algorithm is inspired by an existing approach [2] for publishing categorical data, which we will discuss in Section 3.2.

Algorithm 2 shows the pseudo-code of our method. Given a tuple $t_i \in [-1,1]^d$, the algorithm returns a perturbed tuple t_i^* that has non-zero value on only one attribute A_j ($j \in [d]$). Specifically, A_j is selected uniformly at random from all d attributes of t_i, and $t_i^*[A_j]$ is sampled from the following distribution:

$$\Pr\left[t_i^*[A_j] = x\right] = \begin{cases} \frac{t_i[A_j] \cdot (e^\epsilon-1)+e^\epsilon+1}{2e^\epsilon+2}, & \text{if } x = \frac{e^\epsilon+1}{e^\epsilon-1} \cdot d \\ \frac{-t_i[A_j] \cdot (e^\epsilon-1)+e^\epsilon+1}{2e^\epsilon+2}, & \text{if } x = -\frac{e^\epsilon+1}{e^\epsilon-1} \cdot d \end{cases}$$
$$(4)$$

Observe that in the above method, the output t_i^* contains only one non-zero value, for the randomly chosen attribute A_j. This value is binary; hence, the user u_i only needs to transmit 1 bit to the aggregator indicating its sign, and the latter can re-scale it using parameters ϵ and d. Further, as we show below the correctness of this method does not depend on the choice of A_j as long as it is chosen uniformly at random. Therefore, the value of j can be obtained, e.g., using a public source of random numbers such as a hash value of the user's ID. Therefore, the communication overhead between each user and the aggregator is exactly 1 bit.

The following lemmas establish the theoretical guarantees of Algorithm 2.

LEMMA 1. *Algorithm 2 satisfies ϵ-local differential privacy.*

PROOF. Let t^* be an output of Algorithm 2, and A_j be the only attribute such that $t^*[A_j] \ne 0$. Let t and t' be any two tuples, and u (resp. u') be the Bernoulli variable generated in Line 3 of Algorithm 2 given t (resp. t') as the input. In the following, we focus on the case when $t^*[A_j] = \frac{e^\epsilon+1}{e^\epsilon-1}d$; the case when $t^*[A_j] = -\frac{e^\epsilon+1}{e^\epsilon-1}d$ can be analyzed in a similar manner.

By Algorithm 2, we have

$$\frac{\Pr[t^* \mid t]}{\Pr[t^* \mid t']} = \frac{1/d \cdot \Pr[u=1 \mid t]}{1/d \cdot \Pr[u'=1 \mid t']} \le \frac{\max_t \Pr[u=1 \mid t]}{\min_{t'} \Pr[u'=1 \mid t']}$$

$$= \frac{\max_{t[A_j] \in [-1,1]} (t[A_j] \cdot (e^\epsilon - 1) + e^\epsilon + 1)}{\min_{t'[A_j] \in [-1,1]} (t'[A_j] \cdot (e^\epsilon - 1) + e^\epsilon + 1)} = e^\epsilon.$$

This completes the proof. $\square$

LEMMA 2. *Let t_i^* be the output of Algorithm 2 given an input tuple t_i. Then, for any $j \in [d]$, $\mathbb{E}[t^*[A_j]] = t[A_j]$.*

PROOF. By Equation (4),

$$\mathbb{E}[t^*[A_j]] = \Pr\left[t^*[A_j] = \frac{e^\epsilon+1}{e^\epsilon-1} \cdot d\right] \cdot \frac{e^\epsilon+1}{e^\epsilon-1} \cdot d$$
$$+ \Pr\left[t^*[A_j] = -\frac{e^\epsilon+1}{e^\epsilon-1} \cdot d\right] \cdot \left(-\frac{e^\epsilon+1}{e^\epsilon-1} \cdot d\right)$$
$$+ \Pr\left[t^*[A_j] = 0\right] \cdot 0$$
$$= \frac{2t[A_j] \cdot (e^\epsilon - 1)}{2e^\epsilon - 2} = t[A_j].$$

$\square$

By Lemma 2, the server can use $\frac{1}{n} \sum_{i=1}^{n} t^*[A_j]$ as an unbiased estimator of the mean of A_j. The following lemma shows the accuracy guarantee of this estimator.

LEMMA 3. *For any $j \in [d]$, let $Z[A_j] = \frac{1}{n} \sum_{i=1}^{n} t_i^*[A_j]$ and $X[A_j] = \frac{1}{n} \sum_{i=1}^{n} t_i[A_j]$. With at least $1 - \beta$ probability,*

$$\max_{j \in [d]} |Z[A_j] - X[A_j]| = O\left(\frac{\sqrt{d \log(d/\beta)}}{\epsilon \sqrt{n}}\right).$$

PROOF. First, observe that for any $i \in [d]$ and any $j \in [d]$, the variance of $t_i^*[a_j] - t_i[a_j]$ equals:

$$\mathrm{Var}[t_i^*[a_j] - t_i[a_j]] = \mathrm{Var}[t_i^*[a_j]]$$
$$= \mathbb{E}\left[(t_i^*[a_j])^2\right] - (\mathbb{E}[t_i^*[a_j]])^2$$
$$= \frac{1}{d}\left(\frac{e^\epsilon+1}{e^\epsilon-1} \cdot d\right)^2 - (t_i[a_j])^2 \le \left(\frac{e^\epsilon+1}{e^\epsilon-1}\right)^2 \cdot d.$$

By Bernstein's inequality,

$$\Pr\left[|Z[a_j] - X[a_j]| \ge \lambda\right]$$
$$\le 2 \cdot \exp\left(-\frac{n\lambda^2}{\frac{2}{n}\sum_{i=1}^{n}\mathrm{Var}[t_i^*[a_j] - t_i[a_j]] + \frac{2}{3}\lambda \cdot \frac{e^\epsilon+1}{e^\epsilon-1} \cdot 2d}\right)$$
$$= 2 \cdot \exp\left(-\frac{n\lambda^2}{2d \cdot (O(1/\epsilon^2) + \lambda \cdot O(1/\epsilon))}\right).$$

By the union bound, there exists $\lambda = O\left(\frac{\sqrt{d \log(d/\beta)}}{\epsilon \sqrt{n}}\right)$ such that $\max_{j \in [d]} |Z[a_j] - X[a_j]| < \lambda$ holds with at least $1 - \beta$ probability. $\square$

3.2 Estimating Frequencies for Categorical Attributes

We now focus on the case where each user's data record contains not only numeric attributes but also categorical ones. For each categorical attribute, the aggregator aims to build an accurate histogram containing the frequency estimate for each possible value in the attribute's domain. For example, Samsung may want to know the percentage of users who enable a specific setting, through the diagnostic information report app described in Section 1.1. Note that we can

---

Algorithm 3: Bassily and Smith's method [2]

input : $t_i[A_j] \in [k]$ for each user u_i, privacy budget ϵ, confidence of the error bound β
output: Frequency estimate for each of the k values in attribute A_j

1 Compute $\gamma = \sqrt{\frac{\log(2k/\beta)}{\epsilon^2 n}}$ and $m = \frac{\log(k+1)\log(2/\beta)}{\gamma^2}$;

2 Generate random matrix $\Phi \in \{\pm \frac{1}{\sqrt{m}}\}^{m \times k}$;

3 **for** $i = 1$ *to* n **do**

4   User u_i: draw $s \sim \mathrm{Uniform}(\{1, 2 \ldots, m\})$;

5   User u_i: draw $t \sim \mathrm{Bern}(\frac{e^\epsilon}{e^\epsilon+1})$;

6   User u_i: **if** $t = 1$ **then** $\alpha = c_\epsilon m \Phi[s, t_i[A_j]]$ **else** $\alpha = -c_\epsilon m \Phi[s, t_i[A_j]]$, where $c_\epsilon = \frac{e^\epsilon+1}{e^\epsilon-1}$;

7   User u_i: submit $\langle s, \alpha \rangle$, which represents a k-dimensional vector z_i where the s-th entry is α and the other entries are 0;

8 Compute $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$;

9 **for** $l = 1$ *to* k **do**

10   Estimate the frequency of the l-th value by the inner product of the l-th column of Φ and $\bar{z}$;

11 **return** k frequency estimates obtained above;

---

convert a numeric attribute a categorical one (e.g., display brightness can be discretized to three levels: low, medium and high) and build a histogram accordingly.

Randomized response for binary attributes. For a single binary attribute (e.g., WiFi on/off), it suffices to use the classic randomized response method [20] (also used in Rappor [11]) to estimate the distribution of users. Specifically, suppose that the domain of the binary attribute (let A_j) contains two possible values, -1 and $+1$. Each user u_i reports her true answer $t_i[A_j]$ with probability p, and a random answer with probability $1 - p$. The latter has the same probability to be -1 and $+1$; hence, its expected value is zero. Therefore, the expected value for u_i's reported value is $p \cdot t_i[A_j]$; thus, we can obtain an unbiased estimate by multiplying the reported value by a scaling factor $c_\epsilon = 1/p$.

Meanwhile, comparing u_i's true attribute value and her reported one, the two are the same with probability $p + (1-p)/2$, and they are different with probability $(1 - p)/2$. According to Definition 1, ϵ-local differential privacy requires that $\frac{p+(1-p)/2}{(1-p)/2} \le e^\epsilon$. The equality holds when $p = \frac{e^\epsilon-1}{e^\epsilon+1}$. We thus arrive at the following unbiased mechanism that satisfies ϵ-LDP: each user u_i reports $c_\epsilon \cdot t_i[A_j] = \frac{e^\epsilon+1}{e^\epsilon-1} \cdot t_i[A_j]$ with probability $p+(1-p)/2 = \frac{e^\epsilon}{e^\epsilon+1}$, and $-c_\epsilon \cdot t_i[A_j]$ otherwise (i.e., with probability $\frac{1}{e^\epsilon+1}$).

Once the aggregator receives all reported values for attribute A_j, it computes the average over all users, which is an estimate of the mean value $\mathbb{E}[A_j]$ for A_j. Since A_j can be either $+1$ or -1, the percentage of users with $+1$ (resp. -1) is $\frac{1+\mathbb{E}[A_j]}{2}$ (resp. $\frac{1-\mathbb{E}[A_j]}{2}$).

Bassily and Smith's method. The problem is more complicated when the categorical attribute A_j contains $k > 2$ possible values. In this situation, the aggregator aims to build a histogram that contains the estimated frequency for each of the k possible values. The current state-of-the-art to our knowledge is by Bassily and Smith [2], shown in Algorithm 3, which is proven to satisfy ϵ-LDP and achieve an optimal asymptotical error bound. There are two main ideas in this method. First, the authors assume that the num-

ber of possible values k in the categorical attribute is far larger than the number of users n; hence, the method applies random projection to reduce the dimensionality from k to m. The value of m is chosen carefully so as to obtain the asymptotically optimal error bound. This step essentially transforms the categorical attribute into m binary ones.

Specifically, the random projection is done with a $m \times k$ matrix Φ in which each element is randomly set to either $+\frac{1}{\sqrt{m}}$ or $-\frac{1}{\sqrt{m}}$ with equal probability. This ensures that (i) the inner product of any column in Φ with itself is 1 (which is where the absolute value of each element $\frac{1}{\sqrt{m}}$ comes from) and (ii) the inner product of two different columns in Φ has zero expected value, since the signs are randomly generated. Each user u_i's attribute value $t_i[A_j]$ is then transformed to m binary values by taking the $t_i[A_j]$-th column in Φ.

The second idea is for each user to randomly pick one of the converted m binary attributes, and report a randomized response using the method described earlier. Note that the randomized response needs to be scaled by a factor of m, since each of the m binary attributes has probability $1/m$ to be chosen. The aggregator collects the average for each of the m binary attributes, which is stored as a vector $\bar{z}$. To obtain the frequency estimate of a particular attribute value l, the method takes the inner product of $\bar{z}$ and the l-th column of Φ, which can be proven to yield an unbiased frequency estimate for the l-th value in A_j. Meanwhile, the frequency estimate is proven to be within $O\left(\frac{\sqrt{\log(k/\beta)}}{\epsilon\sqrt{n}}\right)$ error with probability $1 - \beta$ [2], where β is an input to the algorithm.

Proposed method for a single categorical attribute. Harmony generally follows Bassily and Smith's method to estimate value frequencies for a categorical attribute. However, we found that although Bassily and Smith's method achieves optimal asymptotic accuracy, in practice its accuracy tends to be unstable, especially for relatively small categorical domains. The reason is that the random projection matrix Φ introduces considerable noise; in particular, the inner product of two different columns is often non-zero unless k is very large, which is magnified by a large number of users n. Hence, we propose an alternative solution that obtains higher accuracy when $k = o(n)$. In particular, instead of generating random matrix Φ (size $m \times k$ where $m = O(n)$), we construct a binary matrix of size $k \times k$ satisfying that any two column vectors are always orthogonal. The construction algorithm of this matrix can be found in the appendix.

Proposed method for multiple numeric and categorical attributes. Bassily and Smith's method is limited to a single categorical attribute. To extend it to multiple categorical attributes, a straightforward approach is to apply the method once for each attribute separately. In that case, however, the privacy budget ϵ needs to be divided among all attributes, so as to ensure ϵ-LDP as a whole. Without loss of generality, assume that we have d categorical attributes, and we assign ϵ/d budget to each of them. Then, the amount of noise incurred by Bassily and Smith's method on each attribute is increased d times to $O\left(\frac{d\sqrt{\log(d/\beta)}}{\epsilon\sqrt{n}}\right)$, which is unsatisfactory when d is large.

Another approach for extension is to (i) convert the d categorical attributes into a "composite" attribute whose domain equals the Cartesian product of the individual attribute domains, and then (ii) apply Bassily and Smith's method on the composite attribute. This, however, only allows us to (accurately) derive the frequency each composite value (i.e., combination of values from all d individual attributes), but does not provide quality estimation of the frequency of each individual value. Note that these limitations are not specific to Bassily and Smith's method; to our knowledge, there is no existing work (including Rappor) that are designed to handle multiple categorical attributes, let alone a mixture of numeric and categorical ones.

In Harmony, we use a simple and elegant solution to handle multiple attributes: for each numerical attribute, the aggregator estimates its mean value; for each categorical attribute, the aggregator estimates its value frequencies. In particular, given d attributes $A_1, A_2, \ldots, A_d$, the solution asks each user to perform the following:

1. Draw j uniformly at random from set $\{1, 2, \ldots, d\}$;

2. If A_j is a numeric attribute, then submit a noisy version of $t[A_j]$ computed as in Lines 3-7 of Algorithm 2;

3. Otherwise (i.e., A_j is a categorical attribute), compute $\langle s, \alpha \rangle$ as in Lines 4-8 of Algorithm 3, then submits $\langle s, d \cdot \alpha \rangle$ to represent a d-dimensional vector where the s-th entry is $d \cdot \alpha$ and all other entries are zero.

The above solution satisfies ϵ-LDP, which follows from the fact that (i) each user randomly selects one attribute to submit, and (ii) the algorithm used for submitting the selected attribute is ϵ-differentially private. For each numeric attribute, it is easy to see that the solution provides the same accuracy guarantee as Algorithm 2, since both methods handle numeric attributes in exactly the same way. The following lemma states the accuracy guarantee of our solution for categorical attributes.

LEMMA 4. *For each categorical attribute A_j with a domain $\{1, 2, \ldots, k\}$, let x_l be the frequency of the l-th value of A_j, and y_l be the estimation of x_l returned by our solution. With at least $1 - \beta$ probability,*

$$\max_{l \in [k]} |y_l - x_l| = O\left(\frac{\sqrt{d\log(k/\beta)}}{\epsilon\sqrt{n}}\right).$$

PROOF (SKETCH). Consider any user with a tuple t. With respect to A_j, our solution can be regarded as a method that (i) outputs nothing with $d - 1/d$ probability, and (ii) with the remaining $1/d$ probability, applies Bassily and Smith's algorithm on A_j and scale its output up by d times. It follows that the variance of each of our frequency estimators for A_j is d times that of Bassily and Smith's estimator. Based on the analysis in [2], it can be shown that the variance of Bassily and Smith's estimator is $O\left(\frac{1}{n\epsilon^2}\right)$. Therefore, the variance of each of our frequency estimators is $O\left(\frac{d}{n\epsilon^2}\right)$. Combining this with Bernstein's inequality and the union bound, it can be proven that with at least $1 - \beta$ probability, the error in any k of our estimators is $O\left(\frac{\sqrt{d\log(k/\beta)}}{\epsilon\sqrt{n}}\right)$. $\square$

By Lemma 4, when there exist multiple categorical attributes, the error incurred by our approach is a factor of $O(\sqrt{d})$ smaller than that of a solution that repeatedly apply Bassily and Smith's method on each categorical attribute.

4. BUILDING MACHINE LEARNING MODELS USING STOCHASTIC GRADIENT DESCENT

This section investigates building a large class of machine learning models that can be expressed as empirical risk minimization under ϵ-local differential privacy. In particular, we focus on three common types of learning tasks: linear regression, logistic regression, and SVM classification. Section 4.1 introduces the basic approaches for building these models, while Section 4.2 discusses optimizations that lead to improved results in practice.

4.1 Basic Methods

Suppose that each user u_i has a pair $\langle x_i, y_i \rangle$, where $x_i \in [-1, 1]^d$ and $y_i \in [-1, 1]$ (for linear regression) or $y_i \in \{-1, 1\}$ (for logistic regression and SVM classification). Let $\ell(\cdot)$ be a *loss function* that (i) maps a d-dimensional *parameter vector* β into a real number and (ii) is parameterized by x_i and y_i. We aim to identify a parameter vector β^* such that

$$\beta^* = \arg\min_\beta \frac{1}{n} \left(\sum_{i=1}^n \ell(\beta; x_i, y_i) \right) + \frac{\lambda}{2} \|\beta\|_2^2,$$

where $\lambda > 0$ is a regularization parameter. We consider three specific loss functions:

1. Linear regression: $\ell(\beta; x_i, y_i) = (x_i^T \beta - y_i)^2$;

2. Logistic regression: $\ell(\beta; x_i, y_i) = \log\left(1 + e^{-y_i x_i^T \beta}\right)$;

3. SVM (hinge loss): $\ell(\beta; x_i, y_i) = \max\{0, 1 - y_i x_i^T \beta\}$.

For convenience, we define

$$\ell'(\beta; x_i, y_i) = \ell(\beta; x_i, y_i) + \frac{\lambda}{2} \|\beta\|_2^2.$$

One of the most common solutions to compute β^* is *stochastic gradient descent (SGD)*. It starts from an initial parameter vector β_0, and iteratively updates it into $\beta_1, \beta_2, \ldots$ based on the following equation:

$$\beta_{k+1} = \beta_k - \gamma_k \cdot \nabla \ell'(\beta_k; x, y),$$

where $\langle x, y \rangle$ is the tuple of a randomly selected user, $\nabla \ell'(\beta_k; x, y)$ is the gradient of ℓ' at β_k, and γ_k is a constant typically set to $O(1/\sqrt{k})$. It terminates when the difference between β_{k+1} and β_k is sufficiently small.

Under our problem setting, however, $\nabla \ell'$ is not directly available to the aggregator, and needs to be collected in a private manner. Towards this end, existing work [7, 13] has suggested that the aggregator may ask the selected user in each iteration to submit a noisy version of $\nabla \ell'$, by using the Laplace mechanism or Duchi *et al.*'s method (i.e., Algorithm 1). We can straightforwardly improve these existing approaches by perturbing $\nabla \ell'$ using Algorithm 2 instead; however, we observe that such a solution is insufficient for our target application, as we explain in Section 4.2.

4.2 Improvements

Mini-batching. We observe in our experiments that the aforementioned SGD approach yields rather inaccurate results, due to the noise injected in the gradient $\nabla \ell'$ returned by each user. In particular, if each user applies Algorithm 2 to compute $\nabla \ell'(\beta_k; x, y)$, the amount of noise in $\nabla \ell'$ is $O\left(\frac{\sqrt{d \log d}}{\epsilon}\right)$, which is excessively large given that $x \in [-1, 1]^d$. To address this issue, we adopt *mini-batch* gradient descent instead of SGD. That is, each iteration of the algorithm, we involve a group G of users, and ask each of them to submit a noisy version of the gradient; after that, we update the parameter vector β_k with the mean of the noisy gradients, i.e.,

$$\beta_{k+1} = \beta_k - \gamma_k \cdot \frac{1}{|G|} \sum_{i=1}^{|G|} \nabla \ell_i^*,$$

where $\nabla \ell_i^*$ is the noisy gradient submitted by the i-th user in G. This helps because the amount of noise in the average gradient is $O\left(\frac{\sqrt{d \log d}}{\epsilon \sqrt{|G|}}\right)$, which could be acceptable if $|G| = \Omega\left(d \log d / \epsilon^2\right)$.

However, when d is sizable, $|G| = \Omega\left(d \log d / \epsilon^2\right)$ is large. As a consequence, when we allow each user to participate in at most one iteration of the algorithm, the maximum number of iterations (i.e., $n/|G|$) is small. In that case, the algorithm may terminate prematurely and return an inferior parameter vector. One may attempt to mitigate this problem by allowing each user to be involved in $m > 1$ iterations, but it would further increase the amount of noise in each gradient returned. To explain this, suppose that the i-th ($i \in [1, m]$) gradient returned by the user satisfies ϵ_i-differential privacy. By the composition property of differential privacy [17], if we are to enforce ϵ-differential privacy for the user's data, we should have $\sum_{i=1}^m \epsilon_i \leq \epsilon$. Consider that we set $\epsilon_i = \epsilon/m$. Then, the amount of noise in each gradient becomes $O\left(\frac{m\sqrt{d \log d}}{\epsilon}\right)$; accordingly, the acceptable mini-batch size becomes $|G| = \Omega\left(m^2 d \log d / \epsilon^2\right)$, which is m^2 times the acceptable size when we allow each user to participate in at most one iteration. It then follows that the total number of iterations in the algorithm is inversely proportional to $1/m$, i.e., setting $m > 1$ only degrades the performance of the algorithm.

Dimension reduction. For linear regression, instead of increasing m, we propose to apply dimensionality reduction on each user's data, so as to reach an acceptable size of mini-batches. Specifically, the curator first generates a random $r \times d$ matrix P where $r < d$ and each entry has an equal probability to be assigned $1/d$ or $-1/d$. Then, the curator shares P with all users, and asks each user to convert her tuple $\langle x_i, y_i \rangle$ into a reduced tuple $\langle x_i', y_i \rangle$, where $x_i' = Px$. In other words, we project $\{x_i\}$ into a random r-dimensional sub-space, and such a projection is known to preserve several important characteristics of the original data [1]. It can be verified that $x_i' \in [-1, 1]^r$.

Subsequently, each user uses the reduced tuple $\langle x_i', y_i \rangle$ to participate in the mini-batch gradient descent algorithm. In other words, each noisy gradient $\nabla \ell_i^*$ returned by the user is r-dimensional instead of d-dimensional. As such, the average noisy gradient obtained from a mini-batch G of users has an error of $O\left(\frac{\sqrt{r \log r}}{\epsilon \sqrt{|G|}}\right)$ instead of $O\left(\frac{\sqrt{d \log d}}{\epsilon \sqrt{|G|}}\right)$. Accordingly, the acceptable mini-batch size is reduced to $|G| = \Omega\left(r \log r / \epsilon^2\right)$.

Algorithm 4 shows the pseudo-code of our mini-batch gradient descent method with dimension reduction, in the con-

Algorithm 4: An Iterative Method for Empirical Risk Minimization

input : positive integer r, mini-batch size g, privacy parameter ϵ

output: parameter vector $\alpha \in \mathbb{R}^r$

1 generates a random $r \times d$ matrix P each entry has an equal probability to $1/d$ or $-1/d$;

2 initialize a counter $k = 0$, and learning rate γ;

3 initialize a r-dimensional vector $\nabla = \langle 0, 0, \ldots, 0 \rangle$;

4 **while** *true* **do**

5   **if** *a user with a tuple $\langle x_i, y_i \rangle$ comes online* **then**

6   send P to the user;

7   the user computes $x'_i = P x_i$, and derives $\nabla_i = \nabla \ell'(\alpha; x'_i, y_j)$;

8   **if** $\nabla_i \notin [-1, 1]^r$ **then**

9   the user projects ∇_i onto $[-1, 1]^r$;

10   the user applies Algorithm 2 on ∇_i, and submits a noisy gradient ∇_i^*;

11         $k = k + 1$, and $\nabla = \nabla + \nabla_i^*$;

12   **if** $k \mod g = 0$ **then**

13             $\nabla = \frac{\nabla}{g}$, and $\gamma = 1/\sqrt{k/g}$;

14             $\alpha = \alpha - \gamma \cdot \nabla$;

15   **if** *k is sufficiently large or α changes sufficiently small in the last update* **then**

16   **break**;

17 **return** α;

---

text of the Samsung diagnostic tool. The aggregator first generates a random $r \times d$ matrix P, and maintains a r-dimensional parameter vector α (Lines 1-3). (We use α instead of β to denote the parameter vector to avoid confusion on its dimensionality.) After that, whenever a user with a tuple $\langle x_i, y_i \rangle$ comes online, she obtains P and the current α from the aggregator (Line 6). Then, the user computes a reduced tuple $\langle x'_i, y_i \rangle$, as well as the gradient $\nabla_i = \nabla \ell'(\alpha; x'_i, y_j)$ (Line 7). If any entry of ∇_i is larger than 1 (resp. smaller than -1), then the user resets the entry to 1 (resp. -1) (Lines 8-9). This ensures that $\nabla_i \in [-1, 1]^d$, so that it can be a valid input to Algorithm 2. After that, the user computes a noisy gradient ∇_i^* using Algorithm 2, submits it to the aggregator, and then logs off (Line 10).

The aggregator computes the average noisy gradient from every g users (where g is an input parameter), and updates the parameter vector α accordingly (Lines 11-14). When the update to α is sufficiently small or when a sufficiently large number of users have participated, the aggregator terminates the algorithm (Lines 15-16).

5. EXPERIMENTS

5.1 Experimental Settings

For experimental repeatability, we use two public datasets extracted from the *Integrated Public Use Microdata Series* [14], US and BR, which contains census records from the United States and Brazil, respectively. US contains 9M tuples and 23 attributes, among which 6 are numeric (e.g., age) and 17 are categorical (e.g., gender); BR has 4M records and 18 attributes, among which 6 are numeric and 12 are categorical. Both datasets contain a numeric attribute "total income", which we use as the dependent attribute in linear regression, logistic regression, and SVM. We normalize the domain of each numeric attribute into $[-1, 1]$.

5.2 Estimating Means and Frequencies

In the first set of experiments, we consider the task of collecting a noisy tuple from each user to estimate the mean of each numeric attribute and the frequency of each categorical value. As mentioned in Section 1, none of the existing solutions can directly support this task, since they are designed for either numeric or categorical attributes, but not both. To address this issue, we construct a method (referred to as *Hybrid*) by combining the best existing solutions as follows. Let t be a tuple with d_n numeric attributes and d_c categorical attributes. Given t and a privacy budget ϵ, *Hybrid* first constructs a d_n-dimensional tuple t' that contains all numeric values in t, and then release a noisy version of t' by invoking Duchi *et al.*'s method (see Section 3.1) on t', using a privacy budget of $d_n \epsilon / d$. After that, for each categorical value in t, *Hybrid* applies Bassily and Smith's method (see Section 3.2) to release a noisy version with a privacy budget of ϵ / d. By the composition property of differential privacy [17], *Hybrid* ensures ϵ-LDP. Intuitively, *Hybrid* is a best-effort approach to incorporate two states of the art that are designed only for numeric attributes (i.e., Duchi *et al.*'s method) and a single categorical attribute (i.e., Bassily and Smith's method), respectively.

We apply our solution in Section 3 and *Hybrid* on both US and BR to generate noisy tuples, and then use the noisy tuples to estimate the frequency of each value in each categorical domain in US and BR. For each method, we measure the L_∞ error in the estimated value frequencies, and we take the average measurement from 100 runs. Figures 1a and 1b illustrate the results as ϵ varies. Observe that our method is significantly more accurate than *Hybrid* in all cases, and its error is only around $1/4$ of the error incurred by *Hybrid*. This is consistent with the analysis in Section 3.2 that (i) our method has $O\left(\frac{\sqrt{d \log k}}{\epsilon \sqrt{n}}\right)$ estimation error, and (ii) repeatedly applying Bassily and Smith's method on each categorical leads to $O\left(\frac{d \sqrt{\log k}}{\epsilon \sqrt{n}}\right)$ error.

We also use the noisy tuples to estimate the mean of each numeric attribute, and we measure the L_∞ error of each method, averaged over 100 runs. Figure 1c (resp. 1d) shows the results on US (resp. BR) as a function of the privacy budget ϵ. Observe that our solution slightly outperforms *Hybrid*, regardless of the dataset used and the value of ϵ. We also note that, compared with *Hybrid* (which applies Duchi *et al.*'s method to handle numeric attributes), our solution has a much lower communication cost on each user (since it only requires each user to transfer 1 bit), and is much simpler.

5.3 Empirical Risk Minimization

In the second set of experiments, we consider linear regression, logistic regression, and SVM classification on US and BR. For both datasets, we use the numeric attribute "total income" as the dependent variable, and all other attributes as independent variables. Following the standard practice, we transform each categorical attribute A_j with k values into $k - 1$ binary attributes with a domain $\{-1, 1\}$, such that (i) the l-th ($l < k$) value in A_j is represented by a 1 on the l-th binary attribute and a -1 on each of the remaining $k - 2$ attributes, and (ii) the k-th value in A_j is represented by -1 on all binary attributes. After this transformation, the dimensionality of US (resp. BR) becomes 85 (resp. 95). For logistic regression and SVM (which requires
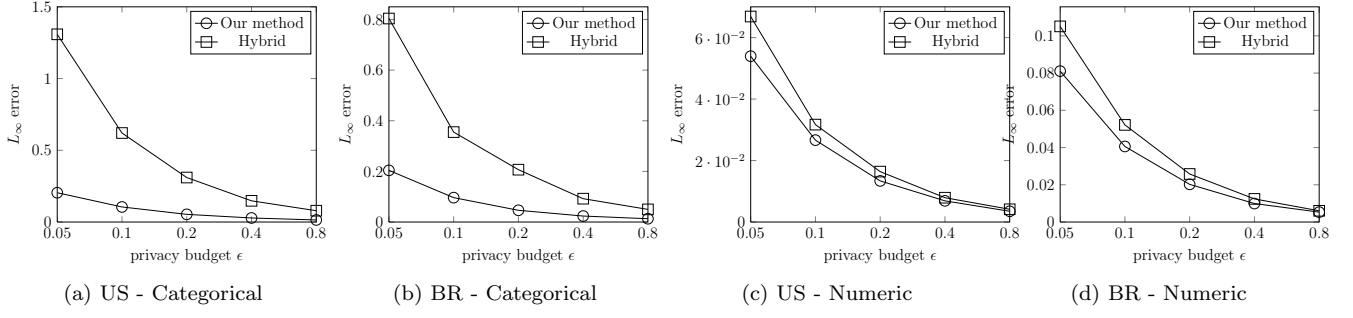
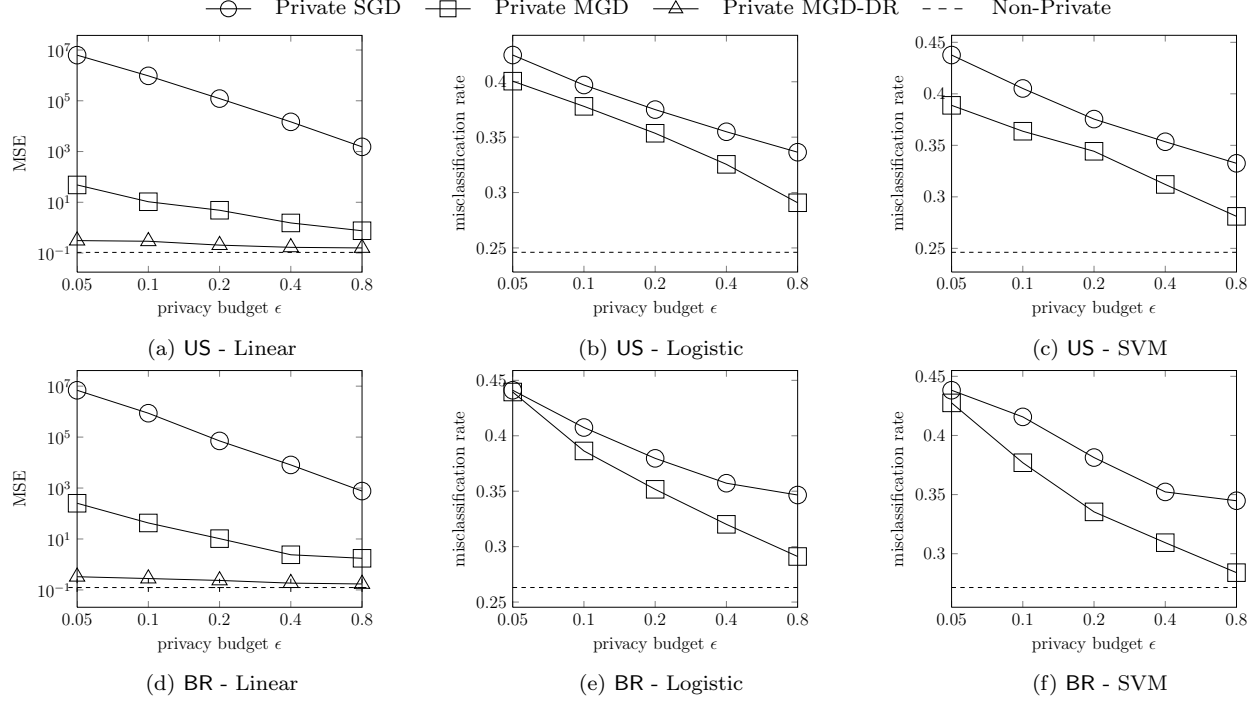Figure 1: Experiments on estimating means and frequencies.

Figure 2: Experimental results for empirical risk minimization.

the dependent variable to be binary), we convert "total income" into a binary attribute by mapping all values which are greater than or equal to the mean to 1, and all other values to -1.

We evaluate four methods: (i) a private version of SGD that involves one user in each iteration, and asks the user to submit a noisy gradient using Duchi *et al.*'s method; (ii) mini-batch gradient descent (MGD), which involves g users in each iteration, and uses the average noisy gradients of those users (generated with Algorithm 2) to update the parameter vector; (iii) MGD with dimension reduction (DR), which is an improved version of MGD that projects the users' data onto an r-dimensional sub-space before the learning task (this method is applied for linear regression only); (iv) a non-private version of SGD. Based on the analysis in Section 4, we set the mini-batch size for MGD and MGD-DR to $g = \max\{2d \log d / \epsilon^2, n/1000\}$ and $g = \max\{2r \log r / \epsilon^2, n/1000\}$, respectively. The term $n/1000$ is to guarantee that our mini-batch size is not too small when ϵ is large. In addition, we set $r = 20$. For all

methods, we set the regularization factor $\lambda = 10^{-4}$.

On each dataset, we use 10-fold cross validation to assess the performance of each method. Figure 2a (resp. 2d) shows the mean squared error (MSE) of the linear regression model generated by each method, under various values of ϵ. Private SGD incurs prohibitive errors in all cases, due to the large amount of noise in the gradient that it obtains in each iteration. MGD alleviates this issue with mini-batches, but still provides unsatisfactory accuracy for linear regression. The reason, as we mentioned in Section 4.2, is that MGD requires using a large mini-batch size g when d is large, which in turn leads to a small total number of iterations and degrades its performance. MGD-DR overcomes the drawback of MGD by incorporating dimension reduction to reduce the required mini-batch size, and hence, it is able to achieve an accuracy that is close to the non-private SGD.

Figures 2b and 2e (2c and 2f) illustrate the misclassification rate of each method for logistic regression (resp. SVM). Overall, our experimental results demonstrate the effectiveness of mini-batches and dimension reduction for empirical

risk minimization under local differential privacy.

6. RELATED WORK

Differential privacy [4, 8, 9] is a strong, mathematically rigorous framework for privacy protection. Unlike earlier privacy-preserving data publication methods which are largely syntactic, differential privacy provides semantic, information-theoretic guarantees on individuals' privacy. Hence, since its proposal in 2003 it had attracted much attention from various research communities in computer science, including theory [10], machine learning [18], data management [21], and systems [19].

Earlier models of differential privacy assume a trusted data curator, who collects and manages the exact private information of individuals, and releases statistics derived from the data under differential privacy requirements. In practice, however, users may not want to share private information with anyone, including the central data curator. Recently, much attention has been shifted to the local differential privacy model, which eliminates the data curator and the collection of exact private information. Specifically, Duchi *et al.* [5] systematically investigate the concept of local differential privacy, propose the minimax framework for LDP based on the information theory, prove upper and lower error bounds of LDP-compliant methods, and analyze the trade-off between privacy and accuracy. Kairouz *et al.* [15] show that a version of randomized response is an optimal mechanism for frequency estimation on a single binary attribute. Kairouz *et al.* [16] study the problem with a categorical attribute with an arbitrary number of possible values, propose two mechanisms: binary and randomized response mechanisms, and prove their optimality when the privacy budget is low and high, respectively. Bassily and Smith [2] propose an asymptotically optimal solution for building succinct histograms over a large categorical domain under LDP.

Erlingsson *et al.* [11] propose the RAPPOR framework, which is based on the randomized response mechanism for publishing a vector of binary values under LDP. They use this mechanism with a Bloom filter, which intuitively adds another level of protection and increases the difficulty for the adversary to infer private information. As a result, it also becomes more difficult derive statistics from the collected data, and they propose a sophisticated solution for this purpose. A follow-up paper [12] extends Rappor to more complex statistics such as joint distributions and association testing, as well as categorical attributes that contain a large number of potential values, such as a user's home page.

7. CONCLUSION

This work systematically investigates the problem of collecting and analyzing users' personal data under ϵ-local differential privacy, in which the aggregator only collects randomized data from the users, and computes statistics based on such data. The proposed solution Harmony is able to collect data records that contain multiple numeric and categorical attributes, and compute accurate statistics from simple ones such as mean and frequency to complex machine learning models such as linear regression, logistic regression and SVM classification. Harmony achieves both optimal asymptotic error bound and high accuracy in practice. Meanwhile, it is highly efficient in terms of communication and compu-

tational overhead. Extensive experiments demonstrate its effectiveness on real data. In the next step, we plan to investigate the application of Harmony in a real use case such as Samsung's diagnostic info report app.

8. REFERENCES

[1] D. Achlioptas. Database-friendly random projections. In *PODS*, 2001.

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

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[4] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

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

[6] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy, data processing inequalities, and minimax rates. *CoRR*, abs/1302.3203, 2013.

[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Privacy aware learning. *J. ACM*, 61(6):38:1–38:57, 2014.

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

[9] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, pages 528–544, 2004.

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

[11] U. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

[12] G. C. Fanti, V. Pihur, and Ú. Erlingsson. Building a RAPPOR with the unknown: Privacy-preserving learning of associations and data dictionaries. *CoRR*, abs/1503.01214, 2015.

[13] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan. Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices. In *ICDCS*, pages 11–20, 2015.

[14] IPUMS. Integrated public use microdata series. https://www.ipums.org.

[15] P. Kairouz, S. Oh, and P. Viswanath. Differentially private multi-party computation: Optimality of non-interactive randomized response. *arXiv preprint arXiv:1407.1546*, 2014.

[16] P. Kairouz, S. Oh, and P. Viswanath. Extremal mechanisms for local differential privacy. In *NIPS*, pages 2879–2887, 2014.

[17] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[18] A. D. Sarwate and K. Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *IEEE Signal Process. Mag.*, 30(5):86–94, 2013.

[19] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series

Algorithm 5: Generation of orthogonal set

---

   input : Dimensionality $d = 2^k$.
   output: Set S of d vectors.
1 $S = \{[1, -1], [1, 1]\}$;
2 **while** $|S| < d$ **do**
3 $S' = \emptyset$;
4   **for** $v \in S$ **do**
5 $S' \leftarrow S' \cup \{v||v, v||(-v)\}$;
6 $S \leftarrow S'$;
7 **return** S;

---

data. In *NDSS*, 2011.

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

[21] Y. Yang, Z. Zhang, G. Miklau, M. Winslett, and X. Xiao. Differential privacy in data publication and analysis. In *SIGMOD*, pages 601–606, 2012.

APPENDIX

LEMMA 5. *The set S returned from Algorithm 5 is an orthogonal set.*

PROOF. We prove this lemma by induction. For the base case, observe that the initial value of $S = \{[1, -1], [1, 1]\}$ is a orthogonal set. Now assume that $S = \{v_i\}$ is a orthogonal set. We will prove that $S' = \{v_i||v_i, v_i||(-v_i)\}$ is also orthogonal set. For any $v'', v' \in S'$, consider the dot product $\langle v'', v' \rangle$, there are two general cases: (i) $\langle v'', v' \rangle = \langle v_i|| \pm v_i, v_j|| \pm v_j \rangle$ and (ii) $\langle v'', v' \rangle = \langle v_i||v_i, v_i|| - v_i \rangle$. In both cases, the inner product equals zero. Thus, lemma is proved. $\square$