Local Differential Privacy for Privacy-preserving Machine Learning

Graham Cormode, gcormode@fb.com, Meta AI

1 Introduction

As discussed in other chapters, Differential Privacy (DP) provides a widely-accepted model of privacy based on introducing carefully calibrated random noise to information revealed about private data. In the standard presentation, the DP model assumes the existence of a trusted aggregator: an entity who holds a collection of information about a population of individuals, and applies differentially private mechanisms to information computed from this data collection. This allows accurate statistics and models to be derived, but comes at a cost: we must be satisfied that we can indeed trust this aggregator to handle the collected data responsibly. In practical applications, this data aggregator is likely to be a powerful entity, such as an internet service provider, technology company, or government, who may collect the private information of millions of individuals. Hence the potential for misuse may be of some concern, even if we have no prior reason to suspect the motives of the aggregator.

In response to this, a number of other models of privacy have been considered which aim to reduce or eliminate the trust placed on the central entity. These can include decentralization – dividing the data among multiple aggregators, so no single one sees the entire information – or cryptographic techniques – which restrict the view of the aggregator of the raw data. In this chapter, we survey an approach known as Local Differential Privacy, or LDP. The LDP approach directly provides a differential privacy guarantee on the results of the computation, and entirely eliminates the need for a trusted aggregator to hold the private data. However, the LDP approach comes at some cost: more computational work is needed, and the results achieve a weaker tradeoff between accuracy and privacy than in the traditional, "centralized" differential privacy model.

The essence of LDP is that each user who holds some private data is now an active participant in the data release process. Instead of passively sending their data to the aggregator and retiring, each user now runs a randomized procedure on their input. The requirement is that the distribution of the message(s) sent by each user individually should satisfy the DP guarantee. Formally, let x_i and x_i' be two possible inputs that user i might hold, let $\mathcal{R}$ denote the (randomized) protocol that user i will apply to generate their messages, and let ε be the desired privacy parameter. We require that for all possible transcripts T of user i's communication, we should have

$$\frac{\Pr[\mathcal{R}(x_i) = T]}{\Pr[\mathcal{R}(x_i') = T]} \leq \exp(\varepsilon) \tag{1}$$

Note that this definition is symmetrical in the roles of x_i and x_i'. This is exactly the standard (pure) differential privacy definition, applied to the inputs of a single user i.

On first glance, attempting to obtain useful results under this restrictive definition may seem doomed to failure. The noise introduced in differentially private mechanisms is calibrated so as to effectively "mask out" the contribution of any single user. Specifically, we usually expect the magnitude of any (numeric) noise added to a function of users' data to be approximately equivalent (in expectation) to the weight of any individual user's data. So we would expect that applying this to the contribution of just a single user would entirely hide the information they are contributing, and prevent any subsequent use of it. Indeed, the first part of this intuition is true: the noisy response of a user should indeed mask out their contribution, and so make it impractical to learn anything about their individual information. However, if we aggregate the reports over a large enough population of users, we can still draw accurate conclusions about the overall population behavior: the signal emerges from the noise. The explanation for this apparent paradox is that since each user independently chooses how to add noise, the noise tends to cancel out as we combine all the user reports, leaving the true answer masked by a smaller magnitude of total noise.

1.1 A first LDP protocol

We illustrate this phenomenon with a simple example drawing on standard DP mechanisms from the centralized case. Suppose we have a population of $N = 1$ million users, who each hold a secret binary value (0 or 1), which encodes some private attribute – say, their opinion on a contentious issue (pro or anti), or some other feature. We would like to learn what is the population-level value of this statistic, i.e., what percentage of our population hold a '1' value. The standard centralized DP approach is to add noise sampled from a Laplace distribution with parameter $1/\varepsilon$. The variance of this distribution is then $2/\varepsilon^2$, and so the expected magnitude of this noise is around $1/\varepsilon$.

This approach can be adapted to the local setting by having each user add this quantity of noise independently to their input. The resulting distribution would have variance $2N/\varepsilon^2$, and hence absolute magnitude proportional to $\sqrt{N}/\varepsilon$. Concretely, consider our case with $N = 1$ million, and $\varepsilon = 1$. Then the local case allows us to estimate the population *fraction* with additive error around 0.001. This is certainly good enough to easily distinguish large values from small, and comparable to the error due to sampling which people to ask. However, the corresponding error in the centralized case is vastly smaller — closer to 10^{-6} in this case.

This small example serves to show that we can obtain sufficient accuracy in the local model of differential privacy, albeit weaker than in the centralized case. To get good results for more complex analysis tasks requires a large population of users participating (honestly) in the protocol, and carefully tuned protocols to maximize the accuracy obtained.

The naive approach of taking an existing approach from the centralized approach does not always work. Consider for example trying to produce a clustering of some input data points. If each user holds a single point (representing their data), then a central DP algorithm is unlikely to produce a very meaningful output for them. Moreover, it is unclear how to combine the results if each user produces a noisy clustering of their single point. Instead, for this and other problems, we would seek novel methods better suited to the local model.

The key properties that we aim to understand for different problems are the tradeoff between accuracy and privacy, as a function of the privacy parameter ε, and the number of participating users, N. Secondarily, we are also concerned about the other computational costs – the time and space required by users to run their part of the protocol, and the time and space needed by the aggregator to interpret all their messages. Lastly, we may also seek to minimize the size of the messages sent by the users, and the number of rounds of interaction between users and the aggregator.

1.2 A brief history of LDP

As will be discussed in subsequent sections, the antecedents of Local Differential Privacy date back at least fifty years, to the work on Randomized Response in survey design (Warner, 1965). More recently, a notion equivalent to local different privacy was introduced by Evfimievski, Gehrke and Srikant (Evfimievski et al., 2003), around the same time that the foundations of differential privacy were being laid (Dinur and Nissim, 2003). The connection between randomized response and differential privacy has been observed in texts on the differential privacy mode (Dwork and Roth, 2014). However, the current interest in LDP can be traced to recent developments in theory and practice. Duchi, Jordan and Wainwright coined the label of "local differential privacy", and studied problems of statistical inference under this model (Duchi et al., 2013). This led to much subsequent interest in the theoretical underpinnings of LDP and its variants. Around the same time, Google published a paper on its RAPPOR system, which is based on LDP (Erlingsson et al., 2014), used to collect browsing statistics privately. Papers on deployments by Apple (Apple, 2017), Microsoft (Ding et al., 2017) and Snap (Pihur et al., 2018) on related problems demonstrated a strong interest in LDP as a practical model for private data collection.

1.3 Overview of the chapter

Following the interest in Local Differential Privacy from both theory and practice, in this chapter we survey a sampling of the developments in LDP. Since its formal introduction less than a decade ago (at time of writing), there have been many hundreds of papers published on the topic, and this chapter provides a very partial view of this topic. We begin by introducing the foundational notion of Randomized Response in Section 2, for revealing information about a binary choice. In Section 3, we show how randomized response has been extended and enhanced to provide the notion of a "Frequency Oracle", which gathers information about the distribution of values held by a collection of users. The Frequency Oracle is the basis for many of the applications of LDP. Some of the basic statistical collection tasks, such as finding frequent items, and capturing the cumulative distribution, are described in Section 4. We move on to more advanced data analysis and modelling tasks in Section 5. Finally, we conclude by reflecting on the limitations on LDP protocols, and describe some alternate related models which attempt to remedy these deficiencies in Section 6.

2 Randomized Response

In this section, we return to the problem stated in the introduction of this chapter, to estimate the proportion of individuals whose binary input is a 1 value. We will describe a simple approach that provides an accurate answer, and show its analysis in some detail. This approach, known as Randomized Response, is at the heart of many LDP algorithms that we will discuss subsequently (albeit in less detail).

The approach from Section 1.1, of adding noise drawn from a Laplace distribution, certainly works, but has some disadvantages. It produces potentially large real numerical values. Since the inputs are binary, it is natural to ask whether we can restrict the messages from each user to the same domain.

If we do so, then we can immediately derive the description of a protocol. For a user with input value $b \in \{0,1\}$, their only option is to report either b truthfully, or lie, and report $1-b$. Suppose we set the probability of truthtelling to be p, and hence the probability of lying is $1-p$. We can assume without loss of generality that $p > \frac{1}{2} > 1-p$. To ensure that the (local) differential privacy property (1) holds, we require that the probability of seeing the same output in the two cases that $b = 0$ and $b = 1$ satisfies

$$\frac{p}{1-p} \leq \exp(\varepsilon).$$

That is, the ratio between (correctly) observing a 1 on input 1, and (erroneously) seeing a 1 on input 0, should not exceed the bound of $\exp(\varepsilon)$. For utility, we would like p to be as large as possible, which means setting $\frac{p}{1-p} = \exp(\varepsilon)$. Rearranging, we obtain that $p = \frac{\exp(\varepsilon)}{1+\exp(\varepsilon)}$.

Suppose we apply this protocol over a large population where a ϕ fraction of users have input 1, and the rest have input 0. Then, for each user, the expected observed value o is given by

$$\mathsf{E}[o] = p\phi + (1-p)(1-\phi)$$

From this, we can form an unbiased estimate for the (unknown) parameter ϕ by rearranging: we write

$$\hat{\phi} = \frac{o - (1-p)}{2p-1}$$

We can observe that the observed value o is equal to the input x with probability p, otherwise they are different. In the former case, the error of $\hat{\phi}$ is $\frac{p-1}{2p-1}$, and in the latter it is $\frac{p}{2p-1}$. Hence, we can also compute the variance of this (unbiased) estimator, as

$$\mathsf{Var}[\hat{\phi}] = \mathsf{E}[(\phi - \hat{\phi})^2] = \frac{p(p-1)^2 + (1-p)p^2}{(2p-1)^2} = \frac{p(1-p)}{(2p-1)^2}$$

Using our setting of $p = \exp(\varepsilon)/(1+\exp(\varepsilon))$, we obtain

$$\mathsf{Var}[\hat{\phi}] = \frac{\exp(\varepsilon)}{1+\exp(\varepsilon)} \cdot \frac{1}{1+\exp(\varepsilon)} \frac{(1+\exp(\varepsilon))^2}{(\exp(\varepsilon)-1)^2} = \frac{\exp(\varepsilon)}{(\exp(\varepsilon)-1)^2}$$

We can approximate this quantity when ε is small, in which case $\exp(\varepsilon) \approx 1 + \varepsilon$ and so $\mathsf{Var}[\hat{\phi}] \approx \frac{1}{\varepsilon^2}$.

Equipped with this understanding, we can observe that the average of N unbiased estimates for a population of N users will have a variance proportional to N/ε^2, and so an expected absolute error proportional to $\sqrt{N}/\varepsilon$. This is comparable in scale to the cruder method based on adding Laplace noise, but comes with a tighter understanding of the behavior (variance less by constant factors), and reduced communication costs (the protocol sends noisy bits instead of noisy real values).

The notion of randomized response was first introduced by Warner (1965), as a method of providing plausible deniability for survey respondents answering potentially embarrassing questions. It was subsequently observed to provide differential privacy, and has been used as the foundation for many other protocols achieving local differential privacy.

3 Frequency Oracles

The Randomized Response protocol in the previous section allows us to estimate the fraction of population satisfying a certain property. Equivalently, it can be viewed as providing an estimate of the parameter of a binomial distribution, when each user samples from the same global Bernoulli distribution. More generally, we might want to estimate the distribution when each user has one out of d possible categorical values. This problem has been widely studied in the context of local differential privacy, and the techniques developed for this problem have been used in the solution of other problems discussed subsequently.

Across a variety of different approaches, some similarities emerge: each user sends a message noisily encoding information about their input, which can be aggregated to provide an estimate of the number of occurrences of a particular category. That is, if each user i has an input value $x_i \in [d]$, we seek to build an estimate for $f(x) = |\{i : x_i = x\}|$. The estimate of $f(x)$ will be a random variable, whose randomness derives from the random choices of the users in adding noise to their input. We refer to the randomized algorithm that can produce an estimate $\hat{f}(x)$ of $f(x)$ for any $x \in [d]$ as a *frequency oracle*. The aim is to minimize the variance of the estimate, as well as keeping the other computational costs low. In this section, we describe the operation of several different frequency oracles, and summarize their properties.

3.1 Direct Encoding

A natural first attempt to build a frequency oracle is to generalize the notion of randomized response to d possible values. As with (binary) randomized response, each user can report their input truthfully with probability p, or pick each of the other possible inputs with probability $(1-p)/(d-1)$ each. Satisfying the LDP property (1) leads to choosing $p = \exp(\varepsilon)/(\exp(\varepsilon)+d-1)$. Here, the expression for the variance of this estimator is a little more complex, so we focus on the variance conditioned on reporting an erroneous value, which tends to dominate the cost. We write this variance bound as Var^*, which is given by $\mathsf{Var}^* = \frac{\exp(\varepsilon)+d-2}{(\exp(\varepsilon)-1)^2}$. We can observe that this agrees exactly with the variance for binary randomized response when setting $d = 2$. This analysis of Direct Encoding, and the definition of Var^*, is due to Wang et al. (2017).

3.2 Unary encoding

An alternative generalization of randomized response is to use a unary encoding of the input, and apply randomized response independently to each bit. That is, we express user i's input as a d-bit "one-hot" encoding, where we set the x_i'th bit to be a 1, and the remaining bits to 0. We then flip each bit independently, and report the noisy result. We can observe that in order to ensure the differential privacy condition (1), we only need to consider that pairs of inputs can change in at most two location: between two possible inputs, there is one location where a 1 is replaced by a 0, and one where a 0 is replaced by a 1.

To improve the accuracy, we can observe that we can choose different probabilities for flipping a 0 to a 1 than for flipping a 1 to a 0. We obtain better accuracy if the former is lower and the latter is higher, since there are more 0s to preserve than 1s in this sparse encoding. Optimizing this choice subject to the DP condition leads to picking the probability of flipping a 1 to 0 as $\frac{1}{2}$, while 0s are flipped to 1s with lower probability of $\frac{1}{1+\exp(\varepsilon)}$. This choices minimizes $\mathsf{Var}^* = \frac{4\exp(\varepsilon)}{(\exp(\varepsilon)-1)^2}$. Observe that this improves over direct encoding for larger d, when $d > 3\exp(\varepsilon)+2$, which is typically the case for moderate values of ε and d. However, this comes at the cost of sending d bit messages, which can become excessive for large values of d. The notion of "optimized" unary encoding is due to Wang et al. (2017).

3.3 Hash encoding

When the domain size d gets moderately large, it can be useful to use hashing techniques to reduce the effective domain size, while still providing enough information to encode the frequency distribution. The idea of hash encoding is for each user to (independently) pick a random hash function, and to use a method such as direct encoding to privately encode the noisy value of the hash of the user's input. Given the noisy encoding and the description of each hash function, an aggregator can build up an accurate picture of the overall population distribution. Analyzing this procedure shows it suffices to use relatively simple hash functions (specifically, hash functions that meet the condition of "pairwise independence"), onto a small domain of possibilities. Optimizing the parameters leads to choosing the range of hash values to be $\exp(\varepsilon)+1$, so that the true hash value is reported with probability $\frac{1}{2}$, while the other choices are each selected with probability $\frac{1}{1+\exp(\varepsilon)}$ — mirroring the probabilities from unary encoding. Following similar calculations, we find $\mathsf{Var}^* = \frac{4\exp(\varepsilon)}{(\exp(\varepsilon)-1)^2}$. Now the messages sent are just $\log_2(\exp(\varepsilon)+1)$ bits to encode the hash value, plus $O(\log d)$ to specify the hash function. However, the work required by the aggregator to build estimators from the N messages is quite large – $O(Nd)$. This optimized local hashing approach is also due to Wang et al. (2017).

3.4 Hadamard encoding

The approach of Hadamard encoding is quite similar to the hashing approach, where the hash function is drawn based on a structured set of possibilities. This allows the decoding of messages to be performed more quickly, while achieving similar accuracy. Specifically, we consider a hash function which maps onto two possible values, -1 and $+1$. The j'th hash function is specified by $h_j(x) = (-1)^{\langle j,x\rangle}$, where $\langle j,x\rangle$ denotes the inner product of the binary representations of j and x. The hash value can be encoded by standard binary randomized response over the two possibilities of -1 and $+1$.

This special set of hash functions can be interpreted as encoding the *Hadamard transform* of the input. The Hadamard transform is an instance of a Fourier transform, and possesses a fast algorithm to transform and invert a vector of values. Consequently, the aggregator can use the Fast Hadamard (inverse) transform to accumulate and compute the frequency estimates in time proportional to $O(N + d\log d)$, assuming d is a power of 2. The message size is 1 bit to encode the noisy value, plus $\log d$ bits to send the value of j that defines the hash function. The resulting variance bound is $\frac{(\exp(\varepsilon)+1)^2}{(\exp(\varepsilon)-1)^2}$. This is close to optimal when $\exp(\varepsilon)$ is close to 1. For larger values of $\exp(\varepsilon)$, variations can be applied, such as sending multiple hash values, to reduce the variance to $O\left(\frac{\exp(\varepsilon)}{(\exp(\varepsilon)-1)^2}\right)$. The idea of using Hadamard encoding to build a frequency oracle appears in a variety of places (Apple, 2017; Nguyên et al., 2016; Acharya et al., 2019).

3.5 Domain Size Reduction

The above approaches work well as d ranges up to moderate size – say, from single digits up to thousands of possibilities. However, when d is huge, all the methods lessen in usefulness, due to increased cost to maintain, and reduced accuracy. Such large domains arise in practice, for example when capturing information about words typed by users, or websites visited, where the set of possibilities easily ranges into the millions.

To cope with this high domain size, protocols can take advantage of dimensionality reduction techniques. Essentially, the protocol specifies a randomly chosen projection of the input items into a lower dimensional space, so that frequencies in the lower dimensional space can be used to provide frequency estimates for items in the original space. Such dimensionality reduction techniques are often referred to as "sketches". Common examples used in the LDP setting include the Bloom Filter, Count sketch and Count-Min sketch (Cormode and Yi, 2020). Importantly, these techniques are all *sparsity preserving*: if we apply them to a single input item, represented as a one-hot vector, then the resulting sketch is also mostly zero, having only a few non-zero values, and adhering to some additional structural restrictions. This means that we can apply the above frequency oracles to a sketch representation of user inputs in an almost black-box fashion. The cost now depends on the (smaller) d' value representing the size of the sketch, instead of the much larger original d value. Papers that pursued this approach provide further details of how to guarantee privacy and accuracy with the Bloom Filter (Erlingsson et al., 2014), Count-Min sketch (Apple, 2017) and Count sketch (Bassily et al., 2020).

4 Heavy Hitters, Marginals and Range Queries

Once we have a frequency oracle, we can apply it to numerous other related statistics-gathering problems. In this section, we describe how this can be done in the case of finding frequent items from a large domain (heavy hitters), computing marginal distributions within multidimensional data, and answering range queries and quantile queries.

4.1 Heavy Hitters

The "heavy hitters" within a distribution are those items whose frequency is large. This problem is naturally closely tied to the use of frequency oracles. The difference is that a frequency oracle allows us to test the frequency of a single item, whereas finding the heavy hitters potentially requires ranging over a very large set of possibilities. In the LDP setting, a number of approaches have been suggested to make this search procedure efficient and accurate.

The naive approach is simply to perform a frequency estimation query for every item in the domain. The disadvantages of this are primarily the computational cost: heavy hitters are often sought over a very large domain of possibilities. For example, the domain may consist of words typed by users on mobile devices, or URLs visited. Considering the number of valid strings of characters to consider, this can easily reach billions or trillions of possibilities. Enumerating all such possibilities can be very time consuming indeed. Moreover, even though each query to a frequency oracle may be quite accurate, over this many probes, there will be some errors, where infrequent items are reported as having a high frequency. Thus, we would expect some amount of false positives.

To reduce the number of queries, and hence false positives, we can seek ways to prune the search space, using ideas from information theory and error-correcting codes. For instance, suppose we knew that there were no heavy hitter items beginning with the letter 'z': then we could save ourselves the effort of testing any sequence of characters starting 'z', and so cut off this part of the search space. For our running example, we will consider inputs that are six letter words over the Roman alphabet, e.g. "apples" or "banana", where each user holds one such word, and we want to find which words are most common.

Some of the first approaches proposed to find heavy hitters were based on breaking the input strings into shorter substrings, and finding which substrings are frequent. In our running example, we could consider all two letter substrings – say, "ba" (the first two characters of banana), or "pl" (the middle two characters of apples). Each user can split their input word into its (disjoint) substrings, and report each of these through a frequency oracle. So, apples is split into ap, pl and es. Once the frequency oracle has been built from all the user submissions, a data analyst can try to find the heavy hitters. They can query the frequency oracle for all heavy substrings: note that posing $26^2 = 676$ queries for all character pairs is much smaller than the 308 million strings of length 6. This then gives a "jigsaw puzzle" to solve: how to recombine the heavy substrings into the correct set of heavy words?

A first approach proposed is to use statistical inference over pairs of substrings (Fanti et al., 2016), although this can potentially get fooled into outputting infrequent combinations. Subsequent approaches seek instead to perform a more reliable search over the space of possibilities. The "sequence fragment puzzle" (SFP) approach tags each substring with more information (Apple, 2017). It first concatenates each substring with a hash value of the whole string, to avoid substrings of different strings getting mixed up with each other. It also tags each substring with the location of the substring within the whole string, which is not private information. So in our example, if the hash value of `apples` was 7, then `pl` would be reported via a frequency oracle as (`pl7`, 3), i.e., the string `pl` occurs at index 3, and is concatenated with the hash value 7. If the string `apples` were a heavy hitter, then the analyst would recover the tuples (`ap7`, 1), (`pl7`, 3), (`es7`, 5), and so have enough information to recover the full string. Another frequency oracle can be kept over the full strings to provide an additional check that the reconstructed strings are indeed heavy.

Instead of collecting all the pieces in one go and putting them back together, hierarchical approaches to solving the heavy hitters problem incrementally build up the strings of heavy items. In this case, each user reports prefixes of their input (via a frequency oracle). In our example, the prefixes of `apples` in multiples of two characters are `ap`, `appl` and `apples`. The data analyst can first test all two character prefixes, and collect only those that appear heavy – e.g., `aa` to `zz`. They can then test only those four character prefixes that extend a heavy prefix – so in our example, this would test from `apaa` to `apzz`, to identify `appl` as heavy. The search proceeds until the full set of heavy hitters has been found. Although this requires a little more work than the SFP approach, since more candidates are considered, the accuracy is increased, as no hash functions are needed. This idea and its variations has appeared under various names – TreeHistogram (Bassily et al., 2020), Prefix Extending Method (Wang et al., 2019), and PrivTrie (Wang et al., 2018b). Across these variants, some recommendations have emerged. For example, it is preferable for each user to only report a single substring, rather than all substrings of their input, as this gives a better tradeoff of accuracy against privacy guarantees. It is also natural to keep information about each prefix length in a separate frequency oracle, to reduce noise.

Last, other approaches have been suggested in the theoretical literature which achieve slightly better results at the expense of rather more complex constructions making use of error correcting codes (Bun et al., 2018; Bassily and Smith, 2015; Bassily et al., 2020). However, these do not seem to have seen use in practice. In practical deployments, the RAPPOR system has used the statistical expectation-maximization approach (Fanti et al., 2016), while Apple's implementation uses their Sequence Fragment Puzzle algorithm (Apple, 2017).

4.2 Marginals

Given inputs represented as d-dimensional feature vectors, it is often desirable to learn statistics about combinations of k features. For example, given inputs recording sex, height and weight of individuals, the three *2-way marginals* capture information about sex and height; sex and weight; and height and weight, respectively. Naively, we could directly apply frequency oracles to solve this, by having each user report information about their values for each k-way marginal. However, as k and d grow, this quickly becomes unsuitable. For $d = 10$ and $k = 3$, there are $\binom{10}{3} = 120$ distinct 3-way marginals. Maintaining this many frequency oracles (one for each marginal) will quickly lose accuracy. At the other extreme, we could simply maintain a single frequency oracle to capture the full d-way distribution, then project the desired k-way marginal by "marginalizing" the unwanted dimensions. This also incurs a lot of inaccuracy, as noise is added up.

Various approaches have been suggested to handle this approach. A first approach, drawing on insights from the theory of representing functions, and techniques used in the centralized privacy setting, is to make further use of the Hadamard transform of the input, as described by Cormode et al. (2018b). It is observed that any marginal distribution over binary data can be expressed as a linear combination of only $\binom{d}{k}$ Hadamard coefficients, which can be much smaller than the 2^d coefficients needed to represent the full binary data. This allows each Hadamard coefficient to be found more accurately, since we can ignore those Hadamard coefficients not needed for any k-way marginal.

The LoPub approach proposed by Ren et al. (2018) captures information about lower degree marginals (say, $k = 2$), then uses various statistical approaches as postprocessing to build up a picture of the higher order marginals. This involves expectation maximization to infer the marginals, and lasso regression to enforce sparsity in the resulting distributions.

An alternative approach is to select a subset of marginal distributions to materialize, so that any desired marginal can be found by marginalizing a larger distribution. The "Consistent Adaptive Local Marginal" (CALM) approach presented by Zhang et al. (2018) defines how to pick such a set of marginals to materialize of a fixed size, chosen to minimize the (squared) error incurred. Additional postprocessing is applied to improve accuracy by removing inconsistencies among the overlapping marginals.

4.3 Range Queries and Quantiles

Often, private data is drawn from an ordered domain — say, salary values, or exam scores. A frequency oracle allows us to learn the distribution of individual values, but more often we would also like to learn the cumulative distribution of values. This captures notions such as the median value, or the fraction of user inputs that fall within a specified range. Broadly, we describe these problems as range queries (what fraction of inputs fall between x and y?), and quantiles (for which input value x does a q fraction of inputs fall below?). Both can be answered by reducing to prefix queries, which ask what fraction of user inputs fall below a given value x.

There are two popular approaches to answering prefix queries that have been described in the LDP setting. Hierarchical approaches impose a regular hierarchy over the (discrete) input domain — such as a binary tree of height h. Then any prefix can be answered by summing up the weights associated with at most h nodes of the (binary) tree. These weights can be found by having each user report information about the items within the imposed binary tree, via a frequency oracle. Tradeoffs can be achieved by varying the depth of the hierarchy, and its branching factor. Transformation based approaches apply an appropriate (linear) transformation to the input — the most relevant being the Haar wavelet transform. This transforms a user's input into a (sparse) set of Haar coefficient values, which can also be collected and aggregated via frequency oracles. The data analyst can then answer prefix queries based on the noisy (but unbiased) wavelet coefficients. Similar approaches have been evaluated in the centralized setting. Studies under the LDP model found that the accuracy of both approaches is very similar in practice (Cormode et al., 2019), and that the hierarchical approach can naturally extend to multi-dimensional range queries with good accuracy (Wang et al., 2019).

5 Local Differential Privacy in Applications

Building on the above techniques for gathering statistics on arbitrary data distributions, there has been much work on solving various modeling and machine learning tasks under the LDP guarantee. In this section, we survey some of the approaches used, across a range of different settings, such as location and network-based data.

5.1 Text and Language Modeling

There are many reasons to want to analyze text. For example, we may wish to understand the evolving use of language. More pragmatically, operating system developers may wish to build better models to help mobile users type more accurately, by automatically correcting spelling errors. Equally, there are strong reasons to consider the text typed by users on their devices to be highly private, so it is not suitable to ship it in bulk to an analyst. Current deployments of LDP have often focused on text and text-like data. For example, Apple's implementation focuses on building custom dictionaries based on currently popular words, and highlighting popular emoji in lists. Meanwhile, Google's collection has focused on popular URLs, which can be represented as long text strings from a large domain.

An instructive example of how this modeling can be combined with LDP is due to Chang and Thakurta (2018), in their work on autocompletion under local differential privacy. Absent of privacy, language modelling relies on building large, high dimensional models. For example, we might seek to predict the next word based on looking at the most recent k words typed, for some moderate choice of k (say, 4-5). However, when we apply the constraints of privacy, we encounter a challenge: the noise for privacy will typically dominate the low-frequencies associated with such rare combinations.

Instead, Chang and Thakurta propose a simpler "tag and words" model. Each word in some text is tagged with its part of speech (noun, verb, adverb etc.), and the tag for the next word is predicted based on the tags of the two most recent words. Then, options for the next word are predicted based on the predicted tag and the last word. This model can be instantiated in LDP by building a frequency oracle of the distributions of tag triples, and word-tag-word triples. Importantly, the size of these distributions is kept relatively low, so the statistics required to instantiate them have higher support, and the privacy noise is reduced. The lesson for LDP protocol design is to prefer simpler models whose parameters can be learned with higher confidence.

5.2 Spatial Data

Information associated with people's location can naturally be very sensitive. Protecting people's location is widely agreed to be a strict requirement of private data handling. A canonical (hypothetical) example often referred to is a person attending a sexual health clinic: any data release which reveals their attendance violates their privacy, and potentially threatens their status and well-being. Nevertheless, revealing suitably private views of location and movement data of populations is considered a valuable aim. It can inform planning for amenities and businesses, as well as influencing political districting and government fund allocations.

A first attempt to handle spatial data is to impose a simple division of space, such as a grid, and to reveal (noisy) counts of the occupancy of each cell. Such approaches have been proposed in the centralized model of privacy, with moderate success. In this case, the aim is to find heavy regions, and can be addressed using frequency oracles. In the local setting, the drawbacks of this approach are that it can be hard to determine the appropriate granularity of the grid. Sparsely populated rural areas may suit coarse cells (of the order of kilometres in size), but densely populated urban areas require much finer representations to best describe them, perhaps only tens of metres in size.

To make progress on this problem, it may be appropriate to relax the privacy constraints. In particular, while individuals may not wish their exact location to be revealed, it may be acceptable to reveal the country that an individual is within. That is, we might be more likely to perturb someone's location within a smaller radius, but much less likely to impose a very large perturbation.

The approach of private spatial data aggregation of Chen et al. (2016) defines such a relaxation of local differential privacy, based on a hierarchy over locations. This can be based on public geographic features, such as countries divided into states, which are further subdivided into cities and quarters. Each user can then decide at what level they are comfortable to be placed in – for example, they may allow the true city to be revealed, but not exactly where in the city they are located. Location distributions can then be revealed via an appropriately generalized frequency oracle. The user's location may be perturbed, but locations outside their specified "safe zone" are constrained to be empty. The system as described can be further extended: users can also choose a "personalized" privacy parameter ε_i; the system can try to group users together to run fewer invocations of a frequency oracle.

5.3 Graphs and Social Network Data

Data emerging from social interactions can naturally be subject to some privacy concerns. While some online social networks expose a certain amount of information to the world, such as lists of "friends" or "followers", other information is more restricted. Most social network systems do not reveal which users have been in close communication, or information about the frequency and intensity of these interactions. Nevertheless, to sociologists and other students of human behavior, learning accurate information about people's actions within (online) social networks is of great interest.

Most work on private graph data analysis takes the approach of (explicitly or implicitly) revealing information in the form of a mathematical model. That is, we consider the input to describe a member of a family of graphs, and we aim to extract certain information about the graph in order to instantiate a statistical graph model. The parameters of the model can be revealed under a suitable model of privacy. Then analysis can be performed, either directly on these noisy parameters, or by sampling new graphs according the graph model, and studying their properties. This approach has proved challenging to guarantee useful results even in the centralized model of differential privacy, so only becomes more difficult in the local regime, where higher volumes of noise are required.

A first question for this setting is how to define a suitable notion of privacy to graph data. In particular, what is the appropriate notion of "neighboring inputs" to apply the DP definition to? In the node privacy model, two graphs are considered to be neighboring if they differ in the adjacency pattern of a single node — that is, the connections from one node in the graph can be completely rewritten. In the edge privacy model, neighboring graphs differ only in the presence or absence of a single edge. The edge privacy definition holds the edge relationship to be the unit of privacy, while the node privacy model focuses on the node as the basic unit. Clearly, edge privacy requires less perturbation of information to achieve, since one node may correspond to a large number of edge links. Arguably, node privacy is the definition to aspire to, since in many social network graphs, the node represents all the information of a single user. This puts it closest to definitions of differential privacy, which seek to bound information based on the addition or removal of an individual (person) from the dataset. Nevertheless, most work has addressed the more tractable edge (LDP) case.

Given a social network graph, where each user has a list of other nodes that they are linked to in the graph, there are two natural approaches to consider first as baselines. First, one might try to reveal the full adjacency list of a user — perhaps, treated as a (relatively sparse) binary vector indicating which links are present. This can be performed under LDP by applying randomized response to each bit in the adjacency vector. However, it will of necessity introduce a very large number of false neighbors for any node. Instead, we may seek a model to instantiate where the parameters have more support, and hence give greater confidence. Second, we could provide very simple information about each node, such as its degree. Under the model of edge privacy, this can be done quite effectively: the average noise will be a small constant, compared to degrees in social networks which are typically hundreds on average. This can be used to instantiate graph models which require only degree information. However, this can be unsatisfying, since node degrees do not capture any useful information about how pairs of nodes interact with others, say. Consequently, we find that the adjacency list approach is too detailed and subject to noise, while the degree approach is too coarse and uninformative. Instead, we seek a graph model which falls in between these two.

Qin et al. (2017) propose such an approach, which is learned iteratively under LDP. It aims to build a description of the graph in terms of the pattern of connections between each node and a set of possible partners. That is, we would like to define a small set of "clusters" of nodes, and describe how many links each individual node has to each of the clusters. The two previous approaches can be viewed as extremes within this setting: trying to reveal the adjacency list can be seen as the case when each cluster is a single other node; revealing degrees corresponds to having a single cluster containing all the nodes. Ideally, we would be given a suitable clustering of nodes (e.g., each cluster could represent the continent on which people live). However, in general no such clustering is available, and so it must be learned from scratch.

Qin et al. describe a way to build a clustering over a set of iterations. In the first iteration, each node is randomly assigned to one of k clusters. Each user can then release a noisy histogram of how their neighbors fall into these clustering. Each histogram is of size k, and is noised by adding values sampled from a Laplace distribution to each entry, sufficient to provide edge privacy. The next stage is to derive a new set of clusters. This is done by applying a clustering procedure to the (noisy) histograms, to find a new set of k clusters. Each node is then assigned to the new cluster in which its histogram is placed. The new cluster assignments are (implicitly) shared with neighbors, and the procedure can be iterated for a fixed number of steps. The intent is that the final set of clusters provide a representative set of different patterns of connection which help to describe the nodes. Finally, once the final (noisy) representation is built, it can be released publicly, and used as the basis of a sampling procedure to sample synthetic graphs with similar connection patterns. Experiments on this approach show that the sampled graphs do well at preserving various graph properties (clustering coefficient, mutual information, etc.) of the original input, while providing an LDP guarantee.

5.4 Classification and Regression

Building accurate predictive models via classification (for predicting categoric values) and regression (for numeric values) is at the heart of the recent explosion of interest in machine learning. In the private data setting, we can think of each user holding one (or a few) different training examples, with the goal being to build an accurate model from these distributed examples. The LDP requirement ensures that we should not be able to learn the private information of any user (although, of course, the learned model may nevertheless allow us to make inferences about individuals and their data).

Minimizing a loss function. Although there are many and varied models and learning procedures described in the literature, there are some similarities that allow a common approach to be proposed. Specifically, the training of many ML models can be expressed as minimizing a loss function over user examples, with features x (as a vector) and label y. The exact form of the loss function depends on the model type being learned, and the regularization being applied, but in many cases the loss is expressed as a sum over a loss from each example under the current model. That is, we can write the objective as trying to minimize the function

$$\sum_i \mathrm{Loss}(\Theta, x_i, y_i) + \lambda N \|\Theta\|_2^2$$

where, i indexes the N examples (x_i, y_i); Θ are the model parameters to vary; λ is a regularization parameter; and Loss is a function that gives the loss for each example under the current parameters. Two cases that can be expressed under this framework are linear regression, where the loss function is $\text{Loss}(\Theta, x_i, y_i) = (x_i^T\Theta - y_i)^2$; and support vector machines with hinge loss, where $\text{Loss}(\Theta, x_i, y_i) = \max(0, 1 - y_i x_i^T\Theta)$.

Although certain models may admit a specific or closed-form solution, a more general approach is to optimize the loss function via stochastic gradient descent. That is, starting from some randomly chosen initial Θ, we choose a new Θ' by evaluating the gradient of the loss function on an appropriate subset of data points, and taking a step in the direction to reduce the loss function. The key aspect that makes this suitable for the LDP model is that, since the total loss is a sum over the loss of each data point, so also is the gradient of the loss function. Hence, we can start to build a protocol in the LDP model: a user i will receive a current value of Θ (which is non-private), and will reveal information about the gradient of the loss function with Θ and x_i, y_i.

Vector Release. To complete the description of the protocol, we need to determine how to output this gradient privately. We refer to the problem of publishing a vector with appropriate (local) differential privacy guarantees as "Vector Release". A first challenge for vector release is that gradient values could be arbitrarily large. This would make it hard to guarantee privacy, since we need to ensure that large values are protected by correspondingly large noise. A standard solution is "clipping": we ensure that the magnitude of each entry in the gradient vector is clipped to the range $[-1, +1]$. It is now possible to adapt existing LDP mechanisms to solve the vector release problem. For an individual coordinate in the range $[-1, +1]$, one can apply a variant of randomized response. A simple approach due to Duchi et al. (2013) is to first randomly round the input to $\{-1, +1\}$: given $r \in [-1, +1]$, map to $+1$ with probability $(r+1)/2$, otherwise map to -1. We can then apply randomized response to the two inputs $\{-1, +1\}$ based on the parameter ε.

In order to handle d dimensional vector inputs, there are alternate approaches to consider. We could apply this rounding to every coordinate in the vector, after reducing the privacy parameter to ε/d, invoking a privacy composition result for LDP. Or, we could pick one (or a few) coordinates to release with a larger privacy budget. This latter approach is advocated in the work of Nguyên et al. (2016), who show that it gives preferable results. Taking the mean of the unbiased responses from the users is used to give a gradient update, and compute a new set of parameters Θ'.

Putting it all together. Finally, we need to consider multiple steps of the gradient descent algorithm. Again, we could have each user participate in multiple rounds, by dividing their privacy budget up into smaller pieces; or divide the users into groups ("minibatches"), and have each user participate in one round only. Analysis due to Nguyên et al. (2016), who proposed and analyzed this protocol, demonstrates that the minibatch approach is clearly preferable. This means that obtaining accurate results can be a challenge, since the population of users is spread out over a number of rounds, and a large number of users is needed to ensure accuracy. This motivates working with a small model dimension d when possible, perhaps by using random projection techniques to reduce the dimensionality.

The need for multiple rounds of interaction means that convergence may be slow, due to the need to wait to receive responses from a large population of distributed users, perhaps using relatively impoverished mobile devices. There has been some theoretical study of whether multiple rounds of interaction are necessary for LDP machine learning tasks. Smith et al. (2017) show constant-round protocols for simple problems like least-squares regression; Zheng et al. (2017) incorporate dimensionality reduction and approximation theory to learn in a single round; and Wang et al. (2018a) use polynomial approximations for smooth loss functions to further improve on these.

5.5 Recommender Systems

Recommender systems form another compelling use-case for Local Differential Privacy. They abstract the problem of providing meaningful recommendations to users, based on their previously expressed interest in items. Recommender systems are already widely used in the non-private setting, to recommend products from an e-commerce site based on past purchases, or to recommend music or movies.

Formally, we imagine that user preferences can be represented by an $n \times m$ matrix R, where n is the number of users and m is the number of items. The value of $R_{i,j}$ is the rating given by user i to item j. The rating system can be a scale (0 to 5), or binary (indicating whether or not the user has bought that item). Typically the matrix R is large and sparse. A common approach to recommender systems is to try to approximately factor R into UV, where U is an $n \times \ell$ matrix and V is $\ell \times m$. The parameter ℓ captures the number of "latent factors". Then the (predicted) ranking for item j for user i is given by $(UV)_{i,j}$.

If we introduce $y_{i,j}$ to indicate whether user i has entered a rating for item j, we can write an objective function for this in order to minimize a loss, $\sum_{i,j} y_{i,j}(r_{i,j} - U_i^T V_j)^2$, which is the squared error in predictions compared to the known ratings. As in the previous section, this can be solved by gradient descent, alternating over updating U and V.

For Local Differential Privacy, Shin et al. (2018) proposed an approach that offers a guarantee per user, across their entire data. It makes use of a protocol for "vector release", as described in the previous section, over multiple iterations. Informally, each user will maintain their presence in U as a vector u_i, which is not shared with anyone else, and their ratings, as a vector r_i. Given a current choice of matrix V, each user will compute a gradient vector based on minimizing their contribution to the loss function, i.e., based on $\text{Loss}(V, u_i, r_i) = \sum_j y_{i,j}((r_i)_j - u_i^T V_j)^2$. As before, clipping can be used to ensure that the gradients stay within $[-1, 1]^m$. These (noisy) gradients are returned to the data analyst, who averages them to obtain a new matrix V', and the procedure is repeated, up to some fixed number k of iterations. The final matrix V_k is then shared with users for them to make future predictions with. The result is effective in practice, although is somewhat static, since it is not convenient to add new users i or items j after the initial training is complete.

5.6 Common Themes for LDP in Machine Learning Applications

Across these various examples of combining modeling and machine learning with local differential privacy, some common themes emerge due to the interaction of privacy requirements with ML models.

Finding a good class of models. In non-private machine learning and data analytics, there is a trend towards bigger and more complex models: larger volumes of training data mean that we can fit richer models, and capture behaviors exhibited by only a small fraction of the training examples. This is not uniformly the case under models of privacy, and particularly so under local differential privacy. Firstly, fitting parts of the model to a small number of examples contradicts the aims of privacy, where we seek to protect the information of individuals or small groups. Moreover, the model of LDP works against this: the noise added must be such that if a parameter is only supported by a small number of individuals, then the recovered value will be completely distorted by the noise. To obtain accurate values, we should ensure that they are supported by a large enough fraction of the users. This is most clearly seen in the text prediction model (Section 5.1): the model is chosen to be compact, and is factored into two small pieces, rather than explicitly trying to learn a larger joint distribution. However, the same phenomenon guides other tasks: the work on graphs described in Section 5.3 seeks a model where each user contributes to a low-degree model (captured by the parameter k) that is much smaller than the number of nodes, n. The works on other ML tasks in Sections 5.4 and 5.5 are similarly affected by the dimensionality d, and it is suggested to choose d relatively small, or to use techniques like random projections to reduce the dimensionality.

Collect data that can be combined linearly. A common thread across all the examples discussed is that the data collected from users is combined in a way that is fundamentally *linear*. That is, we can sum up the debiased user reports. This is clearest for the earlier examples of gathering statistics on frequencies and distributions (Section 4), where the data — whether in the form of histograms or Hadamard/Wavelet coefficients — are simple linear transforms of the users' inputs that can be added together by the data analyst. However, our other examples, of text, spatial data, graphs, classification and recommender systems, can all be viewed as combining data linearly, in the form of vectors, matrices, histograms or distributions.

Note that this does not preclude the use of non-linear models, just that the data collected is most conveniently handled in a linear form. Consider the highly non-linear models that emerge when using (deep) neural networks. We can consider learning the weights for such models under LDP. Many optimization techniques are non-linear, and so are hard to implement under LDP. But various gradient descent methods are feasible here. The computation of the gradient itself is a non-linear function of the input. Nevertheless, as described in Section 5.4, it can be decomposed into the gradient due to each user's input. We can therefore delegate the (non-linear) local gradient computation to the user, who can then use an appropriate LDP mechanism to release information about the gradient vector to the analyst. By ensuring that non-linear tasks are performed either by users or by the analyst, in such a way that the messages sent can be combined linearly, we obtain solutions that can be practical under LDP.

Reduce to well-understood problems. Although we have considered quite a varied range of applications, there are only a few different abstract problems solved by the central parts of the LDP protocol. In many of the cases seen above, the problem reduces either to that of Frequency Oracle, or Vector Release. These two primitives, and some of their generalizations, form the backbone of many LDP protocols. This is a helpful abstraction, since it means that LDP protocol designers do not have to re-solve these basic problems, but can adopt one of the optimized solutions from the literature. It also helps to focus on a feasible design for a new application: we should first attempt to reduce the problem to a model that can be instantiated by one or other of these methods, before turning to think about other possible solution methods.

Noise reduction techniques. In the centralized differential privacy case, the existence of composition theorems make it commonplace to build compound mechanisms using various steps. It is common to treat the overall privacy parameter ε as a "privacy budget", and to divide this budget among subtasks. This also allows protocols to be spread across multiple iterations, where the budget is further subdivided across each iteration. While corresponding composition results hold for LDP, the mathematics of the analysis tend to encourage different approaches to building protocols. Specifically, rather than asking each user to answer multiple questions about their input, it is usually better to partition the users into groups, and ask each group one of the questions. Or, somewhat equivalently, we may ask each user to sample one question to answer out of the set of possibilities. Roughly speaking, this is because LDP protocols built on top of frequency oracles typically have an error behavior that scales with $\frac{1}{\varepsilon\sqrt{n}}$. If we are trying to measure d quantities, then dividing the error budget ε into d pieces produces an error for each quantity of $(d/\varepsilon) \cdot 1/\sqrt{n}$. However, asking groups of (n/d) users to answer each question yields an error proportional to $1/\varepsilon \cdot \sqrt{d/n}$, better by a factor of $\sqrt{d}$. Similar reasons mean that when running a protocol over multiple iterations, it is preferable to have each user participate in only one of the k iterations, rather than devote (ε/k) privacy budget to each round. As noted above, we are also incentivized to pick d and k as small as possible, to further reduce these error bounds.

Data representation. Last, we note that various methods described above have used various (linear) transformations on the users' data. Hadamard transforms are used in the construction of frequency oracles and in gathering marginal statistics, to help maximize the information in data that would otherwise be sparse. Haar wavelet transforms rotate the user data to answer queries related to ranges and quantile queries. Random projections are advocated to reduce the dimensionality of data that may be large and sparse. Finally, histograms can be viewed as a data transformation to reduce the domain size in a structured way. The lesson here is that a transformation like these may help to capture the essence of the data needed to solve a particular problem, and reduce the error.

6 LDP Limitations and Related Models

In this closing section, we reflect on some of the limitations of the Local Differential Privacy model, and some of the variant models that have been proposed as a consequence.

6.1 LDP Limitations

Budget management. A key parameter of any differentially private procedure is the "privacy budget", ε. Despite its centrality to the DP model, there is still much discussion and disagreement around how to set and interpret this parameter (see, for example, Dwork et al. (2020)). In LDP this is a particularly knotty question, since many of the applications aim to make repeated measurements of user data over time. The data collection in Apple operating systems has been criticized, due to the high values of ε used, and the fact that the budget "resets" every day to allow fresh collection, as analyzed by Tang et al. (2017). Other implementations have tried to address this issue, by using "fixed random values" and memoization techniques, but these only provide a partial solution (Erlingsson et al., 2014; Ding et al., 2017). Providing meaningful privacy guarantees over long periods of time remains an open problem.

Strength of ML models. As discussed in the previous section, much work on implementing machine learning models under LDP has focused on models that may be considered more at the more simplistic end: linear regression, or models like SVM and decision trees. Many state-of-the-art results in machine learning are achieved with larger and more complex models, particularly deep neural networks. As noted above, the tradeoff with LDP is different, meaning that large and complex models cannot always be handled well. Work is ongoing to combine the accuracy of deep models with the privacy guarantees of LDP.

Accuracy guarantees. We return to the first example in the introduction: releasing the number of people that hold a particular attribute. We saw that the error in LDP scales with $\sqrt{n}/\varepsilon$, while the error for centralized DP scales with $1/\varepsilon$. The additional dependence on $\sqrt{n}$ may be an Achilles heel for LDP: as we consider increasingly complex tasks, the overall error from privacy quickly stacks up. There are two basic ways to decrease the error of an arbitrary LDP protocol: increase ε or increase n.

Increasing ε means taking an increasingly relaxed notion of privacy, to the point where the precise statement of the guarantee may be statistically meaningless. Growing ε from 1 to 10, say, may look relatively tame, but the consequence is that probabilities of different outcomes are bounded by factors of $\exp(\varepsilon)$, which for $\varepsilon = 10$ is over 22,000. For protocols based on Randomized Response, the probability of reporting a false answer is $1/(\exp(\varepsilon) + 1)$. For ε values of this magnitude, almost every user is reporting their true information, and the guarantees of LDP are consequently quite weak.

Increasing n is perhaps more defensible, but comes at a cost. The error rate decreases only with $\sqrt{n}$, so to halve the error, we need to quadruple the number of participants. For more complex protocols, this means that much larger user populations must be engaged to participate (honestly) in the protocol. Concretely, in Google's deployment, it was noted that at least 10,000 users must report a phenomenon before a clear enough signal could be observed (Erlingsson et al., 2014). Access to tens of thousands, if not millions, of users may be feasible for large technology companies, but is out of reach for lesser groups. This may mean that LDP is primarily of interest only to a minority of organizations.

Vulnerability to 'poisoning'. So-called "poisioning" attacks on machine learning systems are based on the idea of a manipulating a small amount of the training data to force the resulting classifier to achieve poor results. Within the context of local differential privacy, this question asks what influence a small number of colluding users can have on the results? It has been observed that LDP is particularly vulnerable to data poisoning, more so than centralized DP (Cao et al., 2019). The intuition for this is that the random perturbations of randomized response and similar techniques lead us to reweight the averaged responses accordingly, to compensate for the cancellation of the random noise. Input poisoning can take advantage of this reweighting, by throwing all this weight in a particular direction. Different mechanisms vary in their susceptibility to such poisoning (Cheu et al., 2019a), and while some mitigations exist, all LDP methods are vulnerable to poisoning to some extent.

6.2 Alternate models

Federated Learning. The model of Federated Learning is similar to some of the LDP protocols for model training we have seen. In Federated Learning, a central server shares a current model with all users, who each evaluate the model on their local training data, and reply with their suggested updates to the model. Most commonly, these updates are in the form of gradients, which the server can combine by averaging to build the new model. In this form, no effort is made to mask or manipulate the messages from the users. The idea is that, since the users' data remains under their control and is not directly shared with the server, they remain private. However, the revealed gradients can inform on the users' data, particularly if the model is crafted adversarially. This leakage can be limited to just the server if the users establish a secure channel to the server. In order to provide a stronger guarantee, some form of (differential) privacy can additionally be enforced, and research is ongoing into how to achieve the best balance between privacy and utility – see the recent extensive survey due to Kairouz et al. (2021).

Distributed Noise Generation and Secure Multiparty Computation. The goal of the LDP model can be summarized as protecting the input of each user through an appropriate privacy guarantee, and ensuring that the view and output of a data analyst meets a differential privacy guarantee. LDP achieves both simultaneously by enforcing the DP property for each user individually. This implies that DP also holds at the analyst, but as we have seen the accuracy guarantee is weaker than in the centralized DP model. An alternate approach is to use cryptographic techniques to provide privacy for the users, so that the output of the analyst is DP (Dwork et al., 2006). We compare against the centralized DP model, where a typical approach has the analyst compute the exact solution, and add noise from an appropriate distribution (say, Laplace or Gaussian noise). In the "distributed noise generation" approach, each user adds a small fraction (or "share") of that noise, so that the sum of these shares is distributed according to the overall target noise level. This requires the target noise distribution to be "infinitely divisible", but fortunately many distributions appropriate for differential privacy have this property Agarwal et al. (2021); Bagdasaryan et al. (2021); Ghazi et al. (2021); Pagh and Stausholm (2022). To put this into effect, "secure sum" techniques can be used so that the analyst can derive the sum of the submitted (noisy) values, without observing any of the inputs. We could delegate this task to a trusted entity (with a secure hardware implementation), or via a distributed protocol using cryptographic techniques to compute the sum Bonawitz et al. (2017); Bell et al. (2020). The key here is to ensure that the protocol is robust to attempts to evade it, such as if some users collude with the analyst to try to discover the value of one targeted user, and to handle cases when some users may drop out of the protocol (e.g., mobile users moving out of communication range, or running out of battery).

Shuffling. The shuffling model of privacy alters the model of trust by introducing an intermediate step between the users and the data analyst. The intuition is that if we can break the link between the message sent by a user and the identity of that user, then it becomes harder to draw any inference about them, and we can obtain a higher level of (differential) privacy. In the shuffle model, we imagine that there is a "shuffler" who sits between the users and the analyst, and who receives all messages, removes all identifying information about the source, and reveals the multiset of messages to the analyst. Under this model, it is possible to prove "privacy amplification" results: that messages sent with a low privacy guarantee nevertheless provide a stronger privacy guarantee at the analyst. As well as general amplification theorems, results have been shown giving tight bounds for specific problems such as sums and counts (Bittau et al., 2017; Erlingsson et al., 2019; Balle et al., 2019; Cheu et al., 2019b). Importantly, it is observed that users can follow an ε-local DP protocol to determine their (private) messages to the shuffler, and be assured of a stronger ε' DP guarantee for the overall outcome of the shuffled protocol. This motivates further development of LDP protocols which can then be used as a building block within the shuffle model.

To realize the shuffler in practice, we could instantiate a trusted entity to act as the shuffler, who is relied upon to act independently of the analyst. However, this relies on a certain level of trust, so we could instead seek to build a distributed shuffler via cryptographic "mix" networks. There are close connections between distributed noise generation and shuffling: many protocols in the shuffle protocol can be viewed as generating distributed noise, and vice-versa. This implies that secure aggregation techniques can often be applied to build the histogram of outputs from a shuffling protocol Bell et al. (2020).

7 Concluding remarks.

Local Differential Privacy provides a clean definition of privacy that is appropriate for many computations over large, distributed populations who wish to share their data under a suitable privacy guarantee. Several large-scale deployments have demonstrated that LDP is practical for data collection and analysis. However, since accurate and usable results require large populations and larger values of privacy parameter ε, current approaches are reaching the limits of what can be achieved effectively, and generalizations or extensions of the model may be required for more advanced machine learning tasks with fewer participants while still giving strong privacy guarantees.

Acknowledgements. This chapter is based on a tutorial developed in collaboration with Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava and Tianhao Wang (Cormode et al., 2018a). Special thanks to Tianhao Wang for detailed comments and suggestions.

References

Acharya, J., Z. Sun, and H. Zhang, 2019: Hadamard response: Estimating distributions privately, efficiently, and with little communication. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, PMLR, vol. 89 of *Proceedings of Machine Learning Research*, pp. 1120–1129.

Agarwal, N., P. Kairouz, and Z. Liu, 2021: The skellam mechanism for differentially private federated learning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 5052–5064.

Apple, 2017: Apple differential privacy technical overview. https://www.apple.com/privacy/docs/Differential Privacy Overview.pdf.

Bagdasaryan, E., P. Kairouz, S. Mellem, A. Gascón, K. A. Bonawitz, D. Estrin, and M. Gruteser, 2021: Towards sparse federated analytics: Location heatmaps under distributed differential privacy with secure aggregation. *CoRR*, **abs/2111.02356**.

Balle, B., J. Bell, A. Gascón, and K. Nissim, 2019: The privacy blanket of the shuffle model. In *Advances in Cryptology - CRYPTO*, Springer, vol. 11693 of *Lecture Notes in Computer Science*, pp. 638–667.

Bassily, R., K. Nissim, U. Stemmer, and A. Thakurta, 2020: Practical locally private heavy hitters. *J. Mach. Learn. Res.*, **21**, 16:1–16:42.

Bassily, R. and A. D. Smith, 2015: Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, ACM, pp. 127–135.

Bell, J. H., K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, 2020: Secure single-server aggregation with (poly)logarithmic overhead. In *ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 1253–1269.

Bittau, A., Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnés, and B. Seefeld, 2017: Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, ACM, pp. 441–459.

Bonawitz, K. A., V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, 2017: Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 1175–1191.

Bun, M., J. Nelson, and U. Stemmer, 2018: Heavy hitters and the structure of local privacy. In den Bussche, J. V. and M. Arenas, eds., *Proceedings of ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, ACM, pp. 435–447.

Cao, X., J. Jia, and N. Z. Gong, 2019: Data poisoning attacks to local differential privacy protocols. *CoRR*, **abs/1911.02046**.

Chang, J. C.-N. and A. Thakurta, 2018: Autocompletion in local differential privacy. In *IEEE Symposium on Security and Privacy*, p. 2.

Chen, R., H. Li, A. K. Qin, S. P. Kasiviswanathan, and H. Jin, 2016: Private spatial data aggregation in the local setting. In *IEEE International Conference on Data Engineering, ICDE*, IEEE Computer Society, pp. 289–300.

Cheu, A., A. D. Smith, and J. R. Ullman, 2019a: Manipulation attacks in local differential privacy. *CoRR*, **abs/1909.09630**.

Cheu, A., A. D. Smith, J. R. Ullman, D. Zeber, and M. Zhilyaev, 2019b: Distributed differential privacy via shuffling. In *Advances in Cryptology - EUROCRYPT*, Springer, vol. 11476 of *Lecture Notes in Computer Science*, pp. 375–403.

Cormode, G., S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang, 2018a: Privacy at scale: Local differential privacy in practice. Tutorial at SIGMOD and KDD.

Cormode, G., T. Kulkarni, and D. Srivastava, 2018b: Marginal release under local differential privacy. In Das, G., C. M. Jermaine, and P. A. Bernstein, eds., *Proceedings of the International Conference on Management of Data, SIGMOD*, ACM, pp. 131–146.

—, 2019: Answering range queries under local differential privacy. *Proc. VLDB Endow.*, **12(10)**, 1126–1138.

Cormode, G. and K. Yi, 2020: *Small Summaries for Big Data*. CUP.

Ding, B., J. Kulkarni, and S. Yekhanin, 2017: Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pp. 3571–3580.

Dinur, I. and K. Nissim, 2003: Revealing information while preserving privacy. In Neven, F., C. Beeri, and T. Milo, eds., *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM, pp. 202–210.

Duchi, J. C., M. I. Jordan, and M. J. Wainwright, 2013: Local privacy and statistical minimax rates. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, IEEE Computer Society, pp. 429–438.

Dwork, C., K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, 2006: Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology - EUROCRYPT*, Springer, vol. 4004 of *Lecture Notes in Computer Science*, pp. 486–503.

Dwork, C., N. Kohli, and D. Mulligan, 2020: Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality*, **9(2)**.

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

Erlingsson, Ú., V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, 2019: Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms, SODA*, SIAM, pp. 2468–2479.

Erlingsson, Ú., V. Pihur, and A. Korolova, 2014: RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 1054–1067.

Evfimievski, A. V., J. Gehrke, and R. Srikant, 2003: Limiting privacy breaches in privacy preserving data mining. In Neven, F., C. Beeri, and T. Milo, eds., *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM, pp. 211–222.

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

Ghazi, B., N. Golowich, R. Kumar, R. Pagh, and A. Velingker, 2021: On the power of multiple anonymous messages: Frequency estimation and selection in the shuffle model of differential privacy. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, Springer, vol. 12698 of *Lecture Notes in Computer Science*, pp. 463–488.

Kairouz, P., H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, 2021: Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, **14(1-2)**, 1–210.

Nguyên, T. T., X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, 2016: Collecting and analyzing data from smart device users with local differential privacy. *CoRR*, **abs/1606.05053**.

Pagh, R. and N. M. Stausholm, 2022: Infinitely divisible noise in the low privacy regime. In *International Conference on Algorithmic Learning Theory*, PMLR, vol. 167 of *Proceedings of Machine Learning Research*, pp. 881–909.

Pihur, V., A. Korolova, F. Liu, S. Sankuratripati, M. Yung, D. Huang, and R. Zeng, 2018: Differentially-private "draw and discard" machine learning. *CoRR*, **abs/1807.04369**.

Qin, Z., T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, 2017: Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security, CCS*, ACM, pp. 425–438.

Ren, X., C. Yu, W. Yu, S. Yang, X. Yang, J. A. McCann, and P. S. Yu, 2018: Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Trans. Inf. Forensics Secur.*, **13(9)**, 2151–2166.

Shin, H., S. Kim, J. Shin, and X. Xiao, 2018: Privacy enhanced matrix factorization for recommendation with local differential privacy. *IEEE Trans. Knowl. Data Eng.*, **30(9)**, 1770–1782.

Smith, A. D., A. Thakurta, and J. Upadhyay, 2017: Is interaction necessary for distributed private learning? In *IEEE Symposium on Security and Privacy*, IEEE Computer Society, pp. 58–77.

Tang, J., A. Korolova, X. Bai, X. Wang, and X. Wang, 2017: Privacy loss in apple's implementation of differential privacy on macos 10.12. *CoRR*, **abs/1709.02753**.

Wang, D., M. Gaboardi, and J. Xu, 2018a: Empirical risk minimization in non-interactive local differential privacy revisited. In *Advances in Neural Information Processing Systems*, pp. 973–982.

Wang, N., X. Xiao, Y. Yang, T. D. Hoang, H. Shin, J. Shin, and G. Yu, 2018b: Privtrie: Effective frequent term discovery under local differential privacy. In *34th IEEE International Conference on Data Engineering*, IEEE Computer Society, pp. 821–832.

Wang, T., J. Blocki, N. Li, and S. Jha, 2017: Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium, USENIX Security*, USENIX Association, pp. 729–745.

Wang, T., B. Ding, J. Zhou, C. Hong, Z. Huang, N. Li, and S. Jha, 2019: Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the ACM International Conference on Management of Data*, ISBN 9781450356435, p. 159–176.

Wang, T., N. Li, and S. Jha, 2019: Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*.

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

Zhang, Z., T. Wang, N. Li, S. He, and J. Chen, 2018: CALM: consistent adaptive local marginal for marginal release under local differential privacy. In Lie, D., M. Mannan, M. Backes, and X. Wang, eds., *Proceedings of ACM SIGSAC Conference on Computer and Communications Security, CCS*, ACM, pp. 212–229.

Zheng, K., W. Mou, and L. Wang, 2017: Collect at once, use effectively: Making non-interactive locally private learning possible. In *Proceedings of International Conference on Machine Learning, ICML*, PMLR, vol. 70 of *Proceedings of Machine Learning Research*, pp. 4130–4139.