Survey of Differential Privacy

James Zhang
Department of Computer Science
Cornell University
rz234@cornell.edu

Owen Oertell
Department of Computer Science
Cornell University
ojo2@cornell.edu

Stephanie Ma
Department of Computer Science
Cornell University
ym363@cornell.edu

1 Introduction

As datasets become filled with more and more sensitive information, it becomes of greater importance that this sensitive information does not get revealed. However, maintaining usefulness of the data to allow researchers to draw conclusions about the data is a rival concern. We have therefore seen a significant rise in attention towards differential privacy. In particular, a recent series of papers has formalized the notion of *differential privacy* [8]. That is, a database privatization mechanism satisfies differential privacy if the addition or removal of a single element of the database does not change the probability of any outcome of the mechanism by more than some small amount, ϵ. The motivation behind this definition is that although one will be able to determine general patterns from the data, it's impossible to determine specific values of an individual data point.

Indeed differential privacy is not merely the only notion of database privacy. One particular, stronger, notion of privacy is *distributional privacy*. Distributional privacy posits that for a database drawn from a distribution, then privacy-preserving mechanisms should reveal only information about the underlying distribution and nothing else. That is, given a database $\mathcal{D}$ over data points, then there is not a significant change in the probability of any outcome when coming from an entirely new database $\mathcal{D}'$. Such a notion of privacy is less common and we will stick with the prevailing notion of differential privacy for the remainder of this report.

Up until the point of [3], modern differential privacy research had shown a series of lower bounds. In particular, [4] showed that one must add perturbation of results to counting queries on the order of magnitude of $\Omega(\sqrt{n})$ for non-interactive databases. Likewise [7] showed that one must add noise in accordance with a notion of sensitivity of the function. Both of these results hint that when one allows $\mathcal{O}(n)$ queries to the database, adding noise is crucial to preserving a sense of differential privacy. Further, [12] showed that if one-way functions exist, then there is no polytime algorithm which takes in a database and $\mathcal{O}(n^2)$ arbitrary efficiently computable counting queries, and returns an approximate answer to each query while still satisfying differential privacy. Interestingly, these results also hold for "simple" queries, or queries that are computable with AC^0 circuits.

In light of these results, other work has focused on access mechanisms that permit merely a sub-linear number of queries. However, due to the fact that the database must be destroyed afterward, its usefulness is relatively limited. Thus, coming up with other ways to maintain usefulness even with a linear number of queries is desirable.

One of the most common types of queries to a database, and those that will be focused on for the remainder of this survey are halfspace queries. In particular, they have the form of "how many points satisfy predicate φ?". Although this particular class of queries does not encompass everything that

Preprint. Under review.

can be asked of the database, focusing on these queries is able to circumvent some of the lower bounds shown for subset sum queries [7].

One of the most natural things to do when it comes to achieving privacy is adding noise. There are two main schools of thought when it comes to adding noise. In particular, we have *input perturbation techniques* which focus on adding noise to the underlying data and then answering truthfully with the access mechanism. Conversely, there is also *output perturbation techniques* where the true answers are queried from the database but the ending response is perturbed by data [7]. Both of these techniques have their own strengths and weaknesses. In fact, [1] showed that it was possible to reconstruct the *distribution* of the original data and to build classifiers that behave nearly identically to those built with the original data. In what follows we consider what would be most similar to an output perturbation technique.

2 Organization

We discuss an introduction to differential privacy in the first section, cover the related works, and further the current state of differential privacy in the third section. We discuss interval queries and the lower bounds in the fourth and fifth sections. Then cover how to answer halfspace queries. Finally, we discuss future work and open problems.

3 Related Works and the State of Differential Privacy

In the paper [6], Cynthia Dwork provided a brief introduction to the idea of differential privacy and then proceeded to give a survey of results in this field from different papers. One of the main mechanisms used to achieve differential privacy is the *Laplace mechanism* where the output of the query is perturbed by Laplacian noise. We will formally introduce this mechanism in the next section.

The first algorithm she provided is a differentially private algorithm for statistical data inference [8]. Given a database where each row contains k boolean attributes, together with the incidence settings of any ℓ attributes (that is, the proportion of rows satisfying all ℓ attributes), we want to query the incidence settings of any 2ℓ attributes. The key to achieving this goal is through Bayes's Rule and de Morgan's Rules. For example, if A and B are sets of attributes, then $\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B|A]$. Note that if the two probabilities $\Pr[A]$ and $\Pr[B|A]$ are not exact values but only approximates, then multiplying the two probabilities increases the error additively.

To determine $\Pr[B|A]$ accurately, we first find a *heavy set* of A, which is a subset of the database where A occurs more than average. We then find the proportion of B within this set. Intuitively, if the proportion of B deviates significantly from the average in this heavy set of A, then A and B are highly correlated. To obtain a heavy set of A, we sample a random subset from the database, and with constant probability, the proportion of A will be one standard deviation above the average. Therefore, the curator can release a perturbed version of incident rates of A and B to guarantee differential privacy.

The second algorithm is for contingency table release [2]. In this problem, we again have a database where each row contains k boolean attributes, but we are instead given the *contingency table*, which counts the number of rows that satisfy a given subset of the k boolean attributes. Since each attribute is either true or false, we have a k-dimensional table of 2^k cells of numbers. The table is likely to be sparse since it has exponentially many cells, so we typically release the *marginal table*, which records the number of rows that satisfy some small subset of attributes.

The first approach is to add noise to each of the cells in the table using the Laplace mechanism. However, since the errors are additive, the errors for low-dimensional marginals are exponentially large compared to the errors that we add to each cell. The second approach is to directly perturb the marginals using the Laplace mechanism. This does not lead to huge errors, but it can cause the marginal values to be inconsistent. That is, users might get two different values by summing up the marginals in two different ways. The last approach transforms the data into the Fourier domain, perturbs the Fourier coefficients, and then transforms the data back to the contingency table domain. In addition, the authors of the paper used linear programming to obtain a non-negative perturbed dataset and then used rounding to turn each entry into an integer.

The third algorithm answers interval and halfspace queries [3], which we will discuss in detail for the rest of this report.

4 Preliminaries

Before diving into the details of how we preserve differential privacy on some fixed types of queries to the database, we first present crucial definitions for related differential privacy concepts.

There are generally two categories of mechanisms that preserve database privacy: interactive and non-interactive database access mechanisms. We use $A(\cdot)$ to represent a database mechanism. An interactive mechanism, as its name suggests, allows the curator who sits in between the users and the database to answer privacy-preserving queries. In this process, the curator follows the mechanism to "anonymize" sensitive data information and send privacy-preserving responses to users. The database is never released to the public. Given a database D with a user's query Q, the curator outputs the response $A(D, Q)$ following the interactive mechanism. On the other hand, in the non-interactive setting, the curator processes the whole database at once and follows the mechanism to generate a new privacy-preserving database with all sensitive information removed. The curator then can release the new database to the public, and they can keep the original database secretly or destroy it as they wish. Formally, given a database D, the curator outputs a sanitized database $A(D)$ to the public.

Because the mechanisms are almost always probabilistic, we can formalize differential privacy using the definitions below:

Definition 4.1 *Given a database D, we say D' is a neighbor of D if they only differ by a single data point.*

Definition 4.2 *Given an interactive database access mechanism A, we say A satisfies α-differential privacy if for all neighboring database D_1 and D_2, for all queries Q,*

$$\forall x \Pr[A(D_1, Q) = x] \leq e^{\alpha} \Pr[A(D_2, Q) = x].$$

Given a non-interactive database access mechanism A, we say A satisfies α-differential privacy if for all neighboring database D_1 and D_2, for all sanitized database $\hat{D}$,

$$\Pr[A(D_1) = \hat{D}] \leq e^{\alpha} \Pr[A(D_2) = \hat{D}].$$

Intuitively speaking, a database preserves differential privacy if we cannot picture the original database, i.e., based on the mechanism's output it's hard to determine if the response comes from which of the two neighboring databases.

In addition to ensuring sensitive data is not compromised, in the non-interactive setting, we would like the sanitized database to be 'useful'. In other words, we want the publicized, sanitized database to have similar responses as the original database.

Definition 4.3 *We say a query Q_φ is a predicate query for predicate φ if for any database D,*

$$Q_\varphi(D) = \frac{|\{x \in D : \varphi(x)\}|}{|D|}.$$

Definition 4.4 *Given a non-interactive database mechanism A, we say A is (ε, δ)-useful for predicate query class C if with probability $1 - \delta$, for all predicate query $Q \in C$ and for all database D,*

$$|Q(A(D)) - Q(D)| \leq \varepsilon.$$

In the remainder of this paper, we mainly care about non-interactive database mechanisms that generate a sanitized database specialized for a single class of predicate queries. Before we proceed, however, we introduce a very straightforward interactive mechanism PRIVATE$_\alpha$ that achieves α-differential privacy.

Definition 4.5 *Given any database and predicate query Q, we define PRIVATE$_\alpha(D, Q) := Q(D) + Z$, where Z is a random variable drawn from the Laplace distribution, i.e., $Z \sim Lap(1/(n\alpha))$.*

Theorem 4.6 *PRIVATE$_\alpha(D, Q)$ preserves α-differential privacy.[7]*

Proof. For arbitrary adversary $\mathcal{A}$, we have a query function $f_t(x) : D^n \to \mathbb{R}^d$ be a query function.

By the law of conditional probability

$$\frac{\Pr[\text{PRIVATE}_\alpha(D,Q)=t]}{\Pr[\text{PRIVATE}_\alpha(D',Q)=t]} = \prod_i \frac{\Pr[\text{PRIVATE}_\alpha(D,Q)_i=t_i|t_1,\ldots,t_{i-1}]}{\Pr[\text{PRIVATE}_\alpha(D',Q)_i=t_i|t_1,\ldots,t_{i-1}]}$$

For each term in the product, we fix the first $i-1$ coordinates of t. This fixes the values of PRIVATE$_\alpha(D,Q)_i$ and PRIVATE$_\alpha(D',Q)_i$. We therefore know that the conditional distributions are laplacians and so we can bound each term and their product as:

$$\begin{aligned}\prod_i \frac{\Pr[\text{PRIVATE}_\alpha(D,Q)_i=t_i|t_1,\ldots,t_{i-1}]}{\Pr[\text{PRIVATE}_\alpha(D',Q)_i=t_i|t_1,\ldots,t_{i-1}]} &\le \prod_i \exp(|\text{PRIVATE}_\alpha(D,Q)_i - \text{PRIVATE}_\alpha(D',Q)_i|/\lambda) \\ &= \exp(\|\text{PRIVATE}_\alpha(D,Q)_i - \text{PRIVATE}_\alpha(D',Q)_i\|_1/\lambda\end{aligned}$$

Where we complete the proof using the bound that $S(Q) \le \lambda\epsilon$, for all t. □

5 Interval Queries

In this section, we consider an efficient non-interactive mechanism that achieves α-differential privacy and (ε, δ)-usefulness. This mechanism smartly uses the idea of binary search and builds the sanitized databases by utilizing mechanism PRIVATE$_\alpha$. It's also efficient, which means that it can be widely applied to a lot of scenarios in daily use. However, this mechanism will only work for discretized databases, as we will see soon; if it's not discretized, then the mechanism is not guaranteed to be efficient anymore.

We first define what interval queries are: interval query asks the number of data points that sit in that interval. The following definition is only for one-dimensional databases, but we can easily extend this (and also the following mechanism) to other finite-dimensional databases.

Definition 5.1 *We say a query $Q_{[a,b]}$ is an interval query if for any database D,*

$$Q_{[a,b]}(D) = \sum_{x\in D} \frac{\mathbb{1}[a \le x \le b]}{|D|}.$$

Now it's time to introduce the mechanism; without loss of generality, we assume all data points are scattered among the interval $[0, 1]$:

Definition 5.2 *Given a database D that contains n data points and is discretized to b bits, we use PRIVATE$_{\alpha'}$ to query D and use binary searches to partition the entire interval $[0, 1]$ into $2/\varepsilon_1$ sub-intervals, such that the number of points each contains is in $[n(\varepsilon_1/2 - \varepsilon_2), n(\varepsilon_1/2 + \varepsilon_2)]$, i.e., each sub-interval has a probability mass in $[\varepsilon_1/2 - \varepsilon_2, \varepsilon_1/2 + \varepsilon_2]$. The sanitized database can then be constructed by putting $n(\varepsilon_1/2)$ number of points into each sub-interval.*

The exact positions of data points don't matter as long as they are in the right sub-intervals.

Theorem 5.3 *This mechanism can be done efficiently, and achieves α-differential privacy and (ε, δ)-usefulness, if $\alpha' = \frac{\varepsilon\alpha}{4b}$, $\varepsilon_1 = \varepsilon/2$, $\varepsilon_2 = \varepsilon^2/8$, and*

$$n \ge \mathcal{O}(\frac{b(\log b + \log(1/\varepsilon\delta))}{\alpha\varepsilon^3}).$$

Proof. Because D is discretized to b bits, the binary search needs at most b depth to differentiate all distinct data points. Since running binary search once decides a boundary of a sub-interval, we need to perform $\mathcal{O}(b \cdot \frac{2}{\varepsilon_1}) = \mathcal{O}(\frac{b}{\varepsilon})$ number of PRIVATE$_{\alpha'}$ queries in total, so the mechanism can be done in $\mathcal{O}(\frac{nb}{\varepsilon})$.

Figure 1: The sanitized database cannot correctly answer for those intervals that only partially intersect with the query interval.

Because PRIVATE$_{\alpha'}$ preserves α'-differential privacy and we execute PRIVATE$_{\alpha'}$ $\frac{2b}{\varepsilon_1}$ number of times, we know that for all neighboring database D_1, D_2, for all queries $Q_1, \cdots, Q_{2b/\varepsilon_1}$ and for all query answers $x_1, \cdots, x_{2b/\varepsilon_1}$,

$$\Pr[\forall i \in [2b/\varepsilon_1]\text{PRIVATE}_{\alpha'}(D_1, Q_i) = x_i] \leq e^{\alpha' \cdot \frac{2b}{\varepsilon_1}} \Pr[\forall i \in [2b/\varepsilon_1]\text{PRIVATE}_{\alpha'}(D_2, Q_i) = x_i],$$

where $\alpha' \cdot \frac{2b}{\varepsilon_1} = \alpha$. Because the generated sanitized database is uniquely determined by the responses of PRIVATE$_{\alpha'}$, we can conclude that for all sanitized database $\hat{D}$,

$$\Pr[A(D_1) = \hat{D}] \leq e^{\alpha} \Pr[A(D_2) = \hat{D}],$$

which means our mechanism preserves α-differential privacy.

Now we prove that the mechanism is (ε, δ)-useful. Because PRIVATE$_{\alpha'}$ responses are not precise, the mechanism may declare that each sub-interval has probability mass in $[\varepsilon_1/2 - \varepsilon_2, \varepsilon_1/2 + \varepsilon_2]$, but in reality it's not. We formalize this into an event FAILURE: there exists some sub-interval such that its actual probability mass is less than $\varepsilon_1/2 - \varepsilon_2$ or greater than $\varepsilon_1/2 + \varepsilon_2$, i.e., it deviates from ε_1 for more than ε_2.

By CDF of Laplace distribution, $\Pr[Lap(1/\alpha' n) \geq \varepsilon_2] \leq e^{\alpha' n \varepsilon_2}$. Since we run PRIVATE$_{\alpha'}$ at most $2b/\varepsilon_1$ times, by union bound, we have

$$\Pr[\text{FAILURE}] \leq 2b/\varepsilon_1 \cdot e^{\alpha' n \varepsilon_2}$$

Plugging in the lower bound on n, we get

$$\Pr[\text{FAILURE}] \leq \delta.$$

We remain to show that when FAILURE doesn't happen, i.e., when all sub-interval's probability mass is in the range, the difference of query responses of the original and sanitized databases doesn't exceed ε.

We first observe that potentially there is $\frac{2}{\varepsilon_1} \cdot \varepsilon_2 = \varepsilon/2$ error, because for each sub-interval, the number of data points in the sanitized database deviates at most $n \cdot (\varepsilon/2)$ from that in the original database. In addition, the query answer to the sanitized database is almost always incorrect regarding how many points are in the two sub-intervals that the interval query partially intersects with, as Figure 1 shows. Therefore, we can potentially get another $2 \cdot (\varepsilon_1/2) = \varepsilon/2$ error.

In total, the query answer from the sanitized database has at most ε error regarding that from the original database. And because the even FAILURE will not happen for more than $1 - \delta$ probability, we can conclude that our mechanism is (ε, δ)-useful. □

6 Lower Bounds

In this section, we present lower bounds for non-discretized databases. In particular, we represent a database as a collection of points in $\mathbb{R}$ and we do this in the concept class $\mathcal{C}$ for interval queries, or queries that say "How many points are contained within the following interval"). Clearly, this is a 1-dimensional version of a specific set of halfspace queries.

Suppose that there exists some mechanism that allows for usefully answering these queries while still preserving α-differential privacy. Then, we would be able to binary search and answer the median query with values that fall between $1/2 - \delta$ and $1/2 + \delta$. using A. We collate this into a theorem:

Theorem 6.1 *No mechanism A can answer median queries M with outputs that fall between $1/2 - k, 1/2 + k$ percentile with positive probability on any real-valued database D while still preserving α differential privacy for $k < 1/2$ and any α.*

Proof. Consider a real-valued database containing elements in the interval $[0, 1]$. Let $D_0 = (0, \ldots, 0)$, the database containing n 0s.Then we must have by the assumption that $\Pr[A(D_0, M) = 0] > 0$. However, since $[0, 1]$ is a continuous interval, we know that there must be some value $v \in [0, 1]$ where $\Pr[A(D_0, M) = v] = 0$. We create a second database that contains these values. In particular, let $D_n = (v, \ldots, v)$. That is, it contains n values of v. Likewise, for some D_i, let this represent a database containing i values of v and the rest $(n - i)$ values of 0s. We know that $\Pr[A(D_n, M) = v] > 0$ by assumption again. However, for some i, we know that there must be a switch from when $\Pr[A(D_i, M) = v] = 0$ (which is the case for $D = D_0$) and $\Pr[A(D_i, M) = v] > 0$ (which is the case when $D = D_n$. But at the instance D_i to $D_i + 1$, these databases differ by only a single element, which violates differential privacy for all α. □

Essentially what this is saying is that it's impossible to answer the median queries truthfully (i.e. have positive probability of being correct), while still preserving any notion of differential privacy. This argument is indeed similar to those which we have seen in class. This then leads us to the corollary.

Corollary 6.2 *No mechanism can be (ϵ, δ)-useful for the class of interval queries, nor for any class C that generalizes interval queries for higher dimensions (for example, halfspaces, axis-aligned rectangles, or spheres), while preserving α-differential privacy for any $\epsilon = o(n)$ and any α*

Proof. The proof here is quite straightforward. Consider any real-valued database that contains elements in the interval $[0, 1]$. Assume that A is (ϵ, δ) useful for interval queries and preserves differential privacy. By construction, we show that we can answer median queries with outputs that fall between $1/2 - k, 1/2 + k$ percentile, and by theorem 6.1 contradicts the possibility of satisfying differential privacy.

The construction is as follows. Let $\hat{D} = A(D)$, that is, it generates the static database using the privacy-preserving mechanism. Our algorithm then performs a binary search on this $\hat{D}$. Specifically, we want to find some interval $[0, a]$ that contains $n/2 + \epsilon$ points. Since we know that all interval queries on $\hat{D}$ are correct within $\pm\epsilon$, we know that this binary search will be able to find our interval $[0, a]$. However, this means that we have found the median, which is impossible. □

In light of this, and to avoid discretizing the database, we relax our definition of usefulness. In particular,

Definition 6.3 (usefulness definition 2) *A database mechanism A is $(\epsilon, \delta, \gamma)$-useful for queries in class C according to some metric d if w.p. $1 - \delta$, for every $Q \in C$, and every database D, $|Q(A(D)) - Q'(D)| \leq \epsilon$ for some $Q' \in C$ such that $d(Q, Q') \leq \gamma$*

Essentially, this definition is saying that some mechanism is $(\epsilon, \delta, \gamma)$-useful in the case that with high probability, the output of the query on the filtered database is similar to some other, similar query, that would be answered on the full database. In essence it's saying that a mechanism is useful if it is ϵ close to answering some neighboring query completely correctly, but that this is still useful since the queries are close together.

7 Answering Halfspace Queries

We now turn to a problem that is useful in various fields of math and computer science: Halfspace queries.

Definition 7.1

$$H_y(D) = \frac{\left|\left\{x \in D : \sum_{i=1}^{d} x_i \cdot y_i \geq 0\right\}\right|}{|D|}.$$

Intuitively, a halfspace query asks for the proportion of points in the database that lie on one side of a halfspace. Our goal is to give a *non-interactive differentially-private* algorithm that is $(\varepsilon, \delta, \gamma)$-*useful* for halfspace queries.

We have the following definitions:

Definition 7.2 *Define the distance between a point x and a halfspace H_y by $d(x, H_y) = \frac{|x \cdot y|}{||x||}$. Define the distance between two halfspaces H_{y_1} and H_{y_2}, $d(y_1, y_2)$, to be the sine of the angle between y_1 and y_2. If $d(y_1, y_2) \leq \gamma$, we say that they're* γ-close.

With these definitions, we can rephrase our goal: Given a halfspace H_{y_1}, output a value v such that with high probability, $|v - H_{y_2}(D)| < \varepsilon$ for some H_{y_2} that is γ-close to H_{y_1}. Note that even if H_{y_1} and H_{y_2} are close, the result of the halfspace queries $H_{y_1}(D)$ and $H_{y_2}(D)$ are not necessarily close, because there may be many data points lying in one halfspace but not the other. However, this is unlikely to happen if γ is small. One can think of γ as similar to the concept of *margin* in machine learning.

The main challenge to achieving our goal is the lower bound given in the previous section. Since our goal is to release a non-interactive database, we need much more than a sublinear number of queries. Therefore, we will discretize our database as follows:

Definition 7.3 *A halfspace query H_y is* b-discretized *if y_i can be specified with b bits for each $i \in [d]$.*

Another tool that will be very helpful in our algorithm is random projections:

Definition 7.4 *A* random projection *from $\mathbb{R}^d$ onto $\mathbb{R}^k$ is given by a $d \times k$ matrix M where each entry is sampled uniformly and independently from $\{-1, 1\}$. The result of projecting the point $x \in \mathbb{R}^d$ onto $\mathbb{R}^k$ using M is $\frac{1}{\sqrt{k}} xM$.*

Note that we renormalize the projected vector by dividing it by its length, which is $\sqrt{k}$.

The reason that random projections help answer halfspace queries is the following theorem:

Theorem 7.5 (Johnson-Lindenstrauss) *Let P be a random projection from $\mathbb{R}^d$ to $\mathbb{R}^k$, and let $x, y \in \mathbb{R}^d$. Then*

$$\Pr\left[|d(x, H_y) - d(P(x), H_{P(y)})| \geq \frac{\gamma}{4}\right] \leq 2e^{-((\gamma/16)^2 - (\gamma/16)^3)k/4}.$$

In other words, it's very unlikely for a random projection to change the distance between a point and a hyperplane significantly.

We now describe the algorithm given in the paper [3].

Algorithm 1 Differentially Private Halfspace Query Access Mechanism (Curation Stage)

1: Randomly sample m random projections $P_1, \ldots, P_m$ from $\mathbb{R}^d$ onto $\mathbb{R}^k$ and apply them to the database.
2: For each random projection P_i, select a *net of halfspaces* N_i such that for every $y_1 \in \mathbb{R}^k$, there exists $H_{y_2} \in N_i$ such that H_{y_1} and H_{y_2} are $\frac{3\gamma}{4}$-close.
3: We then make halfspace queries on the net of halfspaces in a differentially private way using the Laplace mechanism and release the results.
4: We also release the random projections $P_1, \ldots, P_m$.
5: The original database can now be destroyed.

Algorithm 2 Differentially Private Halfspace Query Access Mechanism (Query Stage)

1: **Input**: given an arbitrary halfspace query H_y.
2: We first project the halfspace using the same set of random projections as in the curation stage.
3: For each projection P_i, we find and query the halfspace in the net N_i that is closest to $P_i(y)$.
4: We return the median value of the queries.

The algorithm has the following guarantee:

Theorem 7.6 *The above algorithm is $(\varepsilon, \delta, \gamma)$-useful while maintaining α-differential privacy for a database of size $poly(\log(\frac{1}{\delta}), \frac{1}{\varepsilon}, \frac{1}{\alpha}, b, d)$, for constant γ.*

Here is the sketch of the proof:

We are choosing m random projections from $\mathbb{R}^d$ to $\mathbb{R}^k$ where there are n rows in the dataset. By Johnson-Lindenstrauss, we can choose k such that the probability of $d(x, H_y)$ changing by more than $\frac{\gamma}{4}$ is at most $\frac{\varepsilon_1}{4}$. This gives us

$$k \geq \frac{4\ln(8/\varepsilon_1)}{(\gamma/16)^2 - (\gamma/16)^3}.$$

We say that a projection P *makes a mistake* on x and H_y if $d(x, H_y) \geq \frac{\gamma}{4}$, but $\operatorname{sgn}(x \cdot y) \neq \operatorname{sgn}(P(x) \cdot P(y))$. Intuitively, the point x is quite far away from the hyperplane, and it's unlikely for a random projection to change the distance significantly, but P changed the distance so much that after the projection, the point lies on the other side of the hyperplane.

The value k is chosen such that the probability that a random projection makes a mistake on x and H_y is at most $\frac{\varepsilon_1}{4}$. Thus, the expected number of mistakes that a random projection makes on a dataset with n rows is $\frac{\varepsilon_1 n}{4}$. Therefore, by Markov's inequality, the probability that a random projection makes more than $\varepsilon_1 n$ mistakes on that dataset is at most $\frac{1}{4}$.

Now we use a Chernoff bound to bound the probability that more than half of the projections (that is, $m/2$ projections) make more than $\varepsilon_1 n$ mistakes. By Chernoff bound, that probability is at most $e^{-m/12}$. Next, since each b-discretized subspace is specified by bd bits, there can only be 2^{bd} different possible b-discretized subspaces, so by a union bound, the probability that more than half of the projections make more than $\varepsilon_1 n$ mistakes relative to any discretized subspace is at most $\delta_1 = 2^{bd} e^{-m/12}$. We can solve for m in terms of δ_1:

$$m \geq 12\left(\ln\frac{1}{\delta_1} + \ln(2)bd\right).$$

We now turn to the differential privacy guarantee. First, we count the number of halfspaces in each of our net of halfspaces. To guarantee that our halfspaces are dense enough such that any other halfspace is $\frac{3\gamma}{4}$ close to one of them, we use normal vectors v where each of v's entries goes from -1 to 1 in $\frac{3\gamma}{4}$ intervals, and we normalize v. In this way, we only need $|N_i| = O(1/\gamma^{k-1})$ halfspaces in each of our nets.

For each of the halfspaces, we make a privacy-preserving query using the Laplace mechanism $\text{PRIVATE}_{\alpha/(m|N_i|)}(P_i(D), H_y)$. For a projected database $P_i(D)$, we ask $m|N_i|$ queries in total, so we get α-differential privacy. By the union bound and the CDF of the Laplace distribution, the probability δ_2 that any query output differs from $H_y(P_i(D))$ by more than ε_2 is at most

$$m \cdot O(1/\gamma^{k-1}) e^{-(\varepsilon_2 n\alpha)/(mO(1/\gamma^{k-1}))}.$$

We can then solve for n in terms of $\delta_2, \varepsilon_2, m$, and k.

Putting everything together, by the analysis of the projections, with probability at least $1 - \delta_1$, using a nearby query ($\frac{3\gamma}{4}$ close) results in fewer than half of the projections making more than $\varepsilon_1 n$ mistakes, and since we're taking the median of the results, we know that with probability at least $1 - \delta_1$, we get at most ε_1 error. In addition, we are choosing n large enough such that with probability at least $1 - \delta_2$, the privacy-preserving queries introduce at most ε_2 error. Therefore, the theorem follows by choosing $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{2}$ and $\delta_1 = \delta_2 = \frac{\delta}{2}$.

8 Future Work and Open Problems

One of the most interesting new directions in differential privacy is the intersection between machine learning and differential privacy. In some sense, these goals are not in direct opposition. Machine learning attempts to learn a function that describes the distribution of the data, not individual points. Differential privacy, in the same vein attempts to obscure the individual points and leave only the underlying distribution. Indeed, there exist differentially private versions of flavors of stochastic gradient descent [11]. Even further, there exist differentially private versions of the multiplicative weights algorithm [5]. Notable, however, many of these algorithms rely on the additive nature of (ϵ, δ)

differential privacy. Although there has been some advances in learning like differentially private risk minimization [1] and learning SVMs in a differentially private manner [10], in the case of more advanced models like large language models and transformers, concerns may arise of such models in fact memorizing the dataset, and current work has been done to learn (and finetune) large language models in a differentially private manner [13].

Another research direction is how to deal with data that is not scalar/vector. That is, such data which is scalar or vector lends itself well to α-differentially private algorithms where they add noise to the output/perturb the data slightly. On the other hand, when the data is in the form of graphs, it becomes less clear how to perturb the data. Some research has been done to try and alleviate this problem [9]. However, it still remains an open question how best to handle such situations.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450, 2000.

[2] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 273–282. Association for Computing Machinery, Inc., June 2007.

[3] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. *CoRR*, abs/1109.2229, 2011.

[4] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, page 202–210, New York, NY, USA, 2003. Association for Computing Machinery.

[5] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and trends in theoretical computer science. Now, 2014.

[6] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, page 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[8] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *24th Annual International Cryptology Conference (CRYPTO 2004)*, volume 3152 of *Lecture Notes in Computer Science*, pages 528–544. Springer Verlag, August 2004.

[9] Tamara T. Mueller, Dmitrii Usynin, Johannes C. Paetzold, Daniel Rueckert, and Georgios Kaissis. Sok: Differential privacy on graph-structured data, 2022.

[10] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *CoRR*, abs/0911.5708, 2009.

[11] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248, 2013.

[12] Jonathan R. Ullman. Answering n^{2+o(1)} counting queries with differential privacy is hard. *CoRR*, abs/1207.6945, 2012.

[13] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, 2022.