Privacy in Statistics and Machine Learning — Spring 2021
Lecture 13 & 14: Factorization and Projection Mechanisms

Adam Smith and Jonathan Ullman

So far we've talked about *linear query release* as a natural and important problem that captures many of the things a data collector would want to do with a sensitive dataset. However, with the lone exception of the binary tree mechanism, we haven't seen interesting algorithms for this problem except for adding Laplace or Gaussian noise calibrated to sensitivity. In the next couple of lectures we're going to do more of a deep dive on this problem and see a variety of exciting algorithms for query release. In this lecture we'll focus on a natural framework for query-release algorithms based on two ideas for reducing the error *factorization* and *projection*. These two tools are often combined in a particular way, leading to what is called the *matrix mechanism*, which itself is the starting point for the US Census Bureau's algorithm for the 2020 Decennial Census. That term and a particular instantiation of this approach originated in [LHR+10], the general approach has been studied in a number of concrete instantiations in many different papers [HT10, BDKT12, NTZ13, ENU20] and we'll focus on introducing a general class of mechanisms that captures a lot of these specific instantiations.

1 Query Release Recap and Histograms

We're going to start by recapping the problem of *linear query release* and then introduce a different viewpoint on the problem that will be useful for developing the framework. Recall that we start with a dataset $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{U}^n$ and we are given a set of *linear statistics* $f_1, \dots, f_k$ where each query is of the form

$$f_i(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^{n} \varphi_i(x_j) \text{ for some } \varphi_i : \mathcal{U} \to \{0,1\} \tag{1}$$

Before we go on, let's note a small differences between this formulation and what've said before: We now *normalize* the queries so that we compute the *average* of $\varphi_j(x_i) = 1$ rather than the *sum*. Because we have defined differential privacy only for datasets of size n, we can think of n as "public" and can freely convert between the two notations without any implications for privacy. Normalizing is largely just a cosmetic difference, but it will make some of the concepts in future lectures easier to understand, and it's important not to forget about it in order to map what we say in this lecture to previous lectures.

Recall that our baseline algorithm for query release is the Gaussian mechanism, where we add noise from a Gaussian distribution with noise scaled to the global ℓ_2-sensitivity: Given a set of queries $f_1, \dots, f_k$, define the function

$$F(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x})) \tag{2}$$

and define the Gaussian mechanism

$$\mathcal{M}(\mathbf{x}) = F(\mathbf{x}) + \mathbf{Z} \text{ where } \mathbf{Z} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{k\times k}) \tag{3}$$

where the noise variance σ^2 has the form $\sigma^2 = c^2_{\varepsilon,\delta}\Delta_2^2$ where

$$\Delta_2 = \max_{x\sim x' \in \mathcal{U}^n} \|F(\mathbf{x}) - F(\mathbf{x}')\|_2 \le \frac{2}{n} \cdot \max_{u\in\mathcal{U}} \|F(u)\|_2 \le \frac{2\sqrt{k}}{n} \tag{4}$$

is the global ℓ_2 sensitivity of the set of statistics, and $c^2_{\varepsilon,\delta} \approx \frac{\log(1/\delta)}{\varepsilon^2}$ is some scaling factor that depends only on the privacy parameters, which we will suppress because the dependence on the privacy parameters themselves is not particularly interesting for this lecture.

For this lecture, we're also going to make a small change in how we measure the error, which will again make things a little easier going forward. Instead of asking for the error to be small for every query, we'll ask for it to be small for the average query. Specifically, given answers $a = (a_1, \ldots, a_k)$, we want to have

$$\frac{1}{k^{1/2}}\|F(\mathbf{x}) - a\|_2 = \left(\frac{1}{k}\sum_{i=1}^{k}(f_i(\mathbf{x}) - a_i)^2\right)^{1/2} \leq \alpha \tag{5}$$

for as small an error α as possible. Note that dividing by $1/\sqrt{k}$ allows us to put the error in the same "units" as the maximum error over all queries. In other words, if the error for each query is at most α then (5) is also at most α.

For the Gaussian mechanism, we can say that the error will be

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|f(\mathbf{x}) - \mathcal{M}(\mathbf{x})\|_2\right) = O\left(c_{\varepsilon,\delta}\Delta_2\right) = O\left(\frac{c_{\varepsilon,\delta}k^{1/2}}{n}\right) \tag{6}$$

where we have used the fact that, for queries such as ours, Δ_2 is never larger than $2\sqrt{k}/n$.

This error rate of $\approx \sqrt{k}/n$ is what we think of as a *baseline* for any linear query release problem, and is what we're going to try to improve in this and the next several lectures.

Remark 1.1. Note that we are focusing on the Gaussian mechanism and (ε, δ)-differential privacy because it gives better accuracy, let's us work with the nicer Gaussian distribution, and avoid having two parallel explanations of the same mechanism. Everything we are saying can be done with the Laplace mechanism and $(\varepsilon, 0)$-differential privacy without any significant conceptual differences.

1.1 The Histogram Representation: Putting the Linear in Linear Queries

OK, now that we did all that somewhat painful recapping, let's do it all again in different notation! The reason we want to switch notation is that it will allow us to express linear queries in the language of—surprise, surprise—linear algebra.

To do so, we'll start by defining the *histogram representation* of a dataset. Given a dataset $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{U}^n$, we can represent the dataset as a *normalized histogram* that tells us how many times each element $u \in \mathcal{U}$ appears in the dataset. Specifically $\mathbf{h}_\mathbf{x} \in \mathbb{R}^{\mathcal{U}}$ and is defined as

$$(\mathbf{h}_\mathbf{x})_u = \frac{\#\left\{j : x_j = u\right\}}{n} \tag{7}$$

For example, if $\mathcal{U} = \{1, 2, 3\}$ and $\mathbf{x} = (1, 3, 3, 2, 3)$ then $\mathbf{h}_\mathbf{x} = (\frac{1}{5}, \frac{1}{5}, \frac{3}{5})$. Note that we can convert back and forth between the dataset and the histogram,[1] so they contain exactly the same information.

Remark 1.2. You may have noticed that the histogram can be *huge* even when the dataset is small. For example, if $n = 2000$ and $\mathcal{U} = \{0, 1\}^{30}$, then the dataset itself is just 60,000 bits (much less than a megabyte), but the histogram representation has over a billion buckets (more than a gigabyte)! Thus, for purposes of implementing algorithms in practice, we'd generally like to avoid switching to the histogram

[1] Strictly speaking, going from the histogram to the dataset loses information about the order of the items in the dataset, but since the queries are symmetric and don't depend on the order, we might as well think of the dataset as unordered anyway.

representation explicitlywhen implementing a mechanism, and think of it more as a tool for describing mechanisms. The computational issues involved in the mechanisms we're going to study are fascinating, but for now we're going to suppress them and focus only on the tradeoff between privacy and accuracy without thinking too hard about computation.

If we are going to design mechanisms in terms of the histogram, we need to understand what happens to the histogram when we change one element of the original dataset. In particular, this can change at most two buckets of the dataset by at most $1/n$ each. Thus we have that for every pair of neighboring datasets of size n

$$\|\mathbf{h}_{\mathbf{x}} - \mathbf{h}_{\mathbf{x}'}\|_1 \leq \frac{2}{n} \tag{8}$$

The reason we like to use the histogram representation is that we can finally make sense of the term *linear queries*—the reason they are called linear is becaue they are a linear function of the dataset's histogram. Specifically, given a single query f defined by a predicate $\varphi : \mathcal{U} \to \{0,1\}$, and $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, we can write

$$f(\mathbf{x}) = \mathbf{v} \cdot \mathbf{h}_{\mathbf{x}} \text{ where } \mathbf{v} = (\varphi(u_1), \ldots, \varphi(u_m)) \tag{9}$$

Then, given a set of queries $f_1, \ldots, f_k$, we can write

$$F(\mathbf{x}) = \mathbf{F}\mathbf{h}_{\mathbf{x}} \text{ where } A_{i,j} = \varphi_i(u_j). \tag{10}$$

Sometimes the hardest part of linear algebra is getting the dimensions right, so here we are writing $\mathbf{h}_{\mathbf{x}}$ as an $m \times 1$ column vector, and A is a $k \times m$ matrix. Thus, the answers are a $k \times 1$ column vector whose i-th row is the answer to the i-th query.

Exercise 1.3. Convince yourself that I didn't mess up the definitions (or tell me where I did).

Another thing to note is that, in this matrix representation, there is really nothig special about having functions φ output values in $\{0,1\}$, and in general we can think about queries that output real numbers in $\mathbb{R}$. Really the only advantage of having the queries output in $\{0,1\}$ is that we can get easy baselines for the sensitivity and have some natural notion of what "good" error bound is.

1.2 The Gaussian Mechanism in the Histogram Representation

In this notation we can write the Gaussian mechanism in a completely equivalent way as

$$\mathcal{M}_{\mathbf{F}}(\mathbf{x}) = \mathbf{F}\mathbf{h}_{\mathbf{x}} + \mathbf{Z} \text{ where } \mathbf{Z} \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{k \times k}) \tag{11}$$

Since we're now expressing the queries as a matrix $\mathbf{F}$, we'd like to be able to express their sensitivity Δ_2 in a language that makes sense for this linear algebraic viewpoint, so let's see how we can write the sensitivity of linear algebra as follows:

$$\Delta_2 = \max_{x \sim x' \in \mathcal{U}^n} \|F(\mathbf{x}) - F(\mathbf{x}')\|_2 \tag{12}$$

$$= \max_{x \sim x' \in \mathcal{U}^n} \|\mathbf{F}(\mathbf{h}_{\mathbf{x}} - \mathbf{h}_{\mathbf{x}'})\|_2 \tag{13}$$

$$\leq \max_{\substack{\mathbf{h},\mathbf{h}' \in \mathbb{R}^m \\ \|\mathbf{h}-\mathbf{h}'\|_1 \leq 2/n}} \|\mathbf{F}(\mathbf{h}_{\mathbf{x}} - \mathbf{h}_{\mathbf{x}'})\|_2 \tag{14}$$

$$= \frac{2}{n} \cdot \max_{\substack{\mathbf{v} \in \mathbb{R}^m \\ \|\mathbf{v}\|_1 \leq 1}} \|\mathbf{F}\mathbf{v}\|_2 \tag{15}$$

$$= \frac{2}{n} \cdot (\text{largest } \ell_2 \text{ norm of any column of } \mathbf{F}) = \frac{2}{n} \cdot \|\mathbf{F}\|_{1 \to 2} \tag{16}$$

where the notation $\|\mathbf{F}\|_{1\to 2}$ is just some fancy notation for the largest ℓ_2 norm of any column of $\mathbf{F}$.[2] To make sure nothing magical is going on here, you can check that $\max_{u\in\mathcal{U}} \|F(u)\|_2$ is just the largest ℓ_2-norm of any colum of $\mathbf{F}$, so the calculation of the sensitivity in (4) is no different from (16) but in our new notation. For keeping the notation reasonable, we're going to use this fancy $\frac{2}{n}\cdot\|\mathbf{F}\|_{1\to 2}$ notation to denote the sensitivity of the queries, but it's exactly the same as the ℓ_2 sensitivity of the queries, so just memorize that and don't worry exactly what this funny notation is supposed to mean.

Given the above, we can set the variance of the noise to be $\sigma^2 = c^2_{\varepsilon,\delta}\|\mathbf{F}\|^2_{1\to 2}$ and recover the same error guarantee we had before, but in different notation.

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{F}\mathbf{h}_{\mathbf{x}} - \mathcal{M}_{\mathbf{F}}(\mathbf{x})\|_2\right) = \mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{Z}\|_2\right) \le O\left(c_{\varepsilon,\delta}\frac{\|\mathbf{F}\|_{1\to 2}}{n}\right) \tag{17}$$

To appreciate what we've all just been through, we have literally just changed the notation on the Gaussian mechanism, so this is all a fancy way of explaining that we add noise to the queries proportional to their ℓ_2 sensitivity.

2 A Framework for Improving the Gaussian Mechanism

OK, we're finally ready to see some ideas for improving the basic Gaussian mechanism. We're going to see three conceptually distinct ways of reducing the error, which can be mixed and matched with each other. The first two can both be viewed as taking advantage of different types of *structure* in the queries, and the latter two is a form of *post-processing* of the Gaussian mechanism that can actually improve the error of the Gaussian mechanism dramatically in some cases.

2.1 Factorization

The first idea is called *factorization*, because we "factor" the mechanism into a differentially private *measurement* phase followed by a *reconstruction* phase that is just a postprocessing of what we measured. To get a feel for the idea, let's consider two special cases. First, suppose we are asked to answer the exact same query k times, so $f_1 = \cdots = f_k$. Then, in our histogram representation, the query matrix will be something like:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

where each row is a copy of the same query. Notice that, as long as the query is not the all-zero function, then the largest ℓ_2 norm of any column will be exactly $\sqrt{k}$, so the Gaussian mechanism will add noise proportional to $\sqrt{k}/n$. Clearly this is wasteful, because we're really only answering a single query with sensitivity $1/n$. In this example it's clear what to do—use the Gaussian mechanism to answer just the query f_1, obtain an answer a, and then output the vector of k answers $(a,\ldots,a)$. Since we're just answering a single query, the error is proportional to $1/n$, which is what it should be.

Although this idea is simple enough, the general principle is that we are *measuring* a single query f_1 and then performing a *linear reconstruction* of the answers to the remaining queries. The set of queries we measure is described by a matrix $\mathbf{M}\in\mathbb{R}^{\ell\times m}$ and the reconstruction is described by another matrix

[2] If you're curious, in general for $p,q>0$ we can define the $p\to q$ operator norm of a matrix to be $\|\mathbf{F}\|_{p\to q} = \max_{\|\mathbf{v}\|_p\le 1}\|\mathbf{F}\|_q$. The $2\to 2$ norm is the most common special case, and is equal to the largest singular value of $\mathbf{F}$.

$\mathbf{R} \in \mathbb{R}^{k \times \ell}$. In the simple example above, the query we measure is just f_1 and the reconstruction matrix maps a number to the vector containing k copies of that number. That is,

$$\mathbf{R} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

All we are actually using here to make this a viable approach is the fact that $\mathbf{RM} = \mathbf{F}$. As long as we have this, then our mechanism will have the form

$$\mathcal{M}_{\mathbf{R},\mathbf{M}}(\mathbf{x}) = \mathbf{R}(\mathbf{M}\mathbf{h}_{\mathbf{x}} + \mathbf{Z}) = \mathbf{R}\mathbf{M}\mathbf{h}_{\mathbf{x}} + \mathbf{R}\mathbf{Z} = \mathbf{F}\mathbf{h}_{\mathbf{x}} + \mathbf{R}\mathbf{Z} \tag{20}$$

In other words, we are just answering the queries $\mathbf{F}$ but with some alternative noise random variable $\mathbf{RZ}$ rather than $\mathbf{Z}$. Note that the Gaussian mechanism itself corresponds to the trivial factorization where $\mathbf{M} = \mathbf{F}$ and $\mathbf{R} = \mathbb{I}$, so in that sense factorization can only help and never hurt.

Exercise 2.1. Express the binary tree mechanism from Lecture 9 as an instance of factorization. Specifically, for the domain $\mathcal{U} = \{1, \ldots, 8\}$, and the set of threshold queries $f_t(\mathbf{x}) = \frac{1}{n} \cdot \#\left\{j : x_j \leq t\right\}$ for $t = 1, \ldots, 8$, write the matrix $\mathbf{F}$, the matrix $\mathbf{M}$ describing the set of queries in the binary tree, and the matrix $\mathbf{R}$ describing how to reconstruct the answers to the threshold queries.

For a given factorization $\mathbf{RM} = \mathbf{F}$, what is the error of the mechanism? To answer this question, we need to understand the quantity

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{F}\mathbf{h}_{\mathbf{x}} - \mathcal{M}_{\mathbf{R},\mathbf{M}}(\mathbf{x})\|_2\right) - \frac{1}{k^{1/2}}\mathbb{E}\left(\|\mathbf{R}\mathbf{Z}\|_2\right) \tag{21}$$

Recall that $\mathbf{Z}$ is from a Gaussian distribution where each coordinate is independent and has variance $\sigma^2 = O(c^2_{\varepsilon,\delta}\|\mathbf{M}\|^2_{1\to 2}/n)$, so the lower the sensitivity of $\mathbf{M}$ the smaller the magnitude of $\mathbf{Z}$ will be. However, multiplying by $\mathbf{R}$ can change the magnitude of the noise, and we need to understand what the noise is. To this end, let's write

$$\mathbf{R} = \begin{bmatrix} -\mathbf{r}_1- \\ \vdots \\ -\mathbf{r}_k- \end{bmatrix} \tag{22}$$

where each $\mathbf{r}_i$ is a vector. Observe that when we compute the answer to the i-th query, we are actually just computing $\mathbf{r}_i(\mathbf{M}\mathbf{h}_{\mathbf{x}} + \mathbf{Z})$, so the distribution of the noise for that query is just $(\mathbf{RZ})_i = \mathbf{r}_i \cdot \mathbf{Z}$. Now recall that if $\mathbf{Z} \sim \mathcal{N}(0, \sigma^2\mathbb{I})$, then $\mathbf{r}_i \cdot \mathbf{Z} \sim \mathcal{N}(0, \|\mathbf{r}_i\|_2^2\sigma^2)$ is a univariate Gaussian. Thus we have

$$\mathbb{E}\left((\mathbf{RZ})_i^2\right) = \|\mathbf{r}_i\|^2\sigma^2 \tag{23}$$

and we have

$$\mathbb{E}\left(\|\mathbf{RZ}\|_2\right) \le \mathbb{E}\left(\|\mathbf{RZ}\|_2^2\right)^{1/2} = \left(\mathbb{E}\left(\sum_{i=1}^{k}(\mathbf{RZ})_i^2\right)\right)^{1/2} \tag{24}$$

$$= \left(\sum_{i=1}^{k}\mathbb{E}\left((\mathbf{RZ})_i^2\right)\right)^{1/2} \tag{25}$$

$$= \left(\sum_{i=1}^{k}\|\mathbf{r}_i\|_2^2\sigma^2\right)^{1/2} \tag{26}$$

$$= \sigma\left(\sum_{i=1}^{k}\|\mathbf{r}_i\|_2^2\right)^{1/2} \tag{27}$$

$$= \sigma\left(\sum_{i=1}^{k}\sum_{j=1}^{\ell}\mathbf{R}_{i,j}^2\right)^{1/2} \tag{28}$$

Note that the double sum in the final quantity is the ℓ_2 norm of the matrix if we ignore the fact that it's a matrix and treat it as a big vector of length $k \times \ell$. This quantity has a name, and it's called the *Frobenius norm* of the matrix $\|\mathbf{R}\|_F^2 = \sum_i \sum_j \mathbf{R}_{i,j}$. Thus, putting everything back together, we can write

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{F}\mathbf{h}_\mathbf{x} - \mathcal{M}_{\text{fact}}(\mathbf{x})\|_2\right) - \frac{\mathbb{E}\left(\|\mathbf{RZ}\|_2\right)}{k^{1/2}} \le \frac{c_{\varepsilon,\delta}\|\mathbf{R}\|_F\|\mathbf{M}\|_{1\to 2}}{k^{1/2}n} \tag{29}$$

Since this might look like alphabet soup, let's go back to our simple example fo repeating the same query k times from (37). In this case the query matrix itself has $\|\mathbf{F}\|_{1\to 2} = k^{1/2}$, so the Gaussian mechanism has error proportional to $k^{1/2}/n$. After we do the factorization we get a new measurement matrix with $\|\mathbf{M}\|_{1\to 2} = 1$, and a reconstruction matrix with $\|\mathbf{R}\|_F = k^{1/2}$. Thus, when we plug this into (29) we get that the error is proportional to $1/n$.

Lastly, notice that there can be many different pairs $\mathbf{R}, \mathbf{M}$ such that $\mathbf{RM} = \mathbf{F}$, and of course we should chose the one that gives us the best error! Since we can describe the best error as being proportional to $\|\mathbf{R}\|_F\|\mathbf{M}\|_{1\to 2}/k^{1/2}$, we can just choose the factorization that minimizes this error! We can actually give a name to the quantity that represents the best error we can achieve via this framework—the *factorization norm* of $\mathbf{F}$.

Definition 2.2. Given a matrix $\mathbf{F} \in \mathbb{R}^{k\times m}$, the *factorization norm* is defined as

$$\gamma(\mathbf{F}) = \min\left\{\frac{\|\mathbf{R}\|_F\|\mathbf{M}\|_{1\to 2}}{k^{1/2}} : \mathbf{RM} = \mathbf{F}\right\} \tag{30}$$

Using this notation, and considering the factorization mechanism that uses the *optimal* factorization $\mathbf{RM} = \mathbf{F}$, we have the following theorem

Theorem 2.3. *For every set of linear queries given by a matrix $\mathbf{F} \in \mathbb{R}^{k\times m}$, there is an (ε, δ)-differentially private mechanism $\mathcal{M}$ that answers these queries with error*

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{F}\mathbf{h}_\mathbf{x} - \mathcal{M}(\mathbf{x})\|_2\right) = O\left(\frac{c_{\varepsilon,\delta}\cdot\gamma(\mathbf{F})}{n}\right) \tag{31}$$

2.2 Approximation

The next avenue for improvement is to give up on the constraint that $\mathbf{RM} = \mathbf{F}$. Suppose we run the factorization mechanism with an *approximate factorization* where $\mathbf{RM} = \mathbf{F} + \mathbf{E}$ for some "small" matrix $\mathbf{E}$? In this more general setting we have

$$\mathcal{M}_{\mathbf{R},\mathbf{M}}(\mathbf{x}) = \mathbf{R}(\mathbf{M}\mathbf{h}_{\mathbf{x}} + \mathbf{Z}) = \mathbf{F}\mathbf{h}_{\mathbf{x}} + \mathbf{E}\mathbf{h}_{\mathbf{x}} + \mathbf{R}\mathbf{Z} \tag{32}$$

So now we have two sources of error: the noise vector $\mathbf{RZ}$ and the error due to approximation $\mathbf{E}\mathbf{h}_{\mathbf{x}}$. Thus, the overall error can now be bounded as follows

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{E}\mathbf{h}_{\mathbf{x}} + \mathbf{R}\mathbf{Z}\|_2\right) \leq \frac{\|\mathbf{E}\mathbf{h}_{\mathbf{x}}\|_2}{k^{1/2}} + \frac{\mathbb{E}\left(\|\mathbf{R}\mathbf{Z}\|_2\right)}{k^{1/2}} \tag{33}$$

$$\leq \frac{\|\mathbf{E}\mathbf{h}_{\mathbf{x}}\|_2}{k^{1/2}} + \frac{c_{\varepsilon,\delta}\|\mathbf{R}\|_F\|\mathbf{M}\|_{1\to 2}}{k^{1/2}n} \tag{34}$$

$$\leq \frac{\|\mathbf{E}\|_{1\to 2}}{k^{1/2}} + \frac{c_{\varepsilon,\delta}\|\mathbf{R}\|_F\|\mathbf{M}\|_{1\to 2}}{k^{1/2}n} \tag{35}$$

where the first inequality is the triangle inequality and linearity of expectation, the second inequality is our analysis of the noise added by then factorization mechanism, and the final inequality is our definition of $\|\mathbf{E}\|_{1\to 2}$ combined with the fact that $\|\mathbf{h}_{\mathbf{x}}\|_1 = 1$. To interpret this analysis, note that if every entry of $\mathbf{E}$ is bounded by α in absolute value, then the additional error due to approximation is at most α.

One (somewhat boring) example where this kind of approximation is useful is when the queries are *approximately* the same but not identical. This can naturally arise if we allow the queries to take values that are arbitrary real numbers instead of values in $\{0, 1\}$. For example, if the query matrix is

$$\mathbf{F} = \begin{bmatrix} 1\pm\alpha & 1\pm\alpha & 1\pm\alpha & 1\pm\alpha & 1\pm\alpha \\ 1\pm\alpha & 1\pm\alpha & 1\pm\alpha & 1\pm\alpha & 1\pm\alpha \\ 1\pm\alpha & 1\pm\alpha & 1\pm\alpha & 1\pm\alpha & 1\pm\alpha \end{bmatrix} \tag{36}$$

then we can use the approximate factorization

$$\mathbf{R} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

to approximate this set of two queries as if it were a single query.

Remark 2.4. There are far more powerful examples of the benefits of approximation, although explaining how they work is beyond the scope of this lecture. But the punchline is that there are examples of sets of 2^d queries where exact factorizaton cannot get error $2^{o(d)}/n$, but approximate factorization can get error $.001 + 2^{O(\sqrt{d})}/n$, which is an exponential improvement [TUV12].

As above, we can choose the factorization $\mathbf{R}, \mathbf{M}$ to optimize this error term, trading off the two sources of error.

Definition 2.5. Given a matrix $\mathbf{F} \in \mathbb{R}^{k\times m}$, the *$\alpha$-approximate factorization norm* is defined as

$$\gamma_\alpha(\mathbf{F}) = \min\left\{\frac{\|\mathbf{R}\|_F\|\mathbf{M}\|_{1\to 2}}{k^{1/2}} : \|\mathbf{RM} - \mathbf{F}\|_{1\to 2} \leq \alpha k^{1/2}\right\} \tag{38}$$

Using this notation, and considering the factorization mechanism that uses the *optimal* factorization $\mathbf{RM} = \mathbf{F} + \mathbf{E}$ subject to the constraint $\|\mathbf{E}\|_{1\to 2} \le \alpha k^{1/2}$, we have the following theorem

Theorem 2.6. *For every set of linear queries given by a matrix $\mathbf{F} \in \mathbb{R}^{k\times m}$, and every $\alpha \ge 0$, there is an (ε, δ)-differentially private mechanism $\mathcal{M}$ that answers these queries with error*

$$\mathbb{E}\left(\frac{1}{k^{1/2}}\|\mathbf{F}\mathbf{h}_{\mathbf{x}} - \mathcal{M}(\mathbf{x})\|_2\right) = O\left(\alpha + \frac{c_{\varepsilon,\delta}\cdot\gamma_\alpha(\mathbf{F})}{n}\right) \tag{39}$$

2.3 Projection

Now we're going to look at a very different approach to improve the error of the mechanism, albeit one that can be combined seamlessly with the factorization approaches we just described. The approach can be described most easily wtih an example: Suppose the data domain is $\mathcal{U} = \{1, 2, 3\}$ and we want to answer the two non-trivial *threhsold queries* on a dataset in $\mathcal{U}^n$. Written in matrix notation, this gives us the linear queries.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{40}$$

Notice that by construction, if a_1^* and a_2^* are the true answers to the first and second queries respectively, then we must have $0 \le a_1^* \le a_2^* \le 1$. However, when we add Gaussian noise to the answers, we will obtain approximate answers a_1 and a_2 that may not satisfy these equations! In other words, we might obtain answers that are close to the true answers, but are not *consistent* in the sense that they are not the exact answers we would obtain from any dataset.

We might not, in principle, care about having consistent answers,[3] enforcing that the answers be consistent with the exact answers on some dataset can actually improve the error—in some cases quite dramatically (we'll see how this can happen in the next lecture)!

For a simple, but again somewhat boring example, for any count query the true answer should lie in the interval $[0, 1]$. If we let a^* be the true answer and a be our noisy answer, then the error is $|a^* - a|$. However, what if $a < 0$, meaning a cannot possibly by the correct answer? We'll, clearly we can only give a more accurate answer if we replace a with $\tilde{a} = \max\{a, 0\}$, and you can check that $|a^* - \tilde{a}|$ is never larger than $|a^* - \tilde{a}|$ and can be smaller. Similarly, we can also let $\tilde{a} = \max\{\min\{a, 1\}, 0\}$ to capture the constrait that $0 \le a^* \le 1$.

I will put a picture here soon. For now, see the lecture video for a visualization.

While this example might seem a little trivial, it's a simple instance of a more general phenomenon. Knowing only about the *definition* of the queries (not the data itself) we have a set of consistent answers C that represents all possible correct answers. Specifically

$$C = \left\{\mathbf{a} \in \mathbb{R}^k : \exists\, \mathbf{h} \in \mathbb{R}^m \text{ s.t. } \|\mathbf{h}\|_1 = 1 \text{ and } \mathbf{a} = \mathbf{F}\mathbf{h}\right\} \tag{41}$$

Remark 2.7. If you're astute, you'll notice that the way we've defined C, it is not exactly the set of all answers that can arise as the exact answers to some dataset. The reason is that not every histogram represents a dataset of size n. For example, if the query is a count, and $n = 2$, then the only possible answers are $\{0, \frac{1}{2}, 1\}$, whereas $C = [0, 1]$. It's more precise to say that C is the convex hull of the set

[3] One reason that you might actually want consistent answers is to make the noisy answers easier to use and interpret in further applications. For example if you have a program that expects to be given as input the number of users born before 1980, that program might crash or give nonsensical answers if you tell it the answer is -1.82.

of answers that are consistent with some dataset. Note, however, that every set of answers that are consistent with a dataset of size n lies in C, and every set of answers in C is consistent with those given by some dataset of arbitrarily large size. Making C a convex set is useful for a number of reasons.

Given that we know the true answers that we're looking for $\mathbf{a}^* = \mathbf{F}\mathbf{h}_{\mathbf{x}}$ are in the set C, we can improve the error by *projecting* into C. That is, we define the projection operator

$$\Pi_C(\mathbf{a}) = \arg\min_{\mathbf{a}' \in C} \|\mathbf{a} - \mathbf{a}'\|_2 \tag{42}$$

and replace the answers $\mathbf{a}$ with $\tilde{\mathbf{a}} = \Pi_C(\mathbf{a})$. There are two really important facts about the projected answers $\tilde{\mathbf{a}}$:

- The error of $\tilde{\mathbf{a}}$ is never larger than $\mathbf{a}$, because the true answers $\mathbf{a}^*$ are in C and C is a convex set. That is
$$\|\tilde{\mathbf{a}} - \mathbf{a}^*\|_2 \leq \|\mathbf{a} - \mathbf{a}^*\|_2 \tag{43}$$
Notice that this inequality holds for *every* vector $\mathbf{a}$. So, if $\mathbf{a}$ is a random variable then projection can't increase the expected ℓ_2 error.
- If $\mathbf{a}$ was obtained with (ε, δ)-differential privacy, then $\tilde{\mathbf{a}} = \Pi_C(\mathbf{a})$ is just a post-processing of $\mathbf{a}$ that does not depend on the original dataset in any way, so the combined algorithm that returns $\tilde{\mathbf{a}}$ is also (ε, δ)-differentially private.

In other words, if what we care about is minimizing ℓ_2 error, then projection is a *cost free* approach for improving the answers. It can never make the error larger and does not increase the privacy parameters in any way.

2.4 Putting in Together

Additional Reading and Watching

- A very impressive line of work has shown that exact factorization mechanisms give approximately optimal error *for any set of linear queries* in the parameter regime where n is quite large [HT10, BDKT12, NTZ13]. The simplest proof of this fact appears in [ENU20], which in particular shows that exact factorization mechanisms are optimal among all mechanisms that add the same noise distribution independently of the dataset $\mathbf{x}$.
- The idea of using projection to improve the error was first described explicitly in [BCD+07] and [HRMS10]. It was first shown to give dramatically improved error for any family of linear queries in the regime where n is somewhat small by [NTZ13].

References

[BCD+07] 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 26th Annual ACM Symposium on Principles of Database Systems*, PODS '07, 2007. ACM.

[BDKT12] Aditya Bhaskara, Daniel Dadush, Ravishankar Krishnaswamy, and Kunal Talwar. Unconditional differentially private mechanisms for linear queries. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, STOC '12, 2012.

[ENU20] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. The power of factorization meisms in local and central differential privacy. In *ACM Symposium on the Theory of Computing*, STOC '20, 2020. https://arxiv.org/abs/1911.08339.

[HRMS10] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment*, 3(1), 2010. https://arxiv.org/abs/0904.0942.

[HT10] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*, 2010.

[LHR+10] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2010.

[NTZ13] Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: The small database and approximate cases. In *ACM Symposium on Theory of Computing*, STOC '13, 2013.

[TUV12] Justin Thaler, Jonathan Ullman, and Salil P. Vadhan. Faster algorithms for privately releasing marginals. In *39th International Colloquium on Automata, Languages, and Programming -*, ICALP '12, 2012. Springer.