4

Releasing Linear Queries with Correlated Error

One of the most fundamental primitives in private data analysis is the ability to answer numeric valued queries on a dataset. In the last section, we began to see tools that would allow us to do this by adding independently drawn noise to the query answers. In this section, we continue this study, and see that by instead adding carefully correlated noise, we can gain the ability to privately answer vastly more queries to high accuracy. Here, we see two specific mechanisms for solving this problem, which we will generalize in the next section.

In this section, we consider algorithms for solving the *query release* problem with better accuracy than we would get by simply using compositions of the Laplace mechanism. The improvements are possible because the set of queries is handled as a whole — even in the online setting! — permitting the noise on individual queries to be correlated. To immediately see that something along these lines might be possible, consider the pair of queries in the differencing attack described in Section 1: "How many people in the database have the sickle cell trait?" and "How many people, not named X, in the database have the sickle cell trait?" Suppose a mechanism answers the first question using the Laplace mechanism and then, when the second question is posed,

responds "You already know the approximate answer, because you just asked me almost the exact same question." This coordinated response to the pair of questions incurs no more privacy loss than either question would do taken in isolation, so a (small) privacy savings has been achieved.

The query release problem is quite natural: given a class of queries $\mathcal{Q}$ over the database, we wish to release some answer a_i for each query $f_i \in \mathcal{Q}$ such that the error $\max_i |a_i - f_i(x)|$ is as low as possible, while still preserving differential privacy.[1] Recall that for any family of low sensitivity queries, we can apply the Laplace mechanism, which adds fresh, independent, noise to the answer to each query. Unfortunately, at a fixed privacy level, for $(\epsilon, 0)$-privacy guarantees, the magnitude of the noise that we must add with the Laplace mechanism scales with $|\mathcal{Q}|$ because this is the rate at which the sensitivity of the combined queries may grow. Similarly, for (ϵ, δ)-privacy guarantees, the noise scales with $\sqrt{|\mathcal{Q}| \ln(1/\delta)}$. For example, suppose that our class of queries $\mathcal{Q}$ consists only of many copies of the same query: $f_i = f^*$ for all i. If we use the Laplace mechanism to release the answers, it will add independent noise, and so each a_i will be an independent random variable with mean $f^*(x)$. Clearly, in this regime, the noise rate must grow with $|\mathcal{Q}|$ since otherwise the average of the a_i will converge to the true value $f^*(x)$, which would be a privacy violation. However, in this case, because $f_i = f^*$ for all i, it would make more sense to approximate f^* only once with $a^* \approx f^*(x)$ and release $a_i = a^*$ for all i. In this case, the noise rate would not have to scale with $|\mathcal{Q}|$ at all. In this section, we aim to design algorithms that are much more accurate than the Laplace mechanism (with error that scales with $\log |\mathcal{Q}|$) by adding non-independent noise as a function of the set of queries.

Recall that our universe is $\mathcal{X} = \{\chi_1, \chi_2, \ldots, \chi_{|\mathcal{X}|}\}$ and that databases are represented by histograms in $\mathbb{N}^{|\mathcal{X}|}$. A *linear query* is simply a counting query, but generalized to take values in the interval $[0, 1]$ rather than only boolean values. Specifically, a linear query f takes the

[1] It is the privacy constraint that makes the problem interesting. Without this constraint, the query release problem is trivially and optimally solved by just outputting exact answers for every query.

form $f : \mathcal{X} \to [0,1]$, and applied to a database x returns either the *sum* or *average* value of the query on the database (we will think of both, depending on which is more convenient for the analysis). When we think of linear queries as returning *average* values, we will refer to them as *normalized* linear queries, and say that they take value:

$$f(x) = \frac{1}{\|x\|_1} \sum_{i=1}^{|\mathcal{X}|} x_i \cdot f(\chi_i).$$

When we think of linear queries as returning *sum* values we will refer to them as *un-normalized* linear queries, and say that they take value:

$$f(x) = \sum_{i=1}^{|\mathcal{X}|} x_i \cdot f(\chi_i).$$

Whenever we state a bound, it should be clear from context whether we are speaking of normalized or un-normalized queries, because they take values in very different ranges. Note that normalized linear queries take values in $[0,1]$, whereas un-normalized queries take values in $[0, \|x\|_1]$.

Note that with this definition linear queries have sensitivity $\Delta f \leq 1$. Later sections will discuss arbitrary low-sensitivity queries.

We will present two techniques, one each for the offline and online cases. Surprisingly, and wonderfully, the offline technique is an immediate application of the exponential mechanism using well-known sampling bounds from learning theory! The algorithm will simply be to apply the exponential mechanism with range equal to the set of all *small* databases y and quality function $u(x,y)$ equal to minus the maximum approximation error incurred by querying y to obtain an approximation for $f(x)$:

$$u(x,y) = -\max_{f \in \mathcal{Q}} |f(x) - f(y)|. \tag{4.1}$$

Sampling bounds (see Lemma 4.3 below) tell us that a random subset of $\ln|\mathcal{Q}|/\alpha^2$ elements of x will very likely give us a good approximation for all $f(x)$ (specifically, with additive error bounded by α), so we know it is sufficient to restrict the set of possible outputs to small databases. We don't actually care that the potential output databases are small, only that they are not too numerous: their number plays a role in the proof of

utility, which is an immediate application of the utility theorem for the exponential mechanism (Theorem 3.11). More specifically, if the total number of potential outputs is not too numerous then, in particular, the total number of low-utility outputs is not too numerous, and therefore the ratio of bad outputs to good outputs (there is at least one) is not too large.

The online mechanism, which, despite not knowing the entire set of queries in advance, will achieve the same accuracy as the offline mechanism, and will be a direct application of the sparse vector technique. As a result, privacy will be immediate, but utility will require a proof. The key will be to argue that, even for a very large set of counting queries, few queries are "significant"; that is, significant queries will be sparse. As with the sparse vector algorithms, we can scale noise according to the number of significant queries, with little dependence on the total number of queries.

Before we go on and present the mechanisms, we will give just one example of a useful class of linear queries.

Example 4.1. Suppose that elements of the database are represented by d *boolean* features. For example, the first feature may represent whether or not the individual is male or female, the second feature may represent whether or not they are a college graduate, the third feature may represent whether or not they are US citizens, etc. That is, our data universe is $\mathcal{X} = \{0,1\}^d$. Given a subset of these attributes $S \subseteq \{1,\ldots,d\}$, we might like to know how many people in the dataset have these attributes. (e.g., "What fraction of the dataset consists of male college graduates with a family history of lung cancer?"). This naturally defines a query called a *monotone conjunction query*, parameterized by a subset of attributes S and defined as $f_S(z) = \prod_{i \in S} z_i$, for $z \in \mathcal{X}$. The class of *all* such queries is simply $\mathcal{Q} = \{f_S : S \subseteq \{1,\ldots,d\}\}$, and has size $|\mathcal{Q}| = 2^d$. A collection of answers to conjunctions is sometimes called a *contingency* or *marginal* table, and is a common method of releasing statistical information about a dataset. Often times, we may not be interested in the answers to *all* conjunctions, but rather just those that ask about subsets of features S of size $|S| = k$ for some fixed k. This class of queries $\mathcal{Q}_k = \{f_S : S \subseteq \{1,\ldots,d\}, |S| = k\}$ has size $\binom{d}{k}$.

This large and useful class of queries is just one example of the sorts of queries that can be accurately answered by the algorithms given in this section. (Note that if we wish to also allow (non-monotone) conjunctions which ask about *negated* attributes, we can do that as well — simply double the feature space from d to $2d$, and set $z_{d+i} = 1 - z_i$ for all $i \in \{1, \ldots, d\}$.)

4.1 An offline algorithm: SmallDB

In this section, we give an algorithm based on the idea of sampling a small database using the exponential mechanism. What we will show is that, for counting queries, it suffices to consider databases that are small: their size will only be a function of the query class, and our desired approximation accuracy α, and crucially *not* on $\|x\|_1$, the size of the private database. This is important because it will allow us to simultaneously guarantee, for all sufficiently large databases, that there is at least *one* database in the range of the exponential mechanism that well approximates x on queries in $\mathcal{Q}$, and that there are not *too many* databases in the range to dissipate the probability mass placed on this "good" database.

Algorithm 4 The Small Database Mechanism

SmallDB$(x, \mathcal{Q}, \varepsilon, \alpha)$

Let $\mathcal{R} \leftarrow \{y \in \mathbb{N}^{|\mathcal{X}|} : \|y\|_1 = \frac{\log|\mathcal{Q}|}{\alpha^2}\}$

Let $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$ be defined to be:

$$u(x, y) = -\max_{f \in \mathcal{Q}} |f(x) - f(y)|$$

Sample And Output $y \in \mathcal{R}$ with the exponential mechanism $\mathcal{M}_E(x, u, \mathcal{R})$

We first observe that the Small Database mechanism preserves ε-differential privacy.

Proposition 4.1. The Small Database mechanism is $(\varepsilon, 0)$ differentially private.

Proof. The Small Database mechanism is simply an instantiation of the exponential mechanism. Therefore, privacy follows from Theorem 3.10. □

We may similarly call on our analysis of the exponential mechanism to understand the utility guarantees of the Small Database mechanism. But first, we must justify our choice of range $\mathcal{R} = \{y \in \mathbb{N}^{|\mathcal{X}|} : \|y\|_1 = \frac{\log|\mathcal{Q}|}{\alpha^2}\}$, the set of all databases of size $\log|\mathcal{Q}|/\alpha^2$.

Theorem 4.2. For any finite class of linear queries $\mathcal{Q}$, if $\mathcal{R} = \{y \in \mathbb{N}^{|\mathcal{X}|} : \|y\|_1 = \frac{\log|\mathcal{Q}|}{\alpha^2}\}$ then for all $x \in \mathbb{N}^{|\mathcal{X}|}$, there exists a $y \in \mathcal{R}$ such that:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$$

In other words, we will show that for any collection of linear queries $\mathcal{Q}$ and for any database x, there is a "small" database y of size $\|y\|_1 = \frac{\log|\mathcal{Q}|}{\alpha^2}$ that approximately encodes the answers to every query in $\mathcal{Q}$, up to error α.

Lemma 4.3 (Sampling Bounds)**.** For any $x \in \mathbb{N}^{|\mathcal{X}|}$ and for any collection of linear queries $\mathcal{Q}$, there exists a database y of size

$$\|y\|_1 = \frac{\log|\mathcal{Q}|}{\alpha^2}$$

such that:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$$

Proof. Let $m = \frac{\log|\mathcal{Q}|}{\alpha^2}$. We will construct a database y by taking m uniformly random samples from the elements of x. Specifically, for $i \in \{1, \ldots, m\}$, let X_i be a random variable taking value $\chi_j \in \mathcal{X}$ with probability $x_j/\|x\|_1$, and let y be the database containing elements $X_1, \ldots, X_m$. Now fix any $f \in \mathcal{Q}$ and consider the quantity $f(y)$. We have:

$$f(y) = \frac{1}{\|y\|_1} \sum_{i=1}^{|\mathcal{X}|} y_i \cdot f(\chi_i) = \frac{1}{m} \sum_{i=1}^{m} f(X_i).$$

We note that each term $f(X_i)$ of the sum is a bounded random variable taking values $0 \leq f(X_i) \leq 1$ with expectation

$$\mathbb{E}[f(X_i)] = \sum_{j=1}^{|\mathcal{X}|} \frac{x_j}{\|x\|_1} f(\chi_j) = f(x)\,,$$

and that the expectation of $f(y)$ is:

$$\mathbb{E}[f(y)] = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}[f(X_i)] = f(x)\,.$$

Therefore, we can apply the Chernoff bound stated in Theorem 3.1 which gives:

$$\Pr\left[|f(y) - f(x)| > \alpha\right] \leq 2e^{-2m\alpha^2}\,.$$

Taking a union bound over all of the linear queries $f \in \mathcal{Q}$, we get:

$$\Pr\left[\max_{f \in \mathcal{Q}} |f(y) - f(x)| > \alpha\right] \leq 2|\mathcal{Q}|e^{-2m\alpha^2}\,.$$

Plugging in $m = \frac{\log|\mathcal{Q}|}{\alpha^2}$ makes the right hand side smaller than 1 (so long as $|\mathcal{Q}| > 2$), proving that there exists a database of size m satisfying the stated bound, which completes the proof of the lemma. □

The proof of Theorem 4.2 simply follows from the observation that $\mathcal{R}$ contains *all* databases of size $\frac{\log|\mathcal{Q}|}{\alpha^2}$.

Proposition 4.4. Let $\mathcal{Q}$ be any class of linear queries. Let y be the database output by $\mathrm{SmallDB}(x, \mathcal{Q}, \varepsilon, \alpha)$. Then with probability $1 - \beta$:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha + \frac{2\left(\frac{\log|\mathcal{X}|\log|\mathcal{Q}|}{\alpha^2} + \log\left(\frac{1}{\beta}\right)\right)}{\varepsilon\|x\|_1}\,.$$

Proof. Applying the utility bounds for the exponential mechanism (Theorem 3.11) with $\Delta u = \frac{1}{\|x\|_1}$ and $\mathrm{OPT}_q(D) \leq \alpha$ (which follows from Theorem 4.2), we find:

$$\Pr\left[\max_{f \in \mathcal{Q}} |f(x) - f(y)| \geq \alpha + \frac{2}{\varepsilon\|x\|_1}\left(\log\left(|\mathcal{R}|\right) + t\right)\right] \leq e^{-t}\,.$$

We complete the proof by (1) noting that $\mathcal{R}$, which is the set of all databases of size at most $\log|\mathcal{Q}|/\alpha^2$, satisfies $|\mathcal{R}| \leq |\mathcal{X}|^{\log|\mathcal{Q}|/\alpha^2}$ and (2) by setting $t = \log\left(\frac{1}{\beta}\right)$. □

Finally, we may now state the utility theorem for SmallDB.

Theorem 4.5. By the appropriate choice of α, letting y be the database output by SmallDB$(x, \mathcal{Q}, \varepsilon, \frac{\alpha}{2})$, we can ensure that with probability $1 - \beta$:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \left(\frac{16 \log |\mathcal{X}| \log |\mathcal{Q}| + 4 \log\left(\frac{1}{\beta}\right)}{\varepsilon \|x\|_1} \right)^{1/3}. \tag{4.2}$$

Equivalently, for any database x with

$$\|x\|_1 \geq \frac{16 \log |\mathcal{X}| \log |\mathcal{Q}| + 4 \log\left(\frac{1}{\beta}\right)}{\varepsilon \alpha^3} \tag{4.3}$$

with probability $1 - \beta$: $\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$.

Proof. By Theorem 4.2, we get:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \frac{\alpha}{2} + \frac{2\left(\frac{4 \log |\mathcal{X}| \log |\mathcal{Q}|}{\alpha^2} + \log\left(\frac{1}{\beta}\right)\right)}{\varepsilon \|x\|_1}.$$

Setting this quantity to be at most α and solving for $\|x\|_1$ yields (4.3). Solving for α yields (4.4). □

Note that this theorem states that for fixed α and ε, even with $\delta = 0$, it is possible to answer almost *exponentially* many queries in the size of the database.[2] This is in contrast to the Laplace mechanism, when we use it directly to answer linear queries, which can only answer *linearly* many.

Note also that in this discussion, it has been most convenient to think about normalized queries. However, we can get the corresponding bounds for unnormalized queries simply by multiplying by $\|x\|_1$:

Theorem 4.6 (Accuracy theorem for un-normalized queries)**.** By the appropriate choice of α, letting y be the database output by

[2] Specifically, solving for k we find that the mechanism can answer k queries for:

$$k \leq \exp\left(O\left(\frac{\alpha^3 \epsilon \|x\|_1}{\log |\mathcal{X}|}\right)\right).$$

SmallDB$(x, \mathcal{Q}, \varepsilon, \frac{\alpha}{2})$, we can ensure that with probability $1 - \beta$:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \|x\|_1^{2/3} \left(\frac{16 \log |\mathcal{X}| \log |\mathcal{Q}| + 4 \log\left(\frac{1}{\beta}\right)}{\varepsilon} \right)^{1/3}. \quad (4.4)$$

More Refined Bounds. We proved that *every* set of linear queries $\mathcal{Q}$ has a collection of databases of size at most $|\mathcal{X}|^{\log |\mathcal{Q}|/\alpha^2}$ that well-approximates every database x with respect to $\mathcal{Q}$ with error at most α. This is often an over-estimate however, since it completely ignores the structure of the queries. For example, if $\mathcal{Q}$ simply contains the same query repeated over and over again, each time in a different guise, then there is no reason that the size of the range of the exponential mechanism should grow with $|\mathcal{Q}|$. Similarly, there may even be classes of queries $\mathcal{Q}$ that have *infinite* cardinality, but nevertheless are well approximated by small databases. For example, queries that correspond to asking whether a point lies within a given interval on the real line form an infinitely large class $\mathcal{Q}$, since there are uncountably many intervals on the real line. Nevertheless, this class of queries exhibits very simple structure that causes it to be well approximated by small databases. By considering more refined structure of our query classes, we will be able to give bounds for differentially private mechanisms which improve over the simple sampling bounds (Lemma 4.3) and can be non-trivial even for doubly exponentially large classes of queries.[3] We will not fully develop these bounds here, but will instead state several results for the simpler class of *counting queries*. Recall that a counting query $f : \mathcal{X} \to \{0, 1\}$ maps database points to boolean values, rather than any value in the interval $[0, 1]$ as linear queries do.

Definition 4.1 (Shattering)**.** A class of counting queries $\mathcal{Q}$ *shatters* a collection of points $S \subseteq \mathcal{X}$ if for every $T \subseteq S$, there exists an $f \in \mathcal{Q}$ such that $\{x \in S : f(x) = 1\} = T$. That is, $\mathcal{Q}$ shatters S if for every one of the $2^{|S|}$ subsets T of S, there is some function in $\mathcal{Q}$ that labels exactly

[3] In fact, our complexity measure for a class of queries can be finite even for *infinite* classes of queries, but here we are dealing with queries over a finite universe, so there do not exist infinitely many distinct queries.

those elements as positive, and does not label any of the elements in $S \setminus T$ as positive.

Note that for $\mathcal{Q}$ to shatter S it must be the case that $|\mathcal{Q}| \geq 2^{|S|}$ since $\mathcal{Q}$ must contain a function f for each subset $T \subseteq S$. We can now define our complexity measure for counting queries.

Definition 4.2 (Vapnik–Chervonenkis (VC) Dimension)**.** A collection of counting queries $\mathcal{Q}$ has VC-dimension d if there exists some set $S \subseteq \mathcal{X}$ of cardinality $|S| = d$ such that $\mathcal{Q}$ shatters S, and $\mathcal{Q}$ does not shatter any set of cardinality $d+1$. We can denote this quantity by VC-DIM($\mathcal{Q}$).

Consider again the class of 1-dimensional intervals on the range $[0, \infty]$ defined over the domain $\mathcal{X} = \mathbb{R}$. The function $f_{a,b}$ corresponding to the interval $[a, b]$ is defined such that $f_{a,b}(x) = 1$ if and only if $x \in [a, b]$. This is an infinite class of queries, but its VC-dimension is 2. For any pair of distinct points $x < y$, there is an interval that contains neither point ($a, b < x$), an interval that contains both points ($a < x < y < b$), and an interval that contains each of the points but not the other ($a < x < b < y$ and $x < a < y < b$). However, for any 3 distinct points $x < y < z$, there is no interval $[a, b]$ such that $f_{a,b}[x] = f_{a,b}[z] = 1$ but $f_{a,b}[y] = 0$.

We observe that the VC-dimension of a finite concept class can never be too large.

Lemma 4.7. For any finite class $\mathcal{Q}$, VC-DIM($\mathcal{Q}$) $\leq \log |\mathcal{Q}|$.

Proof. If VC-DIM($\mathcal{Q}$) $= d$ then $\mathcal{Q}$ shatters some set of items $S \subseteq \mathcal{X}$ of cardinality $|S| = d$. But by the definition of shattering, since S has 2^d distinct subsets, $\mathcal{Q}$ must have at least 2^d distinct functions in it. □

It will turn out that we can essentially replace the term $\log |\mathcal{Q}|$ with the term VC-DIM($\mathcal{Q}$) in our bounds for the SmallDB mechanism. By the previous lemma, this is can only be an improvement for finite classes $\mathcal{Q}$.

Theorem 4.8. For any finite class of linear queries $\mathcal{Q}$, if $\mathcal{R} = \{y \in \mathbb{N}^{|\mathcal{X}|} : \|y\| \in O\left(\frac{\text{VC-DIM}(\mathcal{Q})}{\alpha^2}\right)\}$ then for all $x \in \mathbb{N}^{|\mathcal{X}|}$, there exists a $y \in \mathcal{R}$

such that:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$$

As a result of this theorem, we get the analogue of Theorem 4.5 with VC-dimension as our measure of query class complexity:

Theorem 4.9. Let y be the database output by SmallDB$(x, \mathcal{Q}, \varepsilon, \frac{\alpha}{2})$. Then with probability $1 - \beta$:

$$\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq O\left(\left(\frac{\log|\mathcal{X}|\text{VC-DIM}(\mathcal{Q}) + \log\left(\frac{1}{\beta}\right)}{\varepsilon\|x\|_1}\right)^{1/3}\right)$$

Equivalently, for any database x with

$$\|x\|_1 \geq O\left(\frac{\log|\mathcal{X}|\text{VC-DIM}(\mathcal{Q}) + \log\left(\frac{1}{\beta}\right)}{\varepsilon\alpha^3}\right)$$

with probability $1 - \beta$: $\max_{f \in \mathcal{Q}} |f(x) - f(y)| \leq \alpha$.

An analogous (although more cumbersome) measure of query complexity, the "Fat Shattering Dimension," defines the complexity of a class of linear queries, as opposed to simply counting queries. The Fat Shattering Dimension controls the size of the smallest "α-net" (Definition 5.2 in Section 5) for a class of linear queries $\mathcal{Q}$ as VC-dimension does for counting queries. This measure can similarly be used to give more refined bounds for mechanisms designed to privately release linear queries.

4.2 An online mechanism: private multiplicative weights

We will now give a mechanism for answering queries that arrive online and may be interactively chosen. The algorithm will be a simple combination of the sparse vector algorithm (which can answer threshold queries adaptively), and the exponentiated gradient descent algorithm for learning linear predictors online.

This latter algorithm is also known as Hedge or more generally the multiplicative weights technique. The idea is the following: When we

view the database $D \in \mathbb{N}^{|\mathcal{X}|}$ as a histogram and are interested only in linear queries (i.e., linear functions of this histogram), then we can view the problem of answering linear queries as the problem of learning the linear function D that defines the query answers $\langle D, q \rangle$, given a query $q \in [0,1]^{|\mathcal{X}|}$. If the learning algorithm only needs to access the data using privacy-preserving queries, then rather than having a privacy cost that grows with the number of queries we would like to answer, we can have a privacy cost that grows only with the number of queries the learning algorithm needs to make. The "multiplicative weights" algorithm which we present next is a classical example of such a learning algorithm: it can learn any linear predictor by making only a small number of queries. It maintains at all times a current "hypothesis predictor," and accesses the data only by requiring examples of queries on which its hypothesis predictor differs from the (true) private database by a large amount. Its guarantee is that it will always learn the target linear function up to small error, given only a small number of such examples. How can we find these examples? The sparse vector algorithm that we saw in the previous section allows us to do this on the fly, while paying for only those examples that have high error on the current multiplicative weights hypothesis. As queries come in, we ask whether the true answer to the query differs substantially from the answer to the query on the current multiplicative weights hypothesis. Note that this is a threshold query of the type handled by the sparse vector technique. If the answer is "no" — i.e., the difference, or error, is "below threshold," — then we can respond to the query using the publicly known hypothesis predictor, and have no further privacy loss. If the answer is "yes," meaning that the currently known hypothesis predictor gives rise to an error that is above threshold, then we have found an example appropriate to update our learning algorithm. Because "above threshold" answers correspond exactly to queries needed to update our learning algorithm, the total privacy cost depends only on the learning rate of the algorithm, and not on the total number of queries that we answer.

First we give the multiplicative weights update rule and prove a theorem about its convergence in the language of answering linear queries.

It will be convenient to think of databases x as being probability distributions over the data universe $\mathcal{X}$. That is, letting $\Delta([\mathcal{X}])$ denote the set of probability distributions over the set $[|\mathcal{X}|]$, we have $x \in \Delta([\mathcal{X}])$. Note that we can always scale a database to have this property without changing the normalized value of any linear query.

Algorithm 5 The Multiplicative Weights (MW) Update Rule. It is instantiated with a parameter $\eta \leq 1$. In the following analysis, we will take $\eta = \alpha/2$, where α is the parameter specifying our target accuracy.

MW(x^t, f_t, v_t):

if $v_t < f_t(x^t)$ **then**
  Let $r_t = f_t$
else
  Let $r_t = 1 - f_t$
  (i.e., for all χ_i, $r_t(\chi_i) = 1 - f_t[\chi_i]$)
end if
Update: For all $i \in [|\mathcal{X}|]$ Let

$$\hat{x}_i^{t+1} = \exp(-\eta r_t[i]) \cdot x_i^t$$

$$x_i^{t+1} = \frac{\hat{x}_i^{t+1}}{\sum_{j=1}^{|\mathcal{X}|} \hat{x}_j^{t+1}}$$

Output x^{t+1}.

Theorem 4.10. Fix a class of linear queries $\mathcal{Q}$ and a database $x \in \Delta([\mathcal{X}])$, and let $x^1 \in \Delta([\mathcal{X}])$ describe the uniform distribution over $\mathcal{X}$: $x_i^1 = 1/|\mathcal{X}|$ for all i. Now consider a maximal length sequence of databases x^t for $t \in \{2, \ldots, L\}$ generated by setting $x^{t+1} = \boldsymbol{MW}(x^t, f_t, v_t)$ as described in Algorithm 5, where for each t, $f_t \in \mathcal{Q}$ and $v_t \in \mathbb{R}$ are such that:

1. $|f_t(x) - f_t(x^t)| > \alpha$, and
2. $|f_t(x) - v_t| < \alpha$.

Then it must be that:

$$L \leq 1 + \frac{4 \log |\mathcal{X}|}{\alpha^2}.$$

Note that if we prove this theorem, we will have proven that for the last database x^{L+1} in the sequence it must be that for all $f \in \mathcal{Q}$: $|f(x) - f(x^{L+1})| \leq \alpha$, as otherwise it would be possible to extend the sequence, contradicting maximality. In other words, given *distinguishing queries* f^t, the multiplicative weights update rule learns the private database x with respect to any class of linear queries $\mathcal{Q}$, up to some tolerance α, in only a small number (L) of steps. We will use this theorem as follows. The Private Online Multiplicative Weights algorithm, described (twice!) below, will at all times t have a *public* approximation x^t to the database x. Given an input query f, the algorithm will compute a noisy approximation to the difference $|f(x) - f(x^t)|$. If the (noisy) difference is large, the algorithm will provide a noisy approximation $f(x) + \lambda_t$ to the true answer $f(x)$, where λ_t is drawn from some appropriately chosen Laplace distribution, and the Multiplicative Weights Update Rule will be invoked with parameters $(x^t, f, f(x) + \lambda_t)$. If the update rule is invoked only when the difference $|f(x) - f(x^t)|$ is truly large (Theorem 4.10, condition 1), and if the approximations $f(x) + \lambda_t$ are sufficiently accurate (Theorem 4.10, condition 2), then we can apply the theorem to conclude that updates are not so numerous (because L is not so large) *and* the resulting x^{L+1} gives accurate answers to all queries in $\mathcal{Q}$ (because no distinguishing query remains).

Theorem 4.10 is proved by keeping track of a potential function Ψ measuring the similarity between the hypothesis database x^t at time t, and the true database D. We will show:

1. The potential function does not start out too large.
2. The potential function decreases by a significant amount at each update round.
3. The potential function is always non-negative.

Together, these 3 facts will force us to conclude that there cannot be too many update rounds.

Let us now begin the analysis for the proof of the convergence theorem.

Proof. We must show that any sequence $\{(x^t, f_t, v_t)\}_{t=1,\ldots,L}$ with the property that $|f_t(x^t) - f_t(x)| > \alpha$ and $|v_t - f_t(x)| < \alpha$ cannot have $L > \frac{4\log|\mathcal{X}|}{\alpha^2}$.

We define our potential function as follows. Recall that we here view the database as a probability distribution — i.e., we assume $\|x\|_1 = 1$. Of course this does not require actually modifying the real database. The potential function that we use is the relative entropy, or KL divergence, between x and x^t (when viewed as probability distributions):

$$\Psi_t \stackrel{\text{def}}{=} KL(x\|x^t) = \sum_{i=1}^{|\mathcal{X}|} x[i]\log\left(\frac{x[i]}{x^t[i]}\right).$$

We begin with a simple fact:

Proposition 4.11. For all t: $\Psi_t \geq 0$, and $\Psi_1 \leq \log|\mathcal{X}|$.

Proof. Relative entropy (KL-Divergence) is always a non-negative quantity, by the log-sum inequality, which states that if $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ are non-negative numbers, then

$$\sum_i a_i \log\frac{a_i}{b_i} \geq \left(\sum_i a_i\right)\frac{\sum_i a_i}{\sum_i b_i}.$$

To see that $\Psi_1 \leq \log|\mathcal{X}|$, recall that $x^1[i] = 1/|\mathcal{X}|$ for all i, and so $\Psi_1 = \sum_{i=1}^{|\mathcal{X}|} x[i]\log\left(|\mathcal{X}|x[i]\right)$. Noting that x is a probability distribution, we see that this quantity is maximized when $x[1] = 1$ and $x[i] = 0$ for all $i > 1$, giving $\Psi_i = \log|\mathcal{X}|$. □

We will now argue that at each step, the potential function drops by at least $\alpha^2/4$. Because the potential begins at $\log|\mathcal{X}|$, and must always be non-negative, we therefore know that there can be at most $L \leq 4\log|X|/\alpha^2$ steps in the database update sequence. To begin, let us see exactly how much the potential drops at each step:

Lemma 4.12.

$$\Psi_t - \Psi_{t+1} \geq \eta\left(\langle r_t, x^t\rangle - \langle r_t, x\rangle\right) - \eta^2$$

Proof. Recall that $\sum_{i=1}^{|\mathcal{X}|} x[i] = 1$.

$$\begin{aligned}
\Psi_t - \Psi_{t+1} &= \sum_{i=1}^{|\mathcal{X}|} x[i] \log\left(\frac{x[i]}{x_i^t}\right) - \sum_{i=1}^{|\mathcal{X}|} x[i] \log\left(\frac{x[i]}{x_i^{t+1}}\right) \\
&= \sum_{i=1}^{|\mathcal{X}|} x[i] \log\left(\frac{x_i^{t+1}}{x_i^t}\right) \\
&= \sum_{i=1}^{|\mathcal{X}|} x[i] \log\left(\frac{\hat{x}_i^{t+1}/\sum_i \hat{x}_i^{t+1}}{x_i^t}\right) \\
&= \sum_{i=1}^{|\mathcal{X}|} x[i] \left[\log\left(\frac{x_i^t \exp(-\eta r_t[i]))}{x_i^t}\right)\right. \\
&\quad \left. - \log\left(\sum_{j=1}^{|\mathcal{X}|} \exp(-\eta r_t[j]) x_j^t\right)\right] \\
&= -\left(\sum_{i=1}^{|\mathcal{X}|} x[i]\eta r_t[i]\right) - \log\left(\sum_{i=j}^{|\mathcal{X}|} \exp(-\eta r_t[j]) x_j^t\right) \\
&= -\eta\langle r_t, x\rangle - \log\left(\sum_{j=1}^{|\mathcal{X}|} \exp(-\eta r_t[j]) x_j^t\right) \\
&\geq -\eta\langle r_t, x\rangle - \log\left(\sum_{j=1}^{|\mathcal{X}|} x_j^t (1 + \eta^2 - \eta r_t[j])\right) \\
&= -\eta\langle r_t, x\rangle - \log\left(1 + \eta^2 - \eta\langle r_t, x^t\rangle\right) \\
&\geq \eta\left(\langle r_t, x^t\rangle - \langle r_t, x\rangle\right) - \eta^2.
\end{aligned}$$

The first inequality follows from the fact that:

$$\exp(-\eta r_t[j]) \leq 1 - \eta r_t[j] + \eta^2 (r_t[j])^2 \leq 1 - \eta r_t[j] + \eta^2.$$

The second inequality follows from the fact that $\log(1 + y) \leq y$ for $y > -1$. □

The rest of the proof now follows easily. By the conditions of the database/query sequence (described in the hypothesis for Theorem 4.10 above), for every t,

1. $|f_t(x) - f_t(x^t)| \geq \alpha$ and
2. $|v_t - f_t(x)| < \alpha$.

Thus, $f_t(x) < f_t(x^t)$ if and only if $v_t < f_t(x^t)$. In particular, $r_t = f_t$ if $f_t(x^t) - f_t(x) \geq \alpha$, and $r_t = 1 - f_t$ if $f_t(x) - f_t(x^t) \geq \alpha$. Therefore, by Lemma 4.12 and the choice of $\eta = \alpha/2$ as described in the Update Rule,

$$\Psi_t - \Psi_{t+1} \geq \frac{\alpha}{2}\left(\langle r_t, x^t\rangle - \langle r_t, x\rangle\right) - \frac{\alpha^2}{4} \geq \frac{\alpha}{2}(\alpha) - \frac{\alpha^2}{4} = \frac{\alpha^2}{4}.$$

Finally we know:

$$0 \leq \Psi_L \leq \Psi_0 - L \cdot \frac{\alpha^2}{4} \leq \log|\mathcal{X}| - L\frac{\alpha^2}{4}.$$

Solving, we find: $L \leq \frac{4\log|\mathcal{X}|}{\alpha^2}$. This completes the proof. □

We can now combine the Multiplicative Weights Update Rule with the NumericSparse algorithm to give an interactive query release mechanism. For $(\epsilon, 0)$ privacy, we essentially (with somewhat worse constants) recover the bound for SmallDB. For (ϵ, δ)-differential privacy, we obtain better bounds, by virtue of being able to use the composition theorem. The queries to NumericSparse are asking whether the magnitude of the error given by estimating $f_i(x)$ by applying f_i to the current approximation x^t to x is above an appropriately chosen threshold T, that is, they are asking if $|f(x) - f(x^t)|$ is large. For technical reasons this is done by asking about $f(x) - f(x^t)$ (without the absolute value) and about $f(x^t) - f(x)$. Recall that the NumericSparse algorithm responds with either $\perp$ or some (positive) value exceeding T. We use the mnemonic E for the responses to emphasize that the query is asking about an error.

Theorem 4.13. The Online Multiplicative Weights Mechanism (via NumericSparse) is $(\epsilon, 0)$-differentially private.

Algorithm 6 The Online Multiplicative Weights Mechanism (via NumericSparse) takes as input a private database x, privacy parameters ϵ, δ, accuracy parameters α and β, and a stream of linear queries $\{f_i\}$ that may be chosen adaptively from a class of queries $\mathcal{Q}$. It outputs a stream of answers $\{a_i\}$.

OnlineMW via NumericSparse $(x, \{f_i\}, \epsilon, \delta, \alpha, \beta)$

Let $c \leftarrow \frac{4 \log |\mathcal{X}|}{\alpha^2}$,
if $\delta = 0$ **then**
  Let $T \leftarrow \frac{18c(\log(2|\mathcal{Q}|)+\log(4c/\beta))}{\epsilon||x||_1}$
else
  Let $T \leftarrow \frac{(2+32\sqrt{2})\sqrt{c \log \frac{2}{\delta}}(\log k+\log \frac{4c}{\beta})}{\epsilon||x||_1}$
end if

Initialize NumericSparse$(x, \{f'_i\}, T, c, \epsilon, \delta)$ with a stream of queries $\{f'_i\}$, outputting a stream of answers E_i.
Let $t \leftarrow 0$, and let $x^0 \in \Delta([\mathcal{X}])$ satisfy $x^0_i = 1/|\mathcal{X}|$ for all $i \in [|\mathcal{X}|]$.
for each query f_i **do**
  Let $f'_{2i-1}(\cdot) = f_i(\cdot) - f_i(x^t)$.
  Let $f'_{2i}(\cdot) = f_i(x^t) - f_i(\cdot)$
  if $E_{2i-1} = \perp$ and $E_{2i} = \perp$ **then**
    Let $a_i = f_i(x^t)$
  else
    if $E_{2i-1} \in \mathbb{R}$ **then**
      Let $a_i = f_i(x^t) + E_{2i-1}$
    else
      Let $a_i = f_i(x^t) - E_{2i}$
    end if
    Let $x^{t+1} = MW(x^t, f_i, a_i)$
    Let $t \leftarrow t + 1$.
  end if
end for

Proof. This follows directly from the privacy analysis of NumericSparse, because the OnlineMW algorithm accesses the database only through NumericSparse. □

Speaking informally, the proof of utility for the Online Multiplicative Weights Mechanism (via NumericSparse) uses the utility theorem for the NumericSparse (Theorem 3.28) to conclude that, with high probability, the Multiplicative Weights Update Rule is only invoked when the query f_t is truly a distinguishing query, meaning, $|f_i(x) - f_t(x^t)|$ is "large," *and* the released noisy approximations to $f_i(x)$ are "accurate." Under this assumption, we can apply the convergence theorem (Theorem 4.10) to conclude that the total number of updates is small and therefore the algorithm can answer all queries in $\mathcal{Q}$.

Theorem 4.14. For $\delta = 0$, with probability at least $1-\beta$, for all queries f_i, the Online Multiplicative Weights Mechanism (via NumericSparse) returns an answer a_i such that $|f_i(x) - a_i| \leq 3\alpha$ for any α such that:

$$\alpha \geq \frac{32 \log |\mathcal{X}| \left(\log(|\mathcal{Q}|) + \log\left(\frac{32 \log |\mathcal{X}|}{\alpha^2 \beta}\right)\right)}{\epsilon \alpha^2 ||x||_1}$$

Proof. Recall that, by Theorem 3.28, given k queries and a maximum number c of above-threshold queries, NumericSparse is (α, β)-accurate for any α such that:

$$\alpha \geq \frac{9c(\log k + \log(4c/\beta))}{\epsilon}.$$

In our case $c = 4 \log |\mathcal{X}|/\alpha^2$ and $k = 2|\mathcal{Q}|$, and we have been normalizing, which reduces α by a factor of $||x||_1$. With this in mind, we can take

$$\alpha = \frac{32 \log |\mathcal{X}| \left(\log(|\mathcal{Q}|) + \log\left(\frac{32 \log |\mathcal{X}|}{\alpha^2 \beta}\right)\right)}{\epsilon \alpha^2 ||x||_1}$$

and note that with this value we get $T = 2\alpha$ for the case $\delta = 0$.

Assume we are in this high $(1 - \beta)$ probability case. Then for all i such that f_i triggers an update, $|f_i(x) - f_i(x^t)| \geq T - \alpha = \alpha$ (Theorem 4.10, condition 1). Thus, f_i, a_i form a valid pair of query/value updates as required in the hypothesis of Theorem 4.10 and so, by that theorem, there can be at most $c = \frac{4 \log |\mathcal{X}|}{\alpha^2}$ such update steps.

In addition, still by the accuracy properties of the Sparse Vector algorithm,

1. at most one of E_{2i-1}, E_{2i} will have value $\perp$;

2. for all i such that no update is triggered ($a_i = f_i(x^t)$) we have $|f_i(x) - f_i(x^t)| \leq T + \alpha = 3\alpha$; and
3. for all i such that an update is triggered we have $|f_i(x) - a_i| \leq \alpha$ (Theorem 4.10, condition 2). □

Optimizing the above expression for α and removing the normalization factor, we find that the OnlineMW mechanism can answer each linear query to accuracy 3α except with probability β for:

$$\alpha = ||x||_1^{2/3} \left(\frac{36 \log |\mathcal{X}| \left(\log(|\mathcal{Q}|) + \log \left(\frac{32 \log |\mathcal{X}|^{1/3} ||x||_1^{2/3}}{\beta} \right) \right)}{\epsilon} \right)^{1/3}$$

which is comparable to the SmallDB mechanism.

By repeating the same argument, but instead using the utility theorem for the (ϵ, δ)-private version of Sparse Vector (Theorem 3.28), we obtain the following theorem.

Theorem 4.15. For $\delta > 0$, with probability at least $1 - \beta$, for all queries f_i, OnlineMW returns an answer a_i such that $|f_i(x) - a_i| \leq 3\alpha$ for any α such that:

$$\alpha \geq \frac{(2 + 32\sqrt{2}) \cdot \sqrt{\log |\mathcal{X}| \log \frac{2}{\delta}} \left(\log |\mathcal{Q}| + \log \left(\frac{32 \log |\mathcal{X}|}{\alpha^2 \beta} \right) \right)}{\alpha \epsilon ||x||_1}$$

Again optimizing the above expression for α and removing the normalization factor, we find that the OnlineMW mechanism can answer each linear query to accuracy 3α except with probability β, for:

$$\alpha = ||x||_1^{1/2} \left(\frac{(2 + 32\sqrt{2}) \cdot \sqrt{\log |\mathcal{X}| \log \frac{2}{\delta}} \left(\log |\mathcal{Q}| + \log \left(\frac{32 ||x||_1}{\beta} \right) \right)}{\epsilon} \right)^{1/2}$$

which gives better accuracy (as a function of $||x||_1$) than the SmallDB mechanism. Intuitively, the greater accuracy comes from the iterative nature of the mechanism, which allows us to take advantage of our composition theorems for (ϵ, δ)-privacy. The SmallDB mechanism runs

in just a single shot, and so there is no opportunity to take advantage of composition.

The accuracy of the private multiplicative weights algorithm has dependencies on several parameters, which are worth further discussion. In the end, the algorithm answers queries using the *sparse vector technique* paired with a *learning algorithm for linear functions.* As we proved in the last section, the sparse vector technique introduces error that scales like $O(c \log k/(\epsilon\|x\|_1))$ when a total of k sensitivity $1/\|x\|_1$ queries are made, and at most c of them can have "above threshold" answers, for any threshold T. Recall that these error terms arise because the privacy analysis for the sparse vector algorithm allows us to "pay" only for the above threshold queries, and therefore can add noise $O(c/(\epsilon\|x\|_1))$ to each query. On the other hand, since we end up adding independent Laplace noise with scale $\Omega(c/(\epsilon\|x\|_1))$ to k queries in total, we expect that the maximum error over all k queries is larger by a $\log k$ factor. But what is c, and what queries should we ask? The multiplicative weights learning algorithm gives us a query strategy and a guarantee that no more than $c = O(\log|\mathcal{X}|/\alpha^2)$ queries will be above a threshold of $T = O(\alpha)$, for any α. (The queries we ask are always: " How much does the real answer differ from the predicted answer of the current multiplicative weights hypothesis." The answers to these questions both give us the true answers to the queries, as well as instructions how to update the learning algorithm appropriately when a query is above threshold.) Together, this leads us to set the threshold to be $O(\alpha)$, where α is the expression that satisfies: $\alpha = O(\log|\mathcal{X}|\log k/(\epsilon\|x\|_1\alpha^2))$. This minimizes the two sources of error: error from the sparse vector technique, and error from failing to update the multiplicative weights hypothesis.

4.3 Bibliographical notes

The offline query release mechanism given in this section is from Blum et al. [8], which gave bounds in terms of the VC-Dimension of the query class (Theorem 4.9). The generalization to fat shattering dimension is given in [72].

The online query release mechanism given in this section is from Hardt and Rothblum [44]. This mechanism uses the classic multiplicative weights update method, for which Arora, Hazan and Kale give an excellent survey [1]. Slightly improved bounds for the private multiplicative weights mechanism were given by Gupta et al. [39], and the analysis here follows the presentation from [39].