Foundations and Trends® in
Theoretical Computer Science
Vol. 9, Nos. 3–4 (2014) 211–407

DOI: 10.1561/0400000042

now
the essence of knowledge

The Algorithmic Foundations of Differential Privacy

Cynthia Dwork
Microsoft Research, USA
dwork@microsoft.com

Aaron Roth
University of Pennsylvania, USA
aaroth@cis.upenn.edu

Contents

Abstract

The problem of privacy-preserving data analysis has a long history spanning multiple disciplines. As electronic data about individuals becomes increasingly detailed, and as technology enables ever more powerful collection and curation of these data, the need increases for a robust, meaningful, and mathematically rigorous definition of privacy, together with a computationally rich class of algorithms that satisfy this definition. Differential Privacy is such a definition.

After motivating and discussing the meaning of differential privacy, the preponderance of this monograph is devoted to fundamental techniques for achieving differential privacy, and application of these techniques in creative combinations, using the query-release problem as an ongoing example. A key point is that, by rethinking the computational goal, one can often obtain far better results than would be achieved by methodically replacing each step of a non-private computation with a differentially private implementation. Despite some astonishingly powerful computational results, there are still fundamental limitations — not just on what can be achieved with differential privacy but on what can be achieved with any method that protects against a complete breakdown in privacy. Virtually all the algorithms discussed herein maintain differential privacy against adversaries of arbitrary computational power. Certain algorithms are computationally intensive, others are efficient. Computational complexity for the adversary and the algorithm are both discussed.

We then turn from fundamentals to applications other than query-release, discussing differentially private methods for mechanism design and machine learning. The vast majority of the literature on differentially private algorithms considers a single, static, database that is subject to many analyses. Differential privacy in other models, including distributed databases and computations on data streams is discussed.

Finally, we note that this work is meant as a thorough introduction to the problems and techniques of differential privacy, but is not intended to be an exhaustive survey — there is by now a vast amount of work in differential privacy, and we can cover only a small portion of it.

C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends® in Theoretical Computer Science, vol. 9, nos. 3–4, pp. 211–407, 2014.
DOI: 10.1561/0400000042.

Preface

The problem of privacy-preserving data analysis has a long history spanning multiple disciplines. As electronic data about individuals becomes increasingly detailed, and as technology enables ever more powerful collection and curation of these data, the need increases for a robust, meaningful, and mathematically rigorous definition of privacy, together with a computationally rich class of algorithms that satisfy this definition. *Differential Privacy* is such a definition.

After motivating and discussing the meaning of differential privacy, the preponderance of the book is devoted to fundamental techniques for achieving differential privacy, and application of these techniques in creative combinations (Sections 3–7), using the *query-release* problem as an ongoing example. A key point is that, by rethinking the computational goal, one can often obtain far better results than would be achieved by methodically replacing each step of a non-private computation with a differentially private implementation.

Despite some astonishingly powerful computational results, there are still fundamental limitations — not just on what can be achieved with differential privacy but on what can be achieved with *any* method that protects against a complete breakdown in privacy (Section 8).

Virtually all the algorithms discussed in this book maintain differential privacy against adversaries of arbitrary computational power. Certain algorithms are computationally intensive, others are

efficient. Computational complexity for the adversary and the algorithm are both discussed in Section 9.

In Sections 10 and 11 we turn from fundamentals to applications other than query-release, discussing differentially private methods for mechanism design and machine learning. The vast majority of the literature on differentially private algorithms considers a single, static, database that is subject to many analyses. Differential privacy in other models, including distributed databases and computations on data streams is discussed in Section 12.

Finally, we note that this book is meant as a thorough introduction to the problems and techniques of differential privacy, but is not intended to be an exhaustive survey — there is by now a vast amount of work in differential privacy, and we can cover only a small portion of it.

1

The Promise of Differential Privacy

"Differential privacy" describes a promise, made by a data holder, or *curator*, to a data subject: "You will not be affected, adversely or otherwise, by allowing your data to be used in any study or analysis, no matter what other studies, data sets, or information sources, are available." At their best, differentially private database mechanisms can make confidential data widely available for accurate data analysis, without resorting to data clean rooms, data usage agreements, data protection plans, or restricted views. Nonetheless, data utility will eventually be consumed: the Fundamental Law of Information Recovery states that overly accurate answers to too many questions will destroy privacy in a spectacular way.[1] The goal of algorithmic research on differential privacy is to postpone this inevitability as long as possible.

Differential privacy addresses the paradox of learning nothing about an individual while learning useful information about a population. A medical database may teach us that smoking causes cancer, affecting an insurance company's view of a smoker's long-term medical costs. Has the smoker been harmed by the analysis? Perhaps — his insurance

[1] This result, proved in Section 8.1, applies to *all* techniques for privacy-preserving data analysis, and not just to differential privacy.

premiums may rise, if the insurer knows he smokes. He may also be helped — learning of his health risks, he enters a smoking cessation program. Has the smoker's privacy been compromised? It is certainly the case that more is known about him after the study than was known before, but was his information "leaked"? Differential privacy will take the view that it was not, with the rationale that the impact on the smoker is the same *independent of whether or not he was in the study.* It is the *conclusions reached* in the study that affect the smoker, not his presence or absence in the data set.

Differential privacy ensures that the same conclusions, for example, smoking causes cancer, will be reached, independent of whether any individual opts into or opts out of the data set. Specifically, it ensures that any sequence of outputs (responses to queries) is "essentially" equally likely to occur, independent of the presence or absence of any individual. Here, the probabilities are taken over random choices made by the privacy mechanism (something controlled by the data curator), and the term "essentially" is captured by a parameter, ε. A smaller ε will yield better privacy (and less accurate responses).

Differential privacy is a *definition,* not an algorithm. For a given computational task T and a given value of ε there will be many differentially private algorithms for achieving T in an ε-differentially private manner. Some will have better accuracy than others. When ε is small, finding a highly accurate ε-differentially private algorithm for T can be difficult, much as finding a numerically stable algorithm for a specific computational task can require effort.

1.1 Privacy-preserving data analysis

Differential privacy is a definition of privacy tailored to the problem of privacy-preserving data analysis. We briefly address some concerns with other approaches to this problem.

Data Cannot be Fully Anonymized and Remain Useful. Generally speaking, the richer the data, the more interesting and useful it is. This has led to notions of "anonymization" and "removal of personally identifiable information," where the hope is that portions of the

data records can be suppressed and the remainder published and used for analysis. However, the richness of the data enables "naming" an individual by a sometimes surprising collection of fields, or attributes, such as the combination of zip code, date of birth, and sex, or even the names of three movies and the approximate dates on which an individual watched these movies. This "naming" capability can be used in a *linkage attack* to match "anonymized" records with non-anonymized records in a different dataset. Thus, the medical records of the governor of Massachussetts were identified by matching anonymized medical encounter data with (publicly available) voter registration records, and Netflix subscribers whose viewing histories were contained in a collection of anonymized movie records published by Netflix as training data for a competition on recommendation were identified by linkage with the Internet Movie Database (IMDb).

Differential privacy neutralizes linkage attacks: since being differentially private is a property of the data access mechanism, and is unrelated to the presence or absence of auxiliary information available to the adversary, access to the IMDb would no more permit a linkage attack to someone whose history is in the Netflix training set than to someone not in the training set.

Re-Identification of "Anonymized" Records is Not the Only Risk. Re-identification of "anonymized" data records is clearly undesirable, not only because of the re-identification *per se*, which certainly reveals membership in the data set, but also because the record may contain compromising information that, were it tied to an individual, could cause harm. A collection of medical encounter records from a specific urgent care center on a given date may list only a small number of distinct complaints or diagnoses. The additional information that a neighbor visited the facility on the date in question gives a fairly narrow range of possible diagnoses for the neighbor's condition. The fact that it may not be possible to match a specific record to the neighbor provides minimal privacy protection to the neighbor.

Queries Over Large Sets are Not Protective. Questions about specific individuals cannot be safely answered with accuracy, and indeed one

might wish to reject them out of hand (were it computationally feasible to recognize them). Forcing queries to be over large sets is not a panacea, as shown by the following *differencing attack*. Suppose it is known that Mr. X is in a certain medical database. Taken together, the answers to the two large queries "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?" yield the sickle cell status of Mr. X.

Query Auditing Is Problematic. One might be tempted to *audit* the sequence of queries and responses, with the goal of interdicting any response if, in light of the history, answering the current query would compromise privacy. For example, the auditor may be on the lookout for pairs of queries that would constitute a differencing attack. There are two difficulties with this approach. First, it is possible that *refusing* to answer a query is itself disclosive. Second, query auditing can be computationally infeasible; indeed if the query language is sufficiently rich there may not even exist an algorithmic procedure for deciding if a pair of queries constitutes a differencing attack.

Summary Statistics are Not "Safe." In some sense, the failure of summary statistics as a privacy solution concept is immediate from the differencing attack just described. Other problems with summary statistics include a variety of *reconstruction attacks* against a database in which each individual has a "secret bit" to be protected. The utility goal may be to permit, for example, questions of the form "How many people satisfying property P have secret bit value 1?" The goal of the adversary, on the other hand, is to significantly increase his chance of guessing the secret bits of individuals. The reconstruction attacks described in Section 8.1 show the difficulty of protecting against even a *linear* number of queries of this type: unless sufficient inaccuracy is introduced almost all the secret bits can be reconstructed.

A striking illustration of the risks of releasing summary statistics is in an application of a statistical technique, originally intended for confirming or refuting the presence of an individual's DNA in a forensic mix, to ruling an individual in or out of a genome-wide association study. According to a Web site of the Human Genome Project, "Single nucleotide polymorphisms, or SNPs (pronounced "snips"), are DNA

sequence variations that occur when a single nucleotide (A,T,C, or G) in the genome sequence is altered. For example a SNP might change the DNA sequence AAGGCTAA to ATGGCTAA." In this case we say there are two alleles: A and T. For such a SNP we can ask, given a particular reference population, what are the frequencies of each of the two possible alleles? Given the allele frequencies for SNPs in the reference population, we can examine how these frequencies may differ for a subpopulation that has a particular disease (the "case" group), looking for alleles that are associated with the disease. For this reason, genome-wide association studies may contain the allele frequencies of the case group for large numbers of SNPs. By definition, these allele frequencies are only aggregated statistics, and the (erroneous) assumption has been that, by virtue of this aggregation, they preserve privacy. However, given the genomic data of an individual, it is theoretically possible to determine if the individual is in the case group (and, therefore, has the disease). In response, the National Institutes of Health and Wellcome Trust terminated public access to aggregate frequency data from the studies they fund.

This is a challenging problem even for differential privacy, due to the large number — hundreds of thousands or even one million — of measurements involved and the relatively small number of individuals in any case group.

"Ordinary" Facts are Not "OK." Revealing "ordinary" facts, such as purchasing bread, may be problematic if a data subject is followed over time. For example, consider Mr. T, who regularly buys bread, year after year, until suddenly switching to rarely buying bread. An analyst might conclude Mr. T most likely has been diagnosed with Type 2 diabetes. The analyst might be correct, or might be incorrect; either way Mr. T is harmed.

"Just a Few." In some cases a particular technique may in fact provide privacy protection for "typical" members of a data set, or more generally, "most" members. In such cases one often hears the argument that the technique is adequate, as it compromises the privacy of "just a few" participants. Setting aside the concern that outliers may be precisely those people for whom privacy is most important, the "just a few"

philosophy is not intrinsically without merit: there is a social judgment, a weighing of costs and benefits, to be made. A well-articulated definition of privacy consistent with the "just a few" philosophy has yet to be developed; however, for a single data set, "just a few" privacy can be achieved by randomly selecting a subset of rows and releasing them in their entirety (Lemma 4.3, Section 4). Sampling bounds describing the quality of statistical analysis that can be carried out on random subsamples govern the number of rows to be released. Differential privacy provides an alternative when the "just a few" philosophy is rejected.

1.2 Bibliographic notes

Sweeney [81] linked voter registration records to "anonymized" medical encounter data; Narayanan and Shmatikov carried out a linkage attack against anonymized ranking data published by Netflix [65]. The work on presence in a forensic mix is due to Homer et al. [46]. The first reconstruction attacks were due to Dinur and Nissim [18].

2

Basic Terms

This section motivates and presents the formal definition of differential privacy, and enumerates some of its key properties.

2.1 The model of computation

We assume the existence of a trusted and trustworthy *curator* who holds the data of *individuals* in a database D, typically comprised of some number n of rows. The intuition is that each row contains the data of a single individual, and, still speaking intuitively, the privacy goal is to simultaneously protect every individual row while permitting statistical analysis of the database as a whole.

In the *non-interactive*, or *offline*, model the curator produces some kind of object, such as a "synthetic database," collection of summary statistics, or "sanitized database" once and for all. After this *release* the curator plays no further role and the original data may be destroyed.

A *query* is a function to be applied to a database. The *interactive*, or *online*, model permits the data analyst to ask queries adaptively, deciding which query to pose next based on the observed responses to previous queries.

The trusted curator can be replaced by a protocol run by the set of individuals, using the cryptographic techniques for secure multi-party protocols, but for the most part we will not be appealing to cryptographic assumptions. Section 12 describes this and other models studied in the literature.

When all the queries are known in advance the non-interactive model should give the best accuracy, as it is able to correlate noise knowing the structure of the queries. In contrast, when no information about the queries is known in advance, the non-interactive model poses severe challenges, as it must provide answers to all possible queries. As we will see, to ensure privacy, or even to prevent privacy catastrophes, accuracy will necessarily deteriorate with the number of questions asked, and providing accurate answers to all possible questions will be infeasible.

A *privacy mechanism*, or simply a *mechanism*, is an algorithm that takes as input a database, a universe $\mathcal{X}$ of data types (the set of all possible database rows), random bits, and, optionally, a set of queries, and produces an output string. The hope is that the output string can be decoded to produce relatively accurate answers to the queries, if the latter are present. If no queries are presented then we are in the non-interactive case, and the hope is that the output string can be interpreted to provide answers to future queries.

In some cases we may require that the output string be a *synthetic database*. This is a multiset drawn from the universe $\mathcal{X}$ of possible database rows. The decoding method in this case is to carry out the query on the synthetic database and then to apply some sort of simple transformation, such as multiplying by a scaling factor, to obtain an approximation to the the true answer to the query.

2.2 Towards defining private data analysis

A natural approach to defining privacy in the context of data analysis is to require that the analyst knows no more about any individual in the data set after the analysis is completed than she knew before the analysis was begun. It is also natural to formalize this goal by

requiring that the adversary's prior and posterior views about an individual (i.e., before and after having access to the database) shouldn't be "too different," or that access to the database shouldn't change the adversary's views about any individual "too much." However, if the database teaches anything at all, this notion of privacy is unachievable. For example, suppose the adversary's (incorrect) prior view is that everyone has 2 left feet. Access to the statistical database teaches that almost everyone has one left foot and one right foot. The adversary now has a very different view of whether or not any given respondent has two left feet.

Part of the appeal of before/after, or "nothing is learned," approach to defining privacy is the intuition that if nothing is learned about an individual then the individual cannot be harmed by the analysis. However, the "smoking causes cancer" example shows this intuition to be flawed; the culprit is auxiliary information (Mr. X smokes).

The "nothing is learned" approach to defining privacy is reminiscent of semantic security for a cryptosystem. Roughly speaking, semantic security says that nothing is learned about the plaintext (the unencrypted message) from the ciphertext. That is, anything known about the plaintext after seeing the ciphertext was known before seeing the ciphertext. So if there is auxiliary information saying that the ciphertext is an encryption of either "dog" or "cat," then the ciphertext leaks no further information about which of "dog" or "cat" has been encrypted. Formally, this is modeled by comparing the ability of the eavesdropper to guess which of "dog" and "cat" has been encrypted to the ability of a so-called *adversary simulator*, who has the auxiliary information but does not have access to the ciphertext, to guess the same thing. If for every eavesdropping adversary, and all auxiliary information (to which both the adversary and the simulator are privy), the adversary simulator has essentially the same odds of guessing as does the eavesdropper, then the system enjoys semantic security. Of course, for the system to be useful, the legitimate receiver must be able to correctly decrypt the message; otherwise semantic security can be achieved trivially.

We know that, under standard computational assumptions, semantically secure cryptosystems exist, so why can we not build semantically

secure private database mechanisms that yield answers to queries while keeping individual rows secret?

First, the analogy is not perfect: in a semantically secure cryptosystem there are three parties: the message sender (who encrypts the plaintext message), the message receiver (who decrypts the ciphertext), and the eavesdropper (who is frustrated by her inability to learn anything about the plaintext that she did not already know before it was sent). In contrast, in the setting of private data analysis there are only two parties: the curator, who runs the privacy mechanism (analogous to the sender) and the data analyst, who receives the informative responses to queries (like the message receiver) and also tries to squeeze out privacy-compromising information about individuals (like the eavesdropper). Because the legitimate receiver is the same party as the snooping adversary, the analogy to encryption is flawed: denying all information to the adversary means denying all information to the data analyst.

Second, as with an encryption scheme, we require the privacy mechanism to be useful, which means that it teaches the analyst something she did not previously know. This teaching is unavailable to an adversary simulator; that is, no simulator can "predict" what the analyst has learned. We can therefore look at the database as a weak source of random (unpredictable) bits, from which we can extract some very high quality randomness to be used as a *random pad*. This can be used in an encryption technique in which a secret message is added to a random value (the "random pad") in order to produce a string that information-theoretically hides the secret. Only someone knowing the random pad can learn the secret; any party that knows nothing about the pad learns nothing at all about the secret, no matter his or her computational power. Given access to the database, the analyst can learn the random pad, but the adversary simulator, not given access to the database, learns nothing at all about the pad. Thus, given as auxiliary information the encryption of a secret using the random pad, the analyst can decrypt the secret, but the adversary simulator learns nothing at all about the secret. This yields a huge disparity between the ability of the adversary/analyst to learn the secret and the ability

of the adversary simulator to do the same thing, eliminating all hope of anything remotely resembling semantic security.

The obstacle in both the smoking causes cancer example and the hope for semantic security is auxiliary information. Clearly, to be meaningful, a privacy guarantee must hold even in the context of "reasonable" auxiliary knowledge, but separating reasonable from arbitrary auxiliary knowledge is problematic. For example, the analyst using a government database might be an employee at a major search engine company. What are "reasonable" assumptions about the auxiliary knowledge information available to such a person?

2.3 Formalizing differential privacy

We will begin with the technical definition of differential privacy, and then go on to interpret it. Differential privacy will provide privacy by *process*; in particular it will introduce randomness. An early example of privacy by randomized process is *randomized response*, a technique developed in the social sciences to collect statistical information about embarassing or illegal behavior, captured by having a property P. Study participants are told to report whether or not they have property P as follows:

1. Flip a coin.
2. If **tails**, then respond truthfully.
3. If **heads**, then flip a second coin and respond "Yes" if heads and "No" if tails.

"Privacy" comes from the plausible deniability of any outcome; in particular, if having property P corresponds to engaging in illegal behavior, even a "Yes" answer is not incriminating, since this answer occurs with probability at least 1/4 whether or not the respondent actually has property P. Accuracy comes from an understanding of the noise generation procedure (the introduction of spurious "Yes" and "No" answers from the randomization): The expected number of "Yes" answers is 1/4 times the number of participants who do not have property P plus 3/4 the number having property P. Thus, if p is the true fraction of

participants having property P, the expected number of "Yes" answers is $(1/4)(1-p)+(3/4)p = (1/4)+p/2$. Thus, we can estimate p as twice the fraction answering "Yes" minus $1/2$, that is, $2((1/4)+p/2)-1/2$.

Randomization is essential; more precisely, any *non-trivial* privacy guarantee that holds regardless of all present or even future sources of auxiliary information, including other databases, studies, Web sites, on-line communities, gossip, newspapers, government statistics, and so on, requires randomization. This follows from a simple hybrid argument, which we now sketch. Suppose, for the sake of contradiction, that we have a non-trivial deterministic algorithm. Non-triviality says that there exists a query and two databases that yield different outputs under this query. Changing one row at a time we see there exists a pair of databases differing only in the value of a single row, on which the same query yields different outputs. An adversary knowing that the database is one of these two almost identical databases learns the value of the data in the unknown row.

We will therefore need to discuss the input and output space of randomized algorithms. Throughout this monograph we work with discrete probability spaces. Sometimes we will describe our algorithms as sampling from continuous distributions, but these should always be discretized to finite precision in an appropriately careful way (see Remark 2.1 below). In general, a randomized algorithm with domain A and (discrete) range B will be associated with a mapping from A to the probability simplex over B, denoted $\Delta(B)$:

Definition 2.1 (Probability Simplex)**.** Given a discrete set B, the *probability simplex* over B, denoted $\Delta(B)$ is defined to be:

$$\Delta(B) = \left\{ x \in \mathbb{R}^{|B|} : x_i \geq 0 \text{ for all } i \text{ and } \sum_{i=1}^{|B|} x_i = 1 \right\}$$

Definition 2.2 (Randomized Algorithm)**.** A randomized algorithm $\mathcal{M}$ with domain A and discrete range B is associated with a mapping $M : A \to \Delta(B)$. On input $a \in A$, the algorithm $\mathcal{M}$ outputs $\mathcal{M}(a) = b$ with probability $(M(a))_b$ for each $b \in B$. The probability space is over the coin flips of the algorithm $\mathcal{M}$.

We will think of databases x as being collections of records from a universe $\mathcal{X}$. It will often be convenient to represent databases by their histograms: $x \in \mathbb{N}^{|\mathcal{X}|}$, in which each entry x_i represents the number of elements in the database x of *type* $i \in \mathcal{X}$ (we abuse notation slightly, letting the symbol $\mathbb{N}$ denote the set of all non-negative integers, including zero). In this representation, a natural measure of the distance between two databases x and y will be their ℓ_1 distance:

Definition 2.3 (Distance Between Databases)**.** The ℓ_1 norm of a database x is denoted $\|x\|_1$ and is defined to be:

$$\|x\|_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i| .$$

The ℓ_1 distance between two databases x and y is $\|x - y\|_1$

Note that $\|x\|_1$ is a measure of the *size* of a database x (i.e., the number of records it contains), and $\|x - y\|_1$ is a measure of how many records *differ* between x and y.

Databases may also be represented by multisets of *rows* (elements of $\mathcal{X}$) or even ordered lists of rows, which is a special case of a set, where the row number becomes part of the name of the element. In this case distance between databases is typically measured by the Hamming distance, i.e., the number of rows on which they differ.

However, unless otherwise noted, we will use the histogram representation described above. (Note, however, that even when the histogram notation is more mathematically convenient, in actual implementations, the multiset representation will often be much more concise).

We are now ready to formally define *differential privacy*, which intuitively will guarantee that a randomized algorithm behaves similarly on similar input databases.

Definition 2.4 (Differential Privacy)**.** A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta,$$

where the probability space is over the coin flips of the mechanism $\mathcal{M}$. If $\delta = 0$, we say that $\mathcal{M}$ is ε-differentially private.

Typically we are interested in values of δ that are less than the inverse of any polynomial in the size of the database. In particular, values of δ on the order of $1/\|x\|_1$ are very dangerous: they permit "preserving privacy" by publishing the complete records of a small number of database participants — precisely the "just a few" philosophy discussed in Section 1.

Even when δ is negligible, however, there are theoretical distinctions between $(\varepsilon, 0)$- and (ε, δ)-differential privacy. Chief among these is what amounts to a switch of quantification order. $(\varepsilon, 0)$-differential privacy ensures that, for *every* run of the mechanism $\mathcal{M}(x)$, the output observed is (almost) equally likely to be observed on *every* neighboring database, simultaneously. In contrast (ε, δ)-differential privacy says that for every pair of neighboring databases x, y, it is extremely unlikely that, *ex post facto* the observed value $\mathcal{M}(x)$ will be much more or much less likely to be generated when the database is x than when the database is y. However, given an output $\xi \sim \mathcal{M}(x)$ it may be possible to find a database y such that ξ is much more likely to be produced on y than it is when the database is x. That is, the mass of ξ in the distribution $\mathcal{M}(y)$ may be substantially larger than its mass in the distribution $\mathcal{M}(x)$.

The quantity

$$\mathcal{L}^{(\xi)}_{\mathcal{M}(x)\|\mathcal{M}(y)} = \ln\left(\frac{\Pr[\mathcal{M}(x) = \xi]}{\Pr[\mathcal{M}(y) = \xi]}\right)$$

is important to us; we refer to it as the *privacy loss* incurred by observing ξ. This loss might be positive (when an event is more likely under x than under y) or it might be negative (when an event is more likely under y than under x). As we will see in Lemma 3.17, (ε, δ)-differential privacy ensures that for all adjacent x, y, the absolute value of the privacy loss will be bounded by ε with probability at least $1-\delta$. As always, the probability space is over the coins of the mechanism $\mathcal{M}$.

Differential privacy is immune to *post-processing*: A data analyst, without additional knowledge about the private database, cannot compute a function of the output of a private algorithm $\mathcal{M}$ and make it

less differentially private. That is, if an algorithm protects an individual's privacy, then a data analyst cannot increase privacy loss — either under the formal definition or even in any intuitive sense — simply by sitting in a corner and *thinking about* the output of the algorithm. Formally, the composition of a data-independent mapping f with an (ε, δ)-differentially private algorithm $\mathcal{M}$ is also (ε, δ) differentially private:

Proposition 2.1 (Post-Processing)**.** Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomized algorithm that is (ε, δ)-differentially private. Let $f : R \to R'$ be an arbitrary randomized mapping. Then $f \circ \mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R'$ is (ε, δ)-differentially private.

Proof. We prove the proposition for a deterministic function $f : R \to R'$. The result then follows because any randomized mapping can be decomposed into a convex combination of deterministic functions, and a convex combination of differentially private mechanisms is differentially private.

Fix any pair of neighboring databases x, y with $\|x - y\|_1 \leq 1$, and fix any event $S \subseteq R'$. Let $T = \{r \in R : f(r) \in S\}$. We then have:

$$\begin{aligned} \Pr[f(\mathcal{M}(x)) \in S] &= \Pr[\mathcal{M}(x) \in T] \\ &\leq \exp(\epsilon) \Pr[\mathcal{M}(y) \in T] + \delta \\ &= \exp(\epsilon) \Pr[f(\mathcal{M}(y)) \in S] + \delta \end{aligned}$$

which was what we wanted. □

It follows immediately from Definition 2.4 that $(\varepsilon, 0)$-differential privacy composes in a straightforward way: the composition of two $(\varepsilon, 0)$-differentially private mechanisms is $(2\varepsilon, 0)$-differentially private. More generally (Theorem 3.16), "the epsilons and the deltas add up": the composition of k differentially private mechanisms, where the ith mechanism is $(\varepsilon_i, \delta_i)$-differentially private, for $1 \leq i \leq k$, is $(\sum_i \varepsilon_i, \sum_i \delta_i)$-differentially private.

Group privacy for $(\varepsilon, 0)$-differentially private mechanisms also follows immediately from Definition 2.4, with the strength of the privacy guarantee drops linearly with the size of the group.

Theorem 2.2. Any $(\varepsilon, 0)$-differentially private mechanism $\mathcal{M}$ is $(k\varepsilon, 0)$-differentially private for groups of size k. That is, for all $\|x - y\|_1 \leq k$ and all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(k\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}],$$

where the probability space is over the coin flips of the mechanism $\mathcal{M}$.

This addresses, for example, the question of privacy in surveys that include multiple family members.[1]

More generally, composition and group privacy are not the same thing and the improved composition bounds in Section 3.5.2 (Theorem 3.20), which substantially improve upon the factor of k, do not — and cannot — yield the same gains for group privacy, even when $\delta = 0$.

2.3.1 What differential privacy promises

An Economic View. Differential privacy promises to protect individuals from any *additional* harm that they might face due to their data being in the private database x that they would not have faced had their data not been part of x. Although individuals may indeed face harm once the results $\mathcal{M}(x)$ of a differentially private mechanism $\mathcal{M}$ have been released, differential privacy promises that the probability of harm was not significantly increased by their choice to participate. This is a very utilitarian definition of privacy, because when an individual is deciding whether or not to include her data in a database that will be used in a differentially private manner, it is exactly this difference that she is considering: the probability of harm given that she participates, as compared to the probability of harm given that she does not participate. She has no control over the remaining contents of the database. Given the promise of differential privacy, she is assured that she should

[1] However, as the group gets larger, the privacy guarantee deteriorates, and this is what we want: clearly, if we replace an entire surveyed population, say, of cancer patients, with a completely different group of respondents, say, healthy teenagers, we *should* get different answers to queries about the fraction of respondents who regularly run three miles each day. Although something similar holds for (ε, δ)-differential privacy, the approximation term δ takes a big hit, and we only obtain $(k\varepsilon, ke^{(k-1)\varepsilon}\delta)$-differential privacy for groups of size k.

be almost indifferent between participating and not, from the point of view of future harm. Given any incentive — from altruism to monetary reward — differential privacy may convince her to allow her data to be used. This intuition can be formalized in a utility-theoretic sense, which we here briefly sketch.

Consider an individual i who has arbitrary preferences over the set of all possible future events, which we denote by $\mathcal{A}$. These preferences are expressed by a utility function $u_i : \mathcal{A} \to \mathbb{R}_{\geq 0}$, and we say that individual i experiences utility $u_i(a)$ in the event that $a \in \mathcal{A}$ comes to pass. Suppose that $x \in \mathbb{N}^{|\mathcal{X}|}$ is a data-set containing individual is private data, and that $\mathcal{M}$ is an ε-differentially private algorithm. Let y be a data-set that is identical to x except that it does not include the data of individual i (in particular, $\|x - y\|_1 = 1$), and let $f : \mathrm{Range}(\mathcal{M}) \to \Delta(\mathcal{A})$ be the (arbitrary) function that determines the distribution over future events $\mathcal{A}$, conditioned on the output of mechanism M. By the guarantee of differential privacy, together with the resilience to arbitrary post-processing guaranteed by Proposition 2.1, we have:

$$\begin{aligned}
\mathbb{E}_{a \sim f(\mathcal{M}(x))}[u_i(a)] &= \sum_{a \in \mathcal{A}} u_i(a) \cdot \Pr_{f(\mathcal{M}(x))}[a] \\
&\leq \sum_{a \in \mathcal{A}} u_i(a) \cdot \exp(\varepsilon) \Pr_{f(\mathcal{M}(y))}[a] \\
&= \exp(\varepsilon) \mathbb{E}_{a \sim f(\mathcal{M}(y))}[u_i(a)]
\end{aligned}$$

Similarly,

$$\mathbb{E}_{a \sim f(\mathcal{M}(x))}[u_i(a)] \geq \exp(-\varepsilon) \mathbb{E}_{a \sim f(\mathcal{M}(y))}[u_i(a)].$$

Hence, by promising a guarantee of ε-differential privacy, a data analyst can promise an individual that his expected future utility will not be harmed by more than an $\exp(\varepsilon) \approx (1+\varepsilon)$ factor. Note that this promise holds *independently* of the individual is utility function u_i, and holds *simultaneously* for multiple individuals who may have completely different utility functions.

2.3.2 What differential privacy does not promise

As we saw in the Smoking Causes Cancer example, while differential privacy is an extremely strong guarantee, it does not promise unconditional freedom from harm. Nor does it create privacy where none previously exists. More generally, differential privacy does not guarantee that what one believes to be one's secrets will remain secret. It merely ensures that one's participation in a survey will not in itself be disclosed, nor will participation lead to disclosure of any specifics that one has contributed to the survey. It is very possible that conclusions drawn from the survey may reflect statistical information about an individual. A health survey intended to discover early indicators of a particular ailment may produce strong, even conclusive results; that these conclusions hold for a given individual is not evidence of a differential privacy violation; the individual may not even have participated in the survey (again, differential privacy ensures that these conclusive results would be obtained with very similar probability whether or not the individual participated in the survey). In particular, if the survey teaches us that specific *private* attributes correlate strongly with *publicly observable* attributes, this is not a violation of differential privacy, since this same correlation would be observed with almost the same probability independent of the presence or absence of any respondent.

Qualitative Properties of Differential Privacy. Having introduced and formally defined differential privacy, we recaptiluate its key desirable qualities.

1. *Protection against arbitrary risks*, moving beyond protection against re-identification.
2. *Automatic neutralization of linkage attacks*, including all those attempted with all past, present, *and future* datasets and other forms and sources of auxiliary information.
3. *Quantification of privacy loss.* Differential privacy is not a binary concept, and has a measure of privacy loss. This permits comparisons among different techniques: for a fixed bound on privacy loss, which technique provides better accuracy? For a fixed accuracy, which technique provides better privacy?

4. *Composition.* Perhaps most crucially, the quantification of loss also permits the analysis and control of cumulative privacy loss over multiple computations. Understanding the behavior of differentially private mechanisms under composition enables the design and analysis of complex differentially private algorithms from simpler differentially private building blocks.
5. *Group Privacy.* Differential privacy permits the analysis and control of privacy loss incurred by groups, such as families.
6. *Closure Under Post-Processing* Differential privacy is immune to post-processing: A data analyst, without additional knowledge about the private database, cannot compute a function of the output of a differentially private algorithm M and make it less differentially private. That is, a data analyst cannot increase privacy loss, either under the formal definition or even in any intuitive sense, simply by sitting in a corner and thinking about the output of the algorithm, *no matter what auxiliary information is available.*

These are the signal attributes of differential privacy. Can we prove a converse? That is, do these attributes, or some subset thereof, imply differential privacy? Can differential privacy be weakened in these respects and still be meaningful? These are open questions.

2.3.3 Final remarks on the definition

The Granularity of Privacy. Claims of differential privacy should be carefully scrutinized to ascertain the level of granularity at which privacy is being promised. Differential privacy promises that the behavior of an algorithm will be roughly unchanged even if a single entry in the database is modified. But what constitutes a single entry in the database? Consider for example a database that takes the form of a *graph.* Such a database might encode a social network: each individual $i \in [n]$ is represented by a vertex in the graph, and friendships between individuals are represented by edges.

We could consider differential privacy at a level of granularity corresponding to individuals: that is, we could require that differentially

private algorithms be insensitive to the addition or removal of any *vertex* from the graph. This gives a strong privacy guarantee, but might in fact be stronger than we need. the addition or removal of a single vertex could after all add or remove up to n edges in the graph. Depending on what it is we hope to learn from the graph, insensitivity to n edge removals might be an impossible constraint to meet.

We could on the other hand consider differential privacy at a level of granularity corresponding to edges, and ask our algorithms to be insensitive only to the addition or removal of single, or small numbers of, *edges* from the graph. This is of course a weaker guarantee, but might still be sufficient for some purposes. Informally speaking, if we promise ε-differential privacy at the level of a single edge, then no data analyst should be able to conclude anything about the existence of any subset of $1/\varepsilon$ edges in the graph. In some circumstances, large groups of social contacts might not be considered sensitive information: for example, an individual might not feel the need to hide the fact that the majority of his contacts are with individuals in his city or workplace, because where he lives and where he works are public information. On the other hand, there might be a small number of social contacts whose existence is highly sensitive (for example a prospective new employer, or an intimate friend). In this case, edge privacy should be sufficient to protect sensitive information, while still allowing a fuller analysis of the data than vertex privacy. Edge privacy will protect such an individual's sensitive information provided that he has fewer than $1/\varepsilon$ such friends.

As another example, a differentially private movie recommendation system can be designed to protect the data in the training set at the "event" level of single movies, hiding the viewing/rating of any single movie but not, say, hiding an individual's enthusiasm for cowboy westerns or gore, or at the "user" level of an individual's entire viewing and rating history.

All Small Epsilons Are Alike. When ε is small, $(\varepsilon, 0)$-differential privacy asserts that for all pairs of adjacent databases x, y and all outputs o, an adversary cannot distinguish which is the true database

on the basis of observing o. When ε is small, *failing* to be $(\varepsilon, 0)$-differentially private is not necessarily alarming — for example, the mechanism may be $(2\varepsilon, 0)$-differentially private. The nature of the privacy guarantees with differing but small epsilons are quite similar. But what of large values for ϵ? Failure to be $(15, 0)$-differentially private merely says there exist neighboring databases and an output o for which the ratio of probabilities of observing o conditioned on the database being, respectively, x or y, is large. An output of o might be very unlikely (this is addressed by (ε, δ)-differential privacy); databases x and y might be terribly contrived and ulikely to occur in the "real world"; the adversary may not have the right auxiliary information to recognize that a revealing output has occurred; or may not know enough about the database(s) to determine the value of their symmetric difference. Thus, much as a weak cryptosystem may leak anything from only the least significant bit of a message to the complete decryption key, the failure to be $(\varepsilon, 0)$- or (ε, δ)-differentially private may range from effectively meaningless privacy breaches to complete revelation of the entire database. A large epsilon is large after its own fashion.

A Few Additional Formalisms. Our privacy mechanism $\mathcal{M}$ will often take some auxiliary parameters w as input, in addition to the database x. For example, w may specify a query q_w on the database x, or a collection $\mathcal{Q}_w$ of queries. The mechanism $\mathcal{M}(w, x)$ might (respectively) respond with a differentially private approximation to $q_w(x)$ or to some or all of the queries in $\mathcal{Q}_w$. For all $\delta \geq 0$, we say that a mechanism $\mathcal{M}(\cdot, \cdot)$ satisfies (ε, δ)-differential privacy if for every w, $\mathcal{M}(w, \cdot)$ satisfies (ε, δ)-differential privacy.

Another example of a parameter that may be included in w is a *security parameter* κ to govern how small $\delta = \delta(\kappa)$ should be. That is, $\mathcal{M}(\kappa, \cdot)$ should be $(\varepsilon, \delta(\kappa))$ differentially private for all κ. Typically, and throughout this monograph, we require that δ be a negligible function in κ, i.e., $\delta = \kappa^{-\omega(1)}$. Thus, we think of δ as being cryptographically small, whereas ε is typically thought of as a moderately small constant.

In the case where the auxiliary parameter w specifies a collection $\mathcal{Q}_w = \{q : \mathcal{X}^n \to \mathbb{R}\}$ of queries, we call the mechanism $\mathcal{M}$ a

synopsis generator. A synopsis generator outputs a (differentially private) synopsis $\mathcal{A}$ which can be used to compute answers to all the queries in $\mathcal{Q}_w$. That is, we require that there exists a reconstruction procedure R such that for each input v specifying a query $q_v \in \mathcal{Q}_w$, the reconstruction procedure outputs $R(\mathcal{A}, v) \in \mathbb{R}$. Typically, we will require that with high probability $\mathcal{M}$ produces a synopsis $\mathcal{A}$ such that the reconstruction procedure, using $\mathcal{A}$, computes accurate answers. That is, for all or most (weighted by some distribution) of the queries $q_v \in \mathcal{Q}_w$, the error $|R(\mathcal{A}, v) - q_v(x)|$ will be bounded. We will occasionally abuse notation and refer to the reconstruction procedure taking as input the actual query q (rather than some representation v of it), and outputting $R(\mathcal{A}, q)$.

A special case of a synopsis is a *synthetic database*. As the name suggests, the rows of a synthetic database are of the same type as rows of the original database. An advantage to synthetic databases is that they may be analyzed using the same software that the analyst would use on the original database, obviating the need for a special reconstruction procedure R.

Remark 2.1. Considerable care must be taken when programming real-valued mechanisms, such as the Laplace mechanism, due to subtleties in the implementation of floating point numbers. Otherwise differential privacy can be destroyed, as outputs with non-zero probability on a database x, may, because of rounding, have zero probability on adjacent databases y. This is just one way in which the implementation of floating point requires scrutiny in the context of differential privacy, and it is not unique.

2.4 Bibliographic notes

The definition of differential privacy is due to Dwork et al. [23]; the precise formulation used here and in the literature first appears in [20] and is due to Dwork and McSherry. The term "differential privacy" was coined by Michael Schroeder. The impossibility of semantic security is due to Dwork and Naor [25]. Composition and group privacy for $(\varepsilon, 0)$-differentially private mechanisms is first addressed in [23].

Composition for (ε, δ)-differential privacy was first addressed in [21] (but see the corrected proof in Appendix B, due to Dwork and Lei [22]). The vulnerability of differential privacy to inappropriate implementations of floating point numbers was observed by Mironov, who proposed a mitigation [63].

3

Basic Techniques and Composition Theorems

After reviewing a few probabilistic tools, we present the Laplace mechanism, which gives differential privacy for real (vector) valued queries. An application of this leads naturally to the exponential mechanism, which is a method for differentially private selection from a discrete set of candidate outputs. We then analyze the cumulative privacy loss incurred by composing multiple differentially private mechanisms. Finally we give a method — the sparse vector technique — for privately reporting the outcomes of a potentially very large number of computations, provided that only a few are "significant."

In this section, we describe some of the most basic techniques in differential privacy that we will come back to use again and again. The techniques described here form the basic building blocks for all of the other algorithms that we will develop.

3.1 Useful probabilistic tools

The following concentration inequalities will frequently be useful. We state them in easy to use forms rather than in their strongest forms.

Theorem 3.1 (Additive Chernoff Bound)**.** Let $X_1, \ldots, X_m$ be independent random variables bounded such that $0 \leq X_i \leq 1$ for all i. Let $S = \frac{1}{m}\sum_{i=1}^{m} X_i$ denote their mean, and let $\mu = \mathbb{E}[S]$ denote their expected mean. Then:

$$\Pr[S > \mu + \varepsilon] \leq e^{-2m\varepsilon^2}$$

$$\Pr[S < \mu - \varepsilon] \leq e^{-2m\varepsilon^2}$$

Theorem 3.2 (Multiplicative Chernoff Bound)**.** Let $X_1, \ldots, X_m$ be independent random variables bounded such that $0 \leq X_i \leq 1$ for all i. Let $S = \frac{1}{m}\sum_{i=1}^{m} X_i$ denote their mean, and let $\mu = \mathbb{E}[S]$ denote their expected mean. Then:

$$\Pr[S > (1+\varepsilon)\mu] \leq e^{-m\mu\varepsilon^2/3}$$

$$\Pr[S < (1-\varepsilon)\mu] \leq e^{-m\mu\varepsilon^2/2}$$

When we do not have independent random variables, all is not lost. We may still apply Azuma's inequality:

Theorem 3.3 (Azuma's Inequality)**.** Let f be a function of m random variables $X_1, \ldots, X_m$, each X_i taking values from a set A_i such that $\mathbb{E}[f]$ is bounded. Let c_i denote the maximum effect of X_i on f — i.e., for all $a_i, a_i' \in A_i$:

$$\left|\mathbb{E}[f|X_1, \ldots, X_{i-1}, X_i = a_i] - \mathbb{E}[f|X_1, \ldots, X_{i-1}, X_i = a_i']\right| \leq c_i$$

Then:

$$\Pr\left[f(X_1, \ldots, X_m) \geq \mathbb{E}[f] + t\right] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{m} c_i^2}\right)$$

Theorem 3.4 (Stirling's Approximation)**.** $n!$ can be approximated by $\sqrt{2n\pi}(n/e)^n$:

$$\sqrt{2n\pi}(n/e)^n e^{1/(12n+1)} < n! < \sqrt{2n\pi}(n/e)^n e^{1/(12n)}.$$

3.2 Randomized response

Let us recall the simple randomized response mechanism, described in Section 2, for evaluating the frequency of embarrassing or illegal

behaviors. Let XYZ be such an activity. Faced with the query, "Have you engaged in XYZ in the past week?" the respondent is instructed to perform the following steps:

1. Flip a coin.
2. If **tails**, then respond truthfully.
3. If **heads**, then flip a second coin and respond "Yes" if heads and "No" if tails.

The intuition behind randomized response is that it provides "plausible deniability." For example, a response of "Yes" may have been offered because the first and second coin flips were both Heads, which occurs with probability 1/4. In other words, *privacy is obtained by process*, there are no "good" or "bad" responses. The process by which the responses are obtained affects how they may legitimately be interpreted. As the next claim shows, randomized response is differentially private.

Claim 3.5. The version of randomized response described above is $(\ln 3, 0)$-differentially private.

Proof. Fix a respondent. A case analysis shows that $\Pr[\text{Response} = \text{Yes}|\text{Truth} = \text{Yes}] = 3/4$. Specifically, when the truth is "Yes" the outcome will be "Yes" if the first coin comes up tails (probability 1/2) or the first and second come up heads (probability 1/4)), while $\Pr[\text{Response} = \text{Yes}|\text{Truth} = \text{No}] = 1/4$ (first comes up heads and second comes up tails; probability 1/4). Applying similar reasoning to the case of a "No" answer, we obtain:

$$\frac{\Pr[\text{Response} = \text{Yes}|\text{Truth} = \text{Yes}]}{\Pr[\text{Response} = \text{Yes}|\text{Truth} = \text{No}]} = \frac{3/4}{1/4} = \frac{\Pr[\text{Response} = \text{No}|\text{Truth} = \text{No}]}{\Pr[\text{Response} = \text{No}|\text{Truth} = \text{Yes}]} = 3.$$

□

3.3 The laplace mechanism

Numeric queries, functions $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, are one of the most fundamental types of database queries. These queries map databases to k

real numbers. One of the important parameters that will determine just how accurately we can answer such queries is their ℓ_1 sensitivity:

Definition 3.1 (ℓ_1-sensitivity)**.** The ℓ_1-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is:

$$\Delta f = \max_{\substack{x,y \in \mathbb{N}^{|\mathcal{X}|} \\ \|x-y\|_1 = 1}} \|f(x) - f(y)\|_1.$$

The ℓ_1 sensitivity of a function f captures the magnitude by which a single individual's data can change the function f in the worst case, and therefore, intuitively, the uncertainty in the response that we must introduce in order to hide the participation of a single individual. Indeed, we will formalize this intuition: the sensitivity of a function gives an upper bound on how much we must perturb its output to preserve privacy. One noise distribution naturally lends itself to differential privacy.

Definition 3.2 (The Laplace Distribution)**.** The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function:

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

The variance of this distribution is $\sigma^2 = 2b^2$. We will sometimes write $\text{Lap}(b)$ to denote the Laplace distribution with scale b, and will sometimes abuse notation and write $\text{Lap}(b)$ simply to denote a random variable $X \sim \text{Lap}(b)$.

The Laplace distribution is a symmetric version of the exponential distribution.

We will now define the *Laplace Mechanism*. As its name suggests, the Laplace mechanism will simply compute f, and perturb each coordinate with noise drawn from the Laplace distribution. The scale of the noise will be calibrated to the sensitivity of f (divided by ε).[1]

[1] Alternately, using Gaussian noise with variance calibrated to $\Delta f \ln(1/\delta)/\varepsilon$, one can achieve (ε, δ)-differential privacy (see Appendix A). Use of the Laplace mechanism is cleaner and the two mechanisms behave similarly under composition (Theorem 3.20).

Definition 3.3 (The Laplace Mechanism)**.** Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$\mathcal{M}_L(x, f(\cdot), \varepsilon) = f(x) + (Y_1, \ldots, Y_k)$$

where Y_i are i.i.d. random variables drawn from $\mathrm{Lap}(\Delta f/\varepsilon)$.

Theorem 3.6. The Laplace mechanism preserves $(\varepsilon, 0)$-differential privacy.

Proof. Let $x \in \mathbb{N}^{|\mathcal{X}|}$ and $y \in \mathbb{N}^{|\mathcal{X}|}$ be such that $\|x - y\|_1 \leq 1$, and let $f(\cdot)$ be some function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$. Let p_x denote the probability density function of $\mathcal{M}_L(x, f, \varepsilon)$, and let p_y denote the probability density function of $\mathcal{M}_L(y, f, \varepsilon)$. We compare the two at some arbitrary point $z \in \mathbb{R}^k$

$$\begin{aligned}
\frac{p_x(z)}{p_y(z)} &= \prod_{i=1}^{k} \left(\frac{\exp(-\frac{\varepsilon|f(x)_i - z_i|}{\Delta f})}{\exp(-\frac{\varepsilon|f(y)_i - z_i|}{\Delta f})} \right) \\
&= \prod_{i=1}^{k} \exp\left(\frac{\varepsilon(|f(y)_i - z_i| - |f(x)_i - z_i|)}{\Delta f} \right) \\
&\leq \prod_{i=1}^{k} \exp\left(\frac{\varepsilon|f(x)_i - f(y)_i|}{\Delta f} \right) \\
&= \exp\left(\frac{\varepsilon \cdot \|f(x) - f(y)\|_1}{\Delta f} \right) \\
&\leq \exp(\varepsilon),
\end{aligned}$$

where the first inequality follows from the triangle inequality, and the last follows from the definition of sensitivity and the fact that $\|x - y\|_1 \leq 1$. That $\frac{p_x(z)}{p_y(z)} \geq \exp(-\varepsilon)$ follows by symmetry. □

Example 3.1 (Counting Queries)**.** Counting queries are queries of the form "How many elements in the database satisfy Property P?" We will return to these queries again and again, sometimes in this pure form, sometimes in fractional form ("What fraction of the elements in the databases...?"), sometimes with weights (linear queries), and sometimes in slightly more complex forms (e.g., apply $h : \mathbb{N}^{|\mathcal{X}|} \to [0, 1]$ to each element in the database and sum the results). Counting is an

extremely powerful primitive. It captures everything learnable in the statistical queries learning model, as well as many standard datamining tasks and basic statistics. Since the sensitivity of a counting query is 1 (the addition or deletion of a single individual can change a count by at most 1), it is an immediate consequence of Theorem 3.6 that $(\varepsilon, 0)$-differential privacy can be achieved for counting queries by the addition of noise scaled to $1/\varepsilon$, that is, by adding noise drawn from $\mathrm{Lap}(1/\varepsilon)$. The expected distortion, or error, is $1/\varepsilon$, independent of the size of the database.

A fixed but arbitrary list of m counting queries can be viewed as a vector-valued query. Absent any further information about the set of queries a worst-case bound on the sensitivity of this vector-valued query is m, as a single individual might change every count. In this case $(\varepsilon, 0)$-differential privacy can be achieved by adding noise scaled to m/ε to the true answer to each query.

We sometimes refer to the problem of responding to large numbers of (possibly arbitrary) queries as the *query release problem*.

Example 3.2 (Histogram Queries)**.** In the special (but common) case in which the queries are structurally disjoint we can do much better — we don't necessarily have to let the noise scale with the number of queries. An example is the *histogram query*. In this type of query the universe $\mathbb{N}^{|\mathcal{X}|}$ is partitioned into cells, and the query asks how many database elements lie in each of the cells. Because the cells are disjoint, the addition or removal of a single database element can affect the count in exactly one cell, and the difference to that cell is bounded by 1, so histogram queries have sensitivity 1 and can be answered by adding independent draws from $\mathrm{Lap}(1/\varepsilon)$ to the true count in each cell.

To understand the accuracy of the Laplace mechanism for general queries we use the following useful fact:

Fact 3.7. If $Y \sim \mathrm{Lap}(b)$, then:

$$\Pr[|Y| \geq t \cdot b] = \exp(-t).$$

This fact, together with a union bound, gives us a simple bound on the accuracy of the Laplace mechanism:

Theorem 3.8. Let $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, and let $y = \mathcal{M}_L(x, f(\cdot), \varepsilon)$. Then $\forall \delta \in (0, 1]$:

$$\Pr\left[\|f(x) - y\|_\infty \geq \ln\left(\frac{k}{\delta}\right) \cdot \left(\frac{\Delta f}{\varepsilon}\right)\right] \leq \delta$$

Proof. We have:

$$\begin{aligned}
\Pr\left[\|f(x) - y\|_\infty \geq \ln\left(\frac{k}{\delta}\right) \cdot \left(\frac{\Delta f}{\varepsilon}\right)\right] &= \Pr\left[\max_{i \in [k]} |Y_i| \geq \ln\left(\frac{k}{\delta}\right) \cdot \left(\frac{\Delta f}{\varepsilon}\right)\right] \\
&\leq k \cdot \Pr\left[|Y_i| \geq \ln\left(\frac{k}{\delta}\right) \cdot \left(\frac{\Delta f}{\varepsilon}\right)\right] \\
&= k \cdot \left(\frac{\delta}{k}\right) \\
&= \delta
\end{aligned}$$

where the second to last inequality follows from the fact that each $Y_i \sim \text{Lap}(\Delta f/\varepsilon)$ and Fact 3.7. □

Example 3.3 (First Names)**.** Suppose we wish to calculate which first names, from a list of 10,000 potential names, were the most common among participants of the 2010 census. This question can be represented as a query $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^{10000}$. This is a histogram query, and so has sensitivity $\Delta f = 1$, since every person can only have at most one first name. Using the above theorem, we see that we can simultaneously calculate the frequency of all $10,000$ names with $(1, 0)$-differential privacy, and with probability 95%, no estimate will be off by more than an additive error of $ln(10000/.05) \approx 12.2$. That's pretty low error for a nation of more than $300,000,000$ people!

Differentially Private Selection. The task in Example 3.3 is one of *differentially private selection*: the space of outcomes is discrete and the task is to produce a "best" answer, in this case the most populous histogram cell.

Example 3.4 (Most Common Medical Condition)**.** Suppose we wish to know which condition is (approximately) the most common in the medical histories of a set of respondents, so the set of questions is, for each condition under consideration, whether the individual has ever received a diagnosis of this condition. Since individuals can experience many conditions, the sensitivity of this set of questions can be high. Nonetheless, as we next describe, this task can be addressed using addition of $\mathrm{Lap}(1/\varepsilon)$ noise to each of the counts (note the small scale of the noise, which is independent of the total number of conditions). Crucially, the m noisy counts themselves will *not* be released (although the "winning" count can be released at no extra privacy cost).

Report Noisy Max. Consider the following simple algorithm to determine which of m counting queries has the highest value: Add independently generated Laplace noise $\mathrm{Lap}(1/\varepsilon)$ to each count and return the index of the largest noisy count (we ignore the possibility of a tie). Call this algorithm Report Noisy Max.

Note the "information minimization" principle at work in the Report Noisy Max algorithm: rather than releasing all the noisy counts and allowing the analyst to find the max and its index, only the index corresponding to the maximum is made public. Since the data of an individual can affect all counts, the vector of counts has high ℓ_1-sensitivity, specifically, $\Delta f = m$, and much more noise would be needed if we wanted to release all of the counts using the Laplace mechanism.

Claim 3.9. The Report Noisy Max algorithm is $(\varepsilon, 0)$-differentially private.

Proof. Fix $D = D' \cup \{a\}$. Let c, respectively c', denote the vector of counts when the database is D, respectively D'. We use two properties:

1. *Monotonicity of Counts.* For all $j \in [m]$, $c_j \geq c'_j$; and
2. *Lipschitz Property.* For all $j \in [m]$, $1 + c'_j \geq c_j$.

Fix any $i \in [m]$. We will bound from above and below the ratio of the probabilities that i is selected with D and with D'.

Fix r_{-i}, a draw from $[\mathrm{Lap}(1/\varepsilon)]^{m-1}$ used for all the noisy counts except the ith count. We will argue for each r_{-i} independently. We

use the notation $\Pr[i|\xi]$ to mean the probability that the output of the Report Noisy Max algorithm is i, conditioned on ξ.

We first argue that $\Pr[i|D, r_{-i}] \leq e^{\varepsilon} \Pr[i|D', r_{-i}]$. Define

$$r^* = \min_{r_i} : c_i + r_i > c_j + r_j \ \forall j \neq i.$$

Note that, having fixed r_{-i}, i will be the output (the argmax noisy count) when the database is D if and only if $r_i \geq r^*$.
We have, for all $1 \leq j \neq i \leq m$:

$$\begin{aligned} c_i + r^* &> c_j + r_j \\ \Rightarrow (1 + c'_i) + r^* \geq c_i + r^* &> c_j + r_j \geq c'_j + r_j \\ \Rightarrow c'_i + (r^* + 1) &> c'_j + r_j. \end{aligned}$$

Thus, if $r_i \geq r^* + 1$, then the ith count will be the maximum when the database is D' and the noise vector is (r_i, r_{-i}). The probabilities below are over the choice of $r_i \sim \mathrm{Lap}(1/\varepsilon)$.

$$\begin{aligned} &\Pr[r_i \geq 1 + r^*] \geq e^{-\varepsilon} \Pr[r_i \geq r^*] = e^{-\varepsilon} \Pr[i|D, r_{-i}] \\ \Rightarrow &\Pr[i|D', r_{-i}] \geq \Pr[r_i \geq 1 + r^*] \geq e^{-\varepsilon} \Pr[r_i \geq r^*] = e^{-\varepsilon} \Pr[i|D, r_{-i}], \end{aligned}$$

which, after multiplying through by e^{ε}, yields what we wanted to show: $\Pr[i|D, r_{-i}] \leq e^{\varepsilon} \Pr[i|D', r_{-i}]$.

We now argue that $\Pr[i|D', r_{-i}] \leq e^{\varepsilon} \Pr[i|D, r_{-i}]$. Define

$$r^* = \min_{r_i} : c'_i + r_i > c'_j + r_j \ \forall j \neq i.$$

Note that, having fixed r_{-i}, i will be the output (argmax noisy count) when the database is D' if and only if $r_i \geq r^*$.
We have, for all $1 \leq j \neq i \leq m$:

$$\begin{aligned} c'_i + r^* &> c'_j + r_j \\ \Rightarrow 1 + c'_i + r^* &> 1 + c'_j + r_j \\ \Rightarrow c'_i + (r^* + 1) &> (1 + c'_j) + r_j \\ \Rightarrow c_i + (r^* + 1) \geq c'_i + (r^* + 1) &> (1 + c'_j) + r_j \geq c_j + r_j. \end{aligned}$$

Thus, if $r_i \geq r^* + 1$, then i will be the output (the argmax noisy count) on database D with randomness (r_i, r_{-i}). We therefore have, with probabilities taken over choice of r_i:

$$\Pr[i|D, r_{-i}] \geq \Pr[r_i \geq r^* + 1] \geq e^{-\varepsilon} \Pr[r_i \geq r^*] = e^{-\varepsilon} \Pr[i|D', r_{-i}],$$

which, after multiplying through by e^{ε}, yields what we wanted to show: $\Pr[i|D', r_{-i}] \leq e^{\varepsilon} \Pr[i|D, r_{-i}]$. □

3.4 The exponential mechanism

In both the "most common name" and "most common condition" examples the "utility" of a response (name or medical condition, respectively) we estimated counts using Laplace noise and reported the noisy maximum. In both examples the utility of the response is directly related to the noise values generated; that is, the popularity of the name or condition is appropriately measured on the same scale and in the same units as the magnitude of the noise.

The *exponential mechanism* was designed for situations in which we wish to choose the "best" response but adding noise directly to the computed quantity can completely destroy its value, such as setting a price in an auction, where the goal is to maximize revenue, and adding a small amount of positive noise to the optimal price (in order to protect the privacy of a bid) could dramatically reduce the resulting revenue.

Example 3.5 (Pumpkins.)**.** Suppose we have an abundant supply of pumpkins and four bidders: A, F, I, K, where A, F, I each bid \$1.00 and K bids \$3.01. What is the optimal price? At \$3.01 the revenue is \$3.01, at \$3.00 and at \$1.00 the revenue is \$3.00, but at \$3.02 the revenue is zero!

The exponential mechanism is the natural building block for answering queries with arbitrary utilities (and arbitrary non-numeric range), while preserving differential privacy. Given some arbitrary range $\mathcal{R}$, the exponential mechanism is defined with respect to some utility function $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$, which maps database/output pairs to utility scores. Intuitively, for a fixed database x, the user prefers that the mechanism outputs some element of $\mathcal{R}$ with the maximum possible utility score. Note that when we talk about the sensitivity of the utility score $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$, we care only about the sensitivity of u with respect to its database argument; it can be arbitrarily sensitive in its

range argument:

$$\Delta u \equiv \max_{r \in \mathcal{R}} \max_{x,y:\|x-y\|_1 \leq 1} |u(x,r) - u(y,r)|.$$

The intuition behind the exponential mechanism is to output each possible $r \in \mathcal{R}$ with probability proportional to $\exp(\varepsilon u(x,r)/\Delta u)$ and so the privacy loss is approximately:

$$\ln\left(\frac{\exp(\varepsilon u(x,r)/\Delta u)}{\exp(\varepsilon u(y,r)/\Delta u)}\right) = \varepsilon[u(x,r) - u(y,r)]/\Delta u) \leq \varepsilon.$$

This intuitive view overlooks some effects of a normalization term which arises when an additional person in the database causes the utilities of some elements $r \in \mathcal{R}$ to decrease and others to increase. The actual mechanism, defined next, reserves half the privacy budget for changes in the normalization term.

Definition 3.4 (The Exponential Mechanism)**.** The exponential mechanism $\mathcal{M}_E(x, u, \mathcal{R})$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp(\frac{\varepsilon u(x,r)}{2\Delta u})$.

The exponential mechanism can define a complex distribution over a large arbitrary domain, and so it may not be possible to implement the exponential mechanism efficiently when the range of u is super-polynomially large in the natural parameters of the problem.

Returning to the pumpkin example, utility for a price p on database x is simply the profit obtained when the price is p and the demand curve is as described by x. It is important that the range of *potential* prices is independent of the actual bids. Otherwise there would exist a price with non-zero weight in one dataset and zero weight in a neighboring set, violating differential privacy.

Theorem 3.10. The exponential mechanism preserves $(\varepsilon, 0)$-differential privacy.

Proof. For clarity, we assume the range $\mathcal{R}$ of the exponential mechanism is finite, but this is not necessary. As in all differential privacy proofs, we consider the ratio of the probability that an instantiation

of the exponential mechanism outputs some element $r \in \mathcal{R}$ on two neighboring databases $x \in \mathbb{N}^{|\mathcal{X}|}$ and $y \in \mathbb{N}^{|\mathcal{X}|}$ (i.e., $\|x - y\|_1 \leq 1$).

$$\begin{aligned}
\frac{\Pr[\mathcal{M}_E(x, u, \mathcal{R}) = r]}{\Pr[\mathcal{M}_E(y, u, \mathcal{R}) = r]} &= \frac{\left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right)}{\left(\frac{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(y,r')}{2\Delta u})}\right)} \\
&= \left(\frac{\exp(\frac{\varepsilon u(x,r)}{2\Delta u})}{\exp(\frac{\varepsilon u(y,r)}{2\Delta u})}\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right) \\
&= \exp\left(\frac{\varepsilon(u(x, r') - u(y, r'))}{2\Delta u}\right) \\
&\quad \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(y,r')}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right) \\
&\leq \exp\left(\frac{\varepsilon}{2}\right) \cdot \exp\left(\frac{\varepsilon}{2}\right) \cdot \left(\frac{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}{\sum_{r' \in \mathcal{R}} \exp(\frac{\varepsilon u(x,r')}{2\Delta u})}\right) \\
&= \exp(\varepsilon).
\end{aligned}$$

Similarly, $\frac{\Pr[\mathcal{M}_E(y,u)=r]}{\Pr[\mathcal{M}_E(x,u)=r]} \geq \exp(-\varepsilon)$ by symmetry. □

The exponential mechanism can often give strong utility guarantees, because it discounts outcomes exponentially quickly as their quality score falls off. For a given database x and a given utility measure $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$, let $\mathrm{OPT}_u(x) = \max_{r \in \mathcal{R}} u(x, r)$ denote the maximum utility score of any element $r \in \mathcal{R}$ with respect to database x. We will bound the probability that the exponential mechanism returns a "good" element of $\mathcal{R}$, where good will be measured in terms of $\mathrm{OPT}_u(x)$. The result is that it will be highly unlikely that the returned element r has a utility score that is inferior to $\mathrm{OPT}_u(x)$ by more than an additive factor of $O((\Delta u/\varepsilon) \log |\mathcal{R}|)$.

Theorem 3.11. Fixing a database x, let $\mathcal{R}_{\mathrm{OPT}} = \{r \in \mathcal{R} : u(x, r) = \mathrm{OPT}_u(x)\}$ denote the set of elements in $\mathcal{R}$ which attain utility score

$\mathrm{OPT}_u(x)$. Then:

$$\Pr\left[u(\mathcal{M}_E(x,u,\mathcal{R})) \leq \mathrm{OPT}_u(x) - \frac{2\Delta u}{\varepsilon}\left(\ln\left(\frac{|\mathcal{R}|}{|\mathcal{R}_{\mathrm{OPT}}|}\right) + t\right)\right] \leq e^{-t}$$

Proof.

$$\begin{aligned}\Pr[u(\mathcal{M}_E(x,u,\mathcal{R})) \leq c] &\leq \frac{|\mathcal{R}|\exp(\varepsilon c/2\Delta u)}{|\mathcal{R}_{\mathrm{OPT}}|\exp(\varepsilon \mathrm{OPT}_u(x)/2\Delta u)} \\ &= \frac{|\mathcal{R}|}{|\mathcal{R}_{\mathrm{OPT}}|}\exp\left(\frac{\varepsilon(c - \mathrm{OPT}_u(x))}{2\Delta u}\right).\end{aligned}$$

The inequality follows from the observation that each $r \in \mathcal{R}$ with $u(x,r) \leq c$ has un-normalized probability mass at most $\exp(\varepsilon c/2\Delta u)$, and hence the entire set of such "bad" elements r has total un-normalized probability mass at most $|\mathcal{R}|\exp(\varepsilon c/2\Delta u)$. In contrast, we know that there exist at least $|\mathcal{R}_{\mathrm{OPT}}| \geq 1$ elements with $u(x,r) = \mathrm{OPT}_u(x)$, and hence un-normalized probability mass $\exp(\varepsilon \mathrm{OPT}_u(x)/2\Delta u)$, and so this is a lower bound on the normalization term.

The theorem follows from plugging in the appropriate value for c. □

Since we always have $|\mathcal{R}_{\mathrm{OPT}}| \geq 1$, we can more commonly make use of the following simple corollary:

Corollary 3.12. Fixing a database x, we have:

$$\Pr\left[u(\mathcal{M}_E(x,u,\mathcal{R})) \leq \mathrm{OPT}_u(x) - \frac{2\Delta u}{\varepsilon}\left(\ln\left(|\mathcal{R}|\right) + t\right)\right] \leq e^{-t}$$

As seen in the proofs of Theorem 3.11 and Corollary 3.12, the Exponential Mechanism can be particularly easy to analyze.

Example 3.6 (Best of Two)**.** Consider the simple question of determining which of exactly two medical conditions A and B is more common. Let the two true counts be 0 for condition A and $c > 0$ for condition B. Our notion of utility will be tied to the actual counts, so that conditions with bigger counts have higher utility and $\Delta u = 1$. Thus, the utility of A is 0 and the utility of B is c. Using the Exponential Mechanism

we can immediately apply Corollary 3.12 to see that the probability of observing (wrong) outcome A is at most $2e^{-c(\varepsilon/(2\Delta u))} = 2e^{-c\varepsilon/2}$.

Analyzing Report Noisy Max appears to be more complicated, as it requires understanding what happens in the (probability 1/4) case when the noise added to the count for A is positive and the noise added to the count for B is negative.

A function is *monotonic in the data set* if the addition of an element to the data set cannot cause the value of the function to decrease. Counting queries are monotonic; so is the revenue obtained by offering a fixed price to a collection of buyers.

Consider the *Report One-Sided Noisy Arg-Max* mechanism, which adds noise to the *utility* of each potential output drawn from the *one-sided* exponential distribution with parameter $\varepsilon/\Delta u$ in the case of a monotonic utility, or parameter $\varepsilon/2\Delta u$ for the case of a non-monotonic utility, and reports the resulting arg-max.

With this algorithm, whose privacy proof is almost identical to that of Report Noisy Max (but loses a factor of two when the utility is non-monotonic), we immediately obtain in Example 3.6 above that outcome A is exponentially in $c(\varepsilon/\Delta u) = c\varepsilon$ less likely to be selected than outcome B.

Theorem 3.13. Report One-Sided Noisy Arg-Max, when run with parameter $\varepsilon/2\Delta u$ is ϵ-differentially private.

Remark 3.1. Report noisy max when instantiated with Laplace noise or exponential noise both have similar guarantees to the exponential mechanism, but lead to distinct distributions. It turns out that instantiating report noisy max with the Gumbel distribution leads to an algorithm that samples *exactly* from the exponential mechanism distribution. This fact is folklore in machine learning, and known as the "Gumbel Max Trick".

3.5 Composition theorems

Now that we have several building blocks for designing differentially private algorithms, it is important to understand how we can combine

them to design more sophisticated algorithms. In order to use these tools, we would like that the combination of two differentially private algorithms be differentially private itself. Indeed, as we will see, this is the case. Of course the parameters ε and δ will necessarily degrade — consider repeatedly computing the same statistic using the Laplace mechanism, scaled to give ε-differential privacy each time. The average of the answer given by each instance of the mechanism will eventually converge to the true value of the statistic, and so we cannot avoid that the strength of our privacy guarantee will degrade with repeated use. In this section we give theorems showing how exactly the parameters ε and δ compose when differentially private subroutines are combined.

Let us first begin with an easy warm up: we will see that the independent use of an $(\varepsilon_1, 0)$-differentially private algorithm and an $(\varepsilon_2, 0)$-differentially private algorithm, when taken together, is $(\varepsilon_1 + \varepsilon_2, 0)$-differentially private.

Theorem 3.14. Let $\mathcal{M}_1 : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_1$ be an ε_1-differentially private algorithm, and let $\mathcal{M}_2 : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_2$ be an ε_2-differentially private algorithm. Then their combination, defined to be $\mathcal{M}_{1,2} : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_1 \times \mathcal{R}_2$ by the mapping: $\mathcal{M}_{1,2}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $\varepsilon_1 + \varepsilon_2$-differentially private.

Proof. Let $x, y \in \mathbb{N}^{|\mathcal{X}|}$ be such that $\|x - y\|_1 \leq 1$. Fix any $(r_1, r_2) \in \mathcal{R}_1 \times \mathcal{R}_2$. Then:

$$\begin{aligned}
\frac{\Pr[\mathcal{M}_{1,2}(x) = (r_1, r_2)]}{\Pr[\mathcal{M}_{1,2}(y) = (r_1, r_2)]} &= \frac{\Pr[\mathcal{M}_1(x) = r_1] \Pr[\mathcal{M}_2(x) = r_2]}{\Pr[\mathcal{M}_1(y) = r_1] \Pr[\mathcal{M}_2(y) = r_2]} \\
&= \left(\frac{\Pr[\mathcal{M}_1(x) = r_1]}{\Pr[\mathcal{M}_1(y) = r_1]}\right)\left(\frac{\Pr[\mathcal{M}_2(x) = r_1]}{\Pr[\mathcal{M}_2(y) = r_1]}\right) \\
&\leq \exp(\varepsilon_1)\exp(\varepsilon_2) \\
&= \exp(\varepsilon_1 + \varepsilon_2)
\end{aligned}$$

By symmetry, $\frac{\Pr[\mathcal{M}_{1,2}(x)=(r_1,r_2)]}{\Pr[\mathcal{M}_{1,2}(y)=(r_1,r_2)]} \geq \exp(-(\varepsilon_1 + \varepsilon_2))$. □

The composition theorem can be applied repeatedly to obtain the following corollary:

Corollary 3.15. Let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_i$ be an $(\varepsilon_i, 0)$-differentially private algorithm for $i \in [k]$. Then if $\mathcal{M}_{[k]} : \mathbb{N}^{|\mathcal{X}|} \to \prod_{i=1}^{k} \mathcal{R}_i$ is defined to be $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$, then $\mathcal{M}_{[k]}$ is $(\sum_{i=1}^{k} \varepsilon_i, 0)$-differentially private.

A proof of the generalization of this theorem to (ε, δ)-differential privacy appears in Appendix B:

Theorem 3.16. Let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_i$ be an $(\varepsilon_i, \delta_i)$-differentially private algorithm for $i \in [k]$. Then if $\mathcal{M}_{[k]} : \mathbb{N}^{|\mathcal{X}|} \to \prod_{i=1}^{k} \mathcal{R}_i$ is defined to be $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$, then $\mathcal{M}_{[k]}$ is $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$-differentially private.

It is a strength of differential privacy that composition is "automatic," in that the bounds obtained hold without any special effort by the database curator.

3.5.1 Composition: some technicalities

In the remainder of this section, we will prove a more sophisticated composition theorem. To this end, we will need some definitions and lemmas, rephrasing differential privacy in terms of distance measures between distributions. In the fractional quantities below, if the denominator is zero, then we define the value of the fraction to be infinite (the numerators will always be positive).

Definition 3.5 (KL-Divergence)**.** The KL-Divergence, or Relative Entropy, between two random variables Y and Z taking values from the same domain is defined to be:

$$D(Y\|Z) = \mathbb{E}_{y \sim Y}\left[\ln \frac{\Pr[Y = y]}{\Pr[Z = y]}\right].$$

It is known that $D(Y\|Z) \geq 0$, with equality if and only if Y and Z are identically distributed. However, D is not symmetric, does not satisfy the triangle inequality, and can even be infinite, specifically when $\mathrm{Supp}(Y)$ is not contained in $\mathrm{Supp}(Z)$.

Definition 3.6 (Max Divergence)**.** The Max Divergence between two random variables Y and Z taking values from the same domain is

defined to be:

$$D_\infty(Y\|Z) = \max_{S\subseteq \mathrm{Supp}(Y)} \left[\ln \frac{\Pr[Y\in S]}{\Pr[Z\in S]}\right].$$

The δ-Approximate Max Divergence between Y and Z is defined to be:

$$D_\infty^\delta(Y\|Z) = \max_{S\subseteq \mathrm{Supp}(Y):\Pr[Y\in S]\geq\delta} \left[\ln \frac{\Pr[Y\in S]-\delta}{\Pr[Z\in S]}\right]$$

Remark 3.2. Note that a mechanism $\mathcal{M}$ is

1. ε-differentially private if and only if on every two neighboring databases x and y, $D_\infty(\mathcal{M}(x)\|\mathcal{M}(y)) \leq \varepsilon$ and $D_\infty(\mathcal{M}(y)\|\mathcal{M}(x)) \leq \varepsilon$; and is
2. (ε,δ)-differentially private if and only if on every two neighboring databases x, y: $D_\infty^\delta(\mathcal{M}(x)\|\mathcal{M}(y)) \leq \varepsilon$ and $D_\infty^\delta(\mathcal{M}(y)\|\mathcal{M}(x)) \leq \varepsilon$.

One other distance measure that will be useful is the *statistical distance* between two random variables Y and Z, defined as

$$\Delta(Y,Z) \stackrel{\mathrm{def}}{=} \max_S |\Pr[Y\in S] - \Pr[Z\in S]|.$$

We say that Y and Z are δ-*close* if $\Delta(Y,Z)\leq\delta$.

We will use the following reformulations of approximate max-divergence in terms of exact max-divergence and statistical distance:

Lemma 3.17.

1. $D_\infty^\delta(Y\|Z)\leq\varepsilon$ if and only if there exists a random variable Y' such that $\Delta(Y,Y')\leq\delta$ and $D_\infty(Y'\|Z)\leq\varepsilon$.
2. We have both $D_\infty^\delta(Y\|Z)\leq\varepsilon$ and $D_\infty^\delta(Z\|Y)\leq\varepsilon$ if and only if there exist random variables Y', Z' such that $\Delta(Y,Y')\leq\delta/(e^\varepsilon+1)$, $\Delta(Z,Z')\leq\delta/(e^\varepsilon+1)$, and $D_\infty(Y'\|Z')\leq\varepsilon$.

Proof. For Part 1, suppose there exists Y' δ-close to Y such that $D_\infty(Y\|Z)\leq\varepsilon$. Then for every S,

$$\Pr[Y\in S] \leq \Pr[Y'\in S] + \delta \leq e^\varepsilon\cdot\Pr[Z\in S] + \delta,$$

and thus $D_\infty^\delta(Y\|Z)\leq\varepsilon$.

Conversely, suppose that $D_\infty^\delta(Y\|Z) \le \varepsilon$. Let $S = \{y : \Pr[Y = y] > e^\varepsilon \cdot \Pr[Z = y]\}$. Then

$$\sum_{y \in S} (\Pr[Y = y] - e^\varepsilon \cdot \Pr[Z = y]) = \Pr[Y \in S] - e^\varepsilon \cdot \Pr[Z \in S] \le \delta.$$

Moreover, if we let $T = \{y : \Pr[Y = y] < \Pr[Z = y]\}$, then we have

$$\begin{aligned}\sum_{y \in T} (\Pr[Z = y] - \Pr[Y = y]) &= \sum_{y \notin T} (\Pr[Y = y] - \Pr[Z = y]) \\ &\ge \sum_{y \in S} (\Pr[Y = y] - \Pr[Z = y]) \\ &\ge \sum_{y \in S} (\Pr[Y = y] - e^\varepsilon \cdot \Pr[Z = y])/\end{aligned}$$

Thus, we can obtain Y' from Y by lowering the probabilities on S and raising the probabilities on T to satisfy:

1. For all $y \in S$, $\Pr[Y' = y] = e^\varepsilon \cdot \Pr[Z = y] < \Pr[Y = y]$.
2. For all $y \in T$, $\Pr[Y = y] \le \Pr[Y' = y] \le \Pr[Z = y]$.
3. For all $y \notin S \cup T$, $\Pr[Y' = y] = \Pr[Y = y] \le e^\varepsilon \cdot \Pr[Z = y]$.

Then $D_\infty(Y'\|Z) \le \varepsilon$ by inspection, and

$$\Delta(Y, Y') = \Pr[Y \in S] - \Pr[Y' \in S] = \Pr[Y \in S] - e^\varepsilon \cdot \Pr[Z \in S] \le \delta.$$

We now prove Part 2. Suppose there exist random variables Y' and Z' as stated. Then, for every set S,

$$\begin{aligned}\Pr[Y \in S] &\le \Pr[Y' \in S] + \frac{\delta}{e^\varepsilon + 1} \\ &\le e^\varepsilon \cdot \Pr[Z' \in S] + \frac{\delta}{e^\varepsilon + 1} \\ &\le e^\varepsilon \cdot \left(\Pr[Z \in S] + \frac{\delta}{e^\varepsilon + 1}\right) + \frac{\delta}{e^\varepsilon + 1} \\ &= e^\varepsilon \cdot \Pr[Z \in S] + \delta.\end{aligned}$$

Thus $D_\infty^\delta(Y\|Z) \le \varepsilon$, and by symmetry, $D_\infty^\delta(Z\|Y) \le \varepsilon$.

Conversely, given Y and Z such that $D_\infty^\delta(Y\|Z) \le \varepsilon$ and $D_\infty^\delta(Z\|Y) \le \varepsilon$, we proceed similarly to Part 1. However, instead of simply decreasing the probability mass of Y on S to obtain Y' and

eliminate the gap with $e^\varepsilon \cdot Z$, we also increase the probability mass of Z on S. Specifically, for every $y \in S$, we'll take

$$\begin{aligned}\Pr[Y' = y] &= e^\varepsilon \cdot \Pr[Z' = y] \\ &= \frac{e^\varepsilon}{1+e^\varepsilon} \cdot (\Pr[Y = y] + \Pr[Z = y]) \\ &\in [e^\varepsilon \cdot \Pr[Z = y], \Pr[Y = y]].\end{aligned}$$

This also implies that for $y \in S$, we have:

$$\begin{aligned}&\Pr[Y = y] - \Pr[Y' = y] \\ &\quad = \Pr[Z' = y] - \Pr[Z = y]\frac{\Pr[Y = y] - e^\varepsilon \cdot \Pr[Z = y]}{e^\varepsilon + 1},\end{aligned}$$

and thus

$$\begin{aligned}\alpha &\stackrel{\text{def}}{=} \sum_{y \in S} (\Pr[Y = y] - \Pr[Y' = y]) \\ &= \sum_{y \in S} (\Pr[Z' = y] - \Pr[Z = y]) \\ &= \frac{\Pr[Y \in S] - e^\varepsilon \cdot \Pr[Z \in S]}{e^\varepsilon + 1} \\ &\leq \frac{\delta}{e^\varepsilon + 1}.\end{aligned}$$

Similarly on the set $S' = \{y : \Pr[Z = y] > e^\varepsilon \cdot \Pr[Y = y]\}$, we can decrease the probability mass of Z and increase the probability mass of Y by a total of some $\alpha' \leq \delta/(e^\varepsilon + 1)$ so that for every $y \in S'$, we have $\Pr[Z' = y] = e^\varepsilon \cdot \Pr[Y' = y]$.

If $\alpha = \alpha'$, then we can take $\Pr[Z' = y] = \Pr[Z = y]$ and $\Pr[Y' = y] = \Pr[Y = y]$ for all $y \notin S \cup S'$, giving $D_\infty(Y \| Z) \leq \varepsilon$ and $\Delta(Y, Y') = \Delta(Z, Z') = \alpha$. If $\alpha \neq \alpha'$, say $\alpha > \alpha'$, then we need to still increase the probability mass of Y' and decrease the mass of Z' by a total of $\beta = \alpha - \alpha'$ on points outside of $S \cup S'$ in order to ensure that the probabilities sum to 1. That is, if we try to take the "mass functions" $\Pr[Y' = y]$ and $\Pr[Z' = y]$ as defined above, then while we do have the property that for every y, $\Pr[Y' = y] \leq e^\varepsilon \cdot \Pr[Z' = y]$ and $\Pr[Z' = y] \leq e^\varepsilon \cdot \Pr[Y' = y]$ we also have $\sum_y \Pr[Y' = y] = 1 - \beta$

and $\sum_y \Pr[Z' = y] = 1 + \beta$. However, this means that if we let $R = \{y : \Pr[Y' = y] < \Pr[Z' = y]\}$, then

$$\sum_{y \in R} (\Pr[Z' = y] - \Pr[Y' = y]) \geq \sum_y (\Pr[Z' = y] - \Pr[Y' = y]) = 2\beta.$$

So we can increase the probability mass of Y' on points in R by a total of β and decrease the probability mass of Z' on points in R by a total of β, while retaining the property that for all $y \in R$, $\Pr[Y' = y] \leq \Pr[Z' = y]$. The resulting Y' and Z' have the properties we want: $D_\infty(Y', Z') \leq \varepsilon$ and $\Delta(Y, Y'), \Delta(Z, Z') \leq \alpha$. □

Lemma 3.18. Suppose that random variables Y and Z satisfy $D_\infty(Y\|Z) \leq \varepsilon$ and $D_\infty(Z\|Y) \leq \varepsilon$. Then $D(Y\|Z) \leq \varepsilon \cdot (e^\varepsilon - 1)$.

Proof. We know that for any Y and Z it is the case that $D(Y\|Z) \geq 0$ (via the "log-sum inequality"), and so it suffices to bound $D(Y\|Z) + D(Z\|Y)$. We get:

$$\begin{aligned}
D(Y\|Z) &\leq D(Y\|Z) + D(Z\|Y) \\
&= \sum_y \Pr[Y = y] \cdot \left(\ln \frac{\Pr[Y = y]}{\Pr[Z = y]} + \ln \frac{\Pr[Z = y]}{\Pr[Y = y]}\right) \\
&\quad + (\Pr[Z = y] - \Pr[Y = y]) \cdot \left(\ln \frac{\Pr[Z = y]}{\Pr[Y = y]}\right) \\
&\leq \sum_y [0 + |\Pr[Z = y] - \Pr[Y = y]| \cdot \varepsilon] \\
&= \varepsilon \cdot \sum_y [\max\{\Pr[Y = y], \Pr[Z = y]\} \\
&\quad - \min\{\Pr[Y = y], \Pr[Z = y]\}] \\
&\leq \varepsilon \cdot \sum_y [(e^\varepsilon - 1) \cdot \min\{\Pr[Y = y], \Pr[Z = y]\}] \\
&\leq \varepsilon \cdot (e^\varepsilon - 1).
\end{aligned}$$

□

Lemma 3.19 (Azuma's Inequality)**.** Let $C_1, \ldots, C_k$ be real-valued random variables such that for every $i \in [k]$, $\Pr[|C_i| \leq \alpha] = 1$, and for

every $(c_1, \ldots, c_{i-1}) \in \mathrm{Supp}(C_1, \ldots, C_{i-1})$, we have

$$\mathbb{E}[C_i | C_1 = c_1, \ldots, C_{i-1} = c_{i-1}] \leq \beta.$$

Then for every $z > 0$, we have

$$\Pr\left[\sum_{i=1}^{k} C_i > k\beta + z\sqrt{k} \cdot \alpha\right] \leq e^{-z^2/2}.$$

3.5.2 Advanced composition

In addition to allowing the parameters to degrade more slowly, we would like our theorem to be able to handle more complicated forms of composition. However, before we begin, we must discuss what exactly we mean by composition. We would like our definitions to cover the following two interesting scenarios:

1. Repeated use of differentially private algorithms on the same database. This allows both the repeated use of the same mechanism multiple times, as well as the modular construction of differentially private algorithms from arbitrary private building blocks.
2. Repeated use of differentially private algorithms on *different* databases that may nevertheless contain information relating to the same individual. This allows us to reason about the cumulative privacy loss of a single individual whose data might be spread across multiple data sets, each of which may be used independently in a differentially private way. Since new databases are created all the time, and the adversary may actually influence the makeup of these new databases, this is a fundamentally different problem than repeatedly querying a single, fixed, database.

We want to model composition where the adversary can adaptively affect the databases being input to future mechanisms, as well as the queries to those mechanisms. Let $\mathcal{F}$ be a family of database access mechanisms. (For example $\mathcal{F}$ could be the set of all ε-differentially private mechanisms.) For a probabilistic adversary A, we consider two experiments, Experiment 0 and Experiment 1, defined as follows.

Experiment b for family $\mathcal{F}$ and adversary A:

For $i = 1, \ldots, k$:

1. A outputs two adjacent databases x_i^0 and x_i^1, a mechanism $\mathcal{M}_i \in \mathcal{F}$, and parameters w_i.
2. A receives $y_i \in_R \mathcal{M}_i(w_i, x_{i,b})$.

We allow the adversary A above to be stateful throughout the experiment, and thus it may choose the databases, mechanisms, and the parameters adaptively depending on the outputs of previous mechanisms. We define A's *view* of the experiment to be A's coin tosses and all of the mechanism outputs $(y_1, \ldots, y_k)$. (The x_i^j's, $\mathcal{M}_i$'s, and w_i's can all be reconstructed from these.)

For intuition, consider an adversary who always chooses x_i^0 to hold Bob's data and x_i^1 to differ only in that Bob's data are deleted. Then experiment 0 can be thought of as the "real world," where Bob allows his data to be used in many data releases, and Experiment 1 as an "ideal world," where the outcomes of these data releases do not depend on Bob's data. Our definitions of privacy still require these two experiments to be "close" to each other, in the same way as required by the definitions of differential privacy. The intuitive guarantee to Bob is that the adversary "can't tell", given the output of all k mechanisms, whether Bob's data was ever used.

Definition 3.7. We say that the family $\mathcal{F}$ of database access mechanisms satisfies *ε-differential privacy under k-fold adaptive composition* if for every adversary A, we have $D_\infty(V^0 \| V^1) \leq \varepsilon$ where V^b denotes the view of A in k-fold Composition Experiment b above.

(ε, δ)-differential privacy under k-fold adaptive composition instead requires that $D_\infty^\delta(V^0 \| V^1) \leq \varepsilon$.

Theorem 3.20 (Advanced Composition)**.** For all $\varepsilon, \delta, \delta' \geq 0$, the class of (ε, δ)-differentially private mechanisms satisfies $(\varepsilon', k\delta + \delta')$-differential privacy under k-fold adaptive composition for:

$$\varepsilon' = \sqrt{2k \ln(1/\delta')}\varepsilon + k\varepsilon(e^\varepsilon - 1).$$

Proof. A view of the adversary A consists of a tuple of the form $v = (r, y_1, \ldots, y_k)$, where r is the coin tosses of A and $y_1, \ldots, y_k$ are the outputs of the mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$. Let

$$B = \{v : \Pr[V^0 = v] > e^{\varepsilon'} \cdot \Pr[V^1 = v]\}.$$

We will show that $\Pr[V^0 \in B] \leq \delta$, and hence for every set S, we have

$$\Pr[V^0 \in S] \leq \Pr[V^0 \in B] + \Pr[V^0 \in (S \setminus B)] \leq \delta + e^{\varepsilon'} \cdot \Pr[V^1 \in S].$$

This is equivalent to saying that $D_\infty^\delta(V^0 \| V^1) \leq \varepsilon'$.

It remains to show $\Pr[V^0 \in B] \leq \delta$. Let random variable $V^0 = (R^0, Y_1^0, \ldots, Y_k^0)$ denote the view of A in Experiment 0 and $V^1 = (R^1, Y_1^1, \ldots, Y_k^1)$ the view of A in Experiment 1. Then for a fixed view $v = (r, y_1, \ldots, y_k)$, we have

$$\begin{aligned}
&\ln\left(\frac{\Pr[V^0 = v]}{\Pr[V^1 = v]}\right) \\
&= \ln\left(\frac{\Pr[R^0 = r]}{\Pr[R^1 = r]} \cdot \prod_{i=1}^{k} \frac{\Pr[Y_i^0 = y_i | R^0 = r, Y_1^0 = y_1, \ldots, Y_{i-1}^0 = y_{i-1}]}{\Pr[Y_i^1 = y_i | R^1 = r, Y_1^1 = y_1, \ldots, Y_{i-1}^1 = y_{i-1}]}\right) \\
&= \sum_{i=1}^{k} \ln\left(\frac{\Pr[Y_i^0 = y_i | R^0 = r, Y_1^0 = y_1, \ldots, Y_{i-1}^0 = y_{i-1}]}{\Pr[Y_i^1 = y_i | R^1 = r, Y_1^1 = y_1, \ldots, Y_{i-1}^1 = y_{i-1}]}\right) \\
&\stackrel{\text{def}}{=} \sum_{i=1}^{k} c_i(r, y_1, \ldots, y_i).
\end{aligned}$$

Now for every prefix $(r, y_1, \ldots, y_{i-1})$ we condition on $R^0 = r$, $Y_1^0 = y_1, \ldots, Y_{i-1}^0 = y_{i-1}$, and analyze the expectation and maximum possible value of the random variable $c_i(R^0, Y_1^0, \ldots, Y_i^0) = c_i(r, y_1, \ldots, y_{i-1}, Y_i^0)$. Once the prefix is fixed, the next pair of databases x_i^0 and x_i^1, the mechanism $\mathcal{M}_i$, and parameter w_i output by A are also determined (in both Experiment 0 and 1). Thus Y_i^0 is distributed according to $\mathcal{M}_i(w_i, x_i^0)$. Moreover for any value y_i, we have

$$c_i(r, y_1, \ldots, y_{i-1}, y_i) = \ln\left(\frac{\Pr[\mathcal{M}_i(w_i, x_i^0) = y_i]}{\Pr[\mathcal{M}_i(w_i, x_i^1) = y_i]}\right).$$

By ε-differential privacy this is bounded by ε. We can also reason as follows:

$$\begin{aligned}
&|c_i(r, y_1, \ldots, y_{i-1}, y_i)| \\
&\quad \leq \max\{D_\infty(\mathcal{M}_i(w_i, x_i^0) \| \mathcal{M}_i(w_i, x_i^1)), \\
&\qquad D_\infty(\mathcal{M}_i(w_i, x_i^1) \| \mathcal{M}_i(w_i, x_i^0))\} \\
&\quad = \varepsilon.
\end{aligned}$$

By Lemma 3.18, we have:

$$\begin{aligned}
&\mathbb{E}[c_i(R^0, Y_1^0, \ldots, Y_i^0) | R^0 = r, Y_1^0 = y_1, \ldots, Y_{i-1}^0 = y_{i-1}] \\
&\quad = D(\mathcal{M}_i(w_i, x_i^0) \| \mathcal{M}_i(w_i, x_i^1)) \\
&\quad \leq \varepsilon(e^\varepsilon - 1).
\end{aligned}$$

Thus we can apply Azuma's Inequality to the random variables $C_i = c_i(R^0, Y_1^0, \ldots, Y_i^0)$ with $\alpha = \varepsilon$, $\beta = \varepsilon \cdot \varepsilon_0$, and $z = \sqrt{2\ln(1/\delta)}$, to deduce that

$$\Pr[V^0 \in B] = \Pr\left[\sum_i C_i > \varepsilon'\right] < e^{-z^2/2} = \delta,$$

as desired.

To extend the proof to composition of (ε, δ)-differentially private mechanisms, for $\delta > 0$, we use the characterization of approximate max-divergence from Lemma 3.17 (Part 2) to reduce the analysis to the same situation as in the case of $(\varepsilon, 0)$-indistinguishable sequences. Specifically, using Lemma 3.17, Part 2 for each of the differentially private mechanisms selected by the adversary A and the triangle inequality for statistical distance, it follows that that V^0 is $k\delta$-close to a random variable $W = (R, Z_1, \ldots, Z_k)$ such that for every prefix $r, y_1, \ldots, y_{i-1}$, if we condition on $R = R^1 = r, Z_1 = Y_1^1 = y_1, \ldots, Z_{i-1} = Y_{i-1}^1 = y_{i-1}$, then it holds that $D_\infty(Z_i \| Y_i^1) \leq \varepsilon$ and $D_\infty(Y_i^1 \| Z_i) \leq \varepsilon$.

This suffices to show that $D_\infty^{\delta'}(W \| V^1) \leq \varepsilon'$. Since V^0 is $k\delta$-close to W, Lemma 3.17, Part 1 gives $D^{\delta' + k\delta}(V^0 \| W) \leq \varepsilon'$. □

An immediate and useful corollary tells us a safe choice of ε for each of k mechanisms if we wish to ensure $(\varepsilon', k\delta + \delta')$-differential privacy for a given ε', δ'.

Corollary 3.21. Given target privacy parameters $0 < \varepsilon' < 1$ and $\delta' > 0$, to ensure $(\varepsilon', k\delta + \delta')$ cumulative privacy loss over k mechanisms, it suffices that each mechanism is (ε, δ)-differentially private, where

$$\varepsilon = \frac{\varepsilon'}{2\sqrt{2k\ln(1/\delta')}}.$$

Proof. Theorem 3.20 tells us the composition will be $(\varepsilon^*, k\delta + \delta')$ for all δ', where $\varepsilon^* = \sqrt{2k\ln(1/\delta')} \cdot \varepsilon + k\varepsilon^2$. When $\varepsilon' < 1$, we have that $\varepsilon^* \leq \varepsilon'$ as desired. □

Note that the above corollary gives a rough guide for how to set ε to get desired privacy parameters under composition. When one cares about optimizing constants (which one does when dealing with actual implementations), ε can be set more tightly by appealing directly to the composition theorem.

Example 3.7. Suppose, over the course of his lifetime, Bob is a member of $k = 10,000$ $(\varepsilon_0, 0)$-differentially private databases. Assuming no coordination among these databases — the administrator of any given database may not even be aware of the existence of the other databases — what should be the value of ε_0 so that, over the course of his lifetime, Bob's cumulative privacy loss is bounded by $\varepsilon = 1$ with probability at least $1 - e^{-32}$? Theorem 3.20 says that, taking $\delta' = e^{-32}$ it suffices to have $\varepsilon_0 \leq 1/801$. This turns out to be essentially optimal against an arbitrary adversary, assuming no coordination among distinct differentially private databases.

So how many queries can we answer with non-trivial accuracy? On a database of size n let us say the accuracy is non-trivial if the error is of order $o(n)$. Theorem 3.20 says that for fixed values of ε and δ, it is possible to answer close to n^2 counting queries with non-trivial accuracy. Similarly, one can answer close to n queries while still having noise $o(\sqrt{n})$ — that is, noise less than the sampling error. We will see that it is possible to dramatically improve on these results, handling, in some cases, even an exponential number of queries with noise only slightly larger than $\sqrt{n}$, by coordinating the noise added to the individual responses. It turns out that such coordination is essential: without

coordination the bound in the advanced composition theorem is almost tight.

3.5.3 Laplace versus Gauss

An alternative to adding Laplacian noise is to add Gaussian noise. In this case, rather than scaling the noise to the ℓ_1 sensitivity Δf, we instead scale to the ℓ_2 sensitivity:

Definition 3.8 (ℓ_2-sensitivity)**.** The ℓ_2-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is:

$$\Delta_2(f) = \max_{\substack{x,y \in \mathbb{N}^{|\mathcal{X}|} \\ \|x-y\|_1=1}} \|f(x) - f(y)\|_2.$$

The *Gaussian Mechanism* with parameter b adds zero-mean Gaussian noise with variance b in each of the k coordinates. The following theorem is proved in Appendix A.

Theorem 3.22. Let $\varepsilon \in (0, 1)$ be arbitrary. For $c^2 > 2\ln(1.25/\delta)$, the Gaussian Mechanism with parameter $\sigma \geq c\Delta_2(f)/\varepsilon$ is (ε, δ)-differentially private.

Among the advantages to Gaussian noise is that the noise added for privacy is of the same type as other sources of noise; moreover, the sum of two Gaussians is a Gaussian, so the effects of the privacy mechanism on the statistical analysis may be easier to understand and correct for.

The two mechanisms yield the same cumulative loss under composition, so even though the privacy guarantee is weaker for each individual computation, the cumulative effects over many computations are comparable. Also, if δ is sufficiently (e.g., subpolynomially) small, in practice we will never experience the weakness of the guarantee.

That said, there is a theoretical disadvantage to Gaussian noise, relative to what we experience with Laplace noise. Consider Report Noisy Max (with Laplace noise) in a case in which every candidate output has the same quality score on database x as on its neighbor y. Independent of the number of candidate outputs, the mechanism yields $(\varepsilon, 0)$-differential privacy. If instead we use Gaussian noise and report the max, and if the number of candidates is large compared to $1/\delta$,

then we will exactly select for the events with large Gaussian noise — noise that occurs with probability less than δ. When we are this far out on the tail of the Gaussian we no longer have a guarantee that the observation is within an $e^{\pm\varepsilon}$ factor as likely to occur on x as on y.

3.5.4 Remarks on composition

The ability to analyze cumulative privacy loss under composition gives us a handle on what a world of differentially private databases can offer. A few observations are in order.

Weak Quantification. Assume that the adversary always chooses x_i^0 to hold Bob's data, and x_i^1 to be the same database but with Bob's data deleted. Theorem 3.20, with appropriate choise of parameters, tells us that an adversary — including one that knows or even selects(!) the database pairs — has little advantage in determining the value of $b \in \{0, 1\}$. This is an inherently weak quantification. We can ensure that the adversary is unlikely to distinguish reality from any given alternative, but we cannot ensure this simultaneously for all alternatives. If there are one zillion databases but Bob is a member of only 10,000 of these, then we are not simultaneously protecting Bob's *absence* from all zillion minus ten thousand. This is analogous to the quantification in the definition of (ε, δ)-differential privacy, where we fix in advance a pair of adjacent databases and argue that with high probability the output will be almost equally likely with these two databases.

Humans and Ghosts. Intuitively, an $(\epsilon, 0)$-differentially private database with a small number of bits per record is less protective than a differentially private database with the same choice of ϵ that contains our entire medical histories. So in what sense is our principle privacy measure, ϵ, telling us the same thing about databases that differ radically in the complexity and sensitivity of the data they store? The answer lies in the composition theorems. Imagine a world inhabited by two types of beings: ghosts and humans. Both types of beings behave the same, interact with others in the same way, write, study, work, laugh, love, cry, reproduce, become ill, recover, and age in the same fashion. The only difference is that ghosts have no records in

databases, while humans do. The goal of the privacy adversary is to determine whether a given 50-year old, the "target," is a ghost or a human. Indeed, the adversary is given all 50 years to do so. The adversary does not need to remain passive, for example, she can organize clinical trials and enroll patients of her choice, she can create humans to populate databases, effectively creating the worst-case (for privacy) databases, she can expose the target to chemicals at age 25 and again at 35, and so on. She can know everything about the target that could possibly be entered into any database. She can know which databases the target would be in, were the target human. The composition theorems tell us that the privacy guarantees of each database — regardless of the data type, complexity, and sensitivity — give comparable protection for the human/ghost bit.

3.6 The sparse vector technique

The Laplace mechanism can be used to answer adaptively chosen low sensitivity queries, and we know from our composition theorems that the privacy parameter degrades proportionally to the number of queries answered (or its square root). Unfortunately, it will often happen that we have a very large number of questions to answer — too many to yield a reasonable privacy guarantee using independent perturbation techniques, even with the advanced composition theorems of Section 3.5. In some situations however, we will only care to know the identity of the queries that lie above a certain threshold. In this case, we can hope to gain over the naïve analysis by discarding the numeric answer to queries that lie significantly below the threshold, and merely reporting that they do indeed lie below the threshold. (We will be able to get the numeric values of the above-threshold queries as well, at little additional cost, if we so choose). This is similar to what we did in the Report Noisy Max mechanism in section 3.3, and indeed iterating either that algorithm or the exponential mechanism would be an option for the non-interactive, or offline, case.

In this section, we show how to analyze a method for this in the online setting. The technique is simple — add noise and report only

whether the noisy value exceeds the threshold — and our emphasis is on the analysis, showing that privacy degrades only with the number of queries which actually lie above the threshold, rather than with the total number of queries. This can be a huge savings if we know that the set of queries that lie above the threshold is much smaller than the total number of queries — that is, if the answer vector is *sparse*.

In a little more detail, we will consider a sequence of events — one for each query — which occur if a query evaluated on the database exceeds a given (known, public) threshold. Our goal will be to release a bit vector indicating, for each event, whether or not it has occurred. As each query is presented, the mechanism will compute a noisy response, compare it to the (publicly known) threshold, and, if the threshold is exceeded, reveal this fact. For technical reasons in the proof of privacy (Theorem 3.24), the algorithm works with a noisy version $\hat{T}$ of the threshold T. While T is public the noisy version $\hat{T}$ is not.

Rather than incurring a privacy loss for each *possible* query, the analysis below will result in a privacy cost only for the query values that are near or above the threshold.

The Setting. Let m denote the total number of sensitivity 1 queries, which may be chosen adaptively. Without loss of generality, there is a single threshold T fixed in advance (alternatively, each query can have its own threshold, but the results are unchanged). We will be adding noise to query values and comparing the results to T. A *positive* outcome means that a noisy query value exceeds the threshold. We expect a small number c of noisy values to exceed the threshold, and we are releasing only the noisy values above the threshold. The algorithm will use c in its stopping condition.

We will first analyze the case in which the algorithm halts after $c = 1$ above-threshold query, and show that this algorithm is ϵ-differentially private no matter how long the *total* sequence of queries is. We will then analyze the case of $c > 1$ by using our composition theorems, and derive bounds both for $(\epsilon, 0)$ and (ϵ, δ)-differential privacy.

We first argue that AboveThreshold, the algorithm specialized to the case of only one above-threshold query, is private and accurate.

Algorithm 1 Input is a private database D, an adaptively chosen stream of sensitivity 1 queries $f_1, \ldots$, and a threshold T. Output is a stream of responses $a_1, \ldots$

AboveThreshold$(D, \{f_i\}, T, \epsilon)$
  Let $\hat{T} = T + \text{Lap}\left(\frac{2}{\epsilon}\right)$.
  for Each query i **do**
    Let $\nu_i = \text{Lap}(\frac{4}{\epsilon})$
    if $f_i(D) + \nu_i \geq \hat{T}$ **then**
      Output $a_i = \top$.
      Halt.
    else
      Output $a_i = \perp$.
    end if
  end for

Theorem 3.23. AboveThreshold is $(\epsilon, 0)$-differentially private.

Proof. Fix any two neighboring databases D and D'. Let A denote the random variable representing the output of **AboveThreshold**$(D, \{f_i\}, T, \epsilon)$ and let A' denote the random variable representing the output of **AboveThreshold**$(D', \{f_i\}, T, \epsilon)$. The output of the algorithm is some realization of these random variables, $a \in \{\top, \perp\}^k$ and has the form that for all $i < k$, $a_i = \perp$ and $a_k = \top$. There are two types of random variables internal to the algorithm: the noisy threshold $\hat{T}$ and the perturbations to each of the k queries, $\{\nu_i\}_{i=1}^k$. For the following analysis, we will fix the (arbitrary) values of $\nu_1, \ldots, \nu_{k-1}$ and take probabilities over the randomness of ν_k and $\hat{T}$. Define the following quantity representing the maximum noisy value of any query $f_1, \ldots, f_{k-1}$ evaluated on D:

$$g(D) = \max_{i<k} \left(f_i(D) + \nu_i\right)$$

In the following, we will abuse notation and write $\Pr[\hat{T} = t]$ as shorthand for the pdf of $\hat{T}$ evaluated at t (similarly for ν_k), and write $\mathbf{1}[x]$ to denote the indicator function of event x. Note that fixing the values

of $\nu_1, \ldots, \nu_{k-1}$ (which makes $g(D)$ a deterministic quantity), we have:

$$\begin{aligned}
\Pr_{\hat{T},\nu_k}[A = a] &= \Pr_{\hat{T},\nu_k}[\hat{T} > g(D) \text{ and } f_k(D) + \nu_k \geq \hat{T}] \\
&= \Pr_{\hat{T},\nu_k}[\hat{T} \in (g(D), f_k(D) + \nu_k]] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Pr[\nu_k = v] \\
&\quad \cdot \Pr[\hat{T} = t]\mathbf{1}[t \in (g(D), f_k(D) + v]]dvdt \\
&\doteq *
\end{aligned}$$

We now make a change of variables. Define:

$$\hat{v} = v + g(D) - g(D') + f_k(D') - f_k(D)$$

$$\hat{t} = t + g(D) - g(D')$$

and note that for any D, D', $|\hat{v} - v| \leq 2$ and $|\hat{t} - t| \leq 1$. This follows because each query $f_i(D)$ is 1-sensitive, and hence the quantity $g(D)$ is 1-sensitive as well. Applying this change of variables, we have:

$$\begin{aligned}
* &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Pr[\nu_k = \hat{v}] \cdot \Pr[\hat{T} = \hat{t}]\mathbf{1}[(t + g(D) - g(D')) \\
&\qquad\qquad \in (g(D), f_k(D') + v + g(D) - g(D'))]dvdt \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Pr[\nu_k = \hat{v}] \cdot \Pr[\hat{T} = \hat{t}]\mathbf{1}[(t \in (g(D'), f_k(D') + v]]dvdt \\
&\leq \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(\epsilon/2) \Pr[\nu_k = v] \\
&\quad \cdot \exp(\epsilon/2) \Pr[\hat{T} = t]\mathbf{1}[(t \in (g(D'), f_k(D') + v]]dvdt \\
&= \exp(\epsilon) \Pr_{\hat{T},\nu_k}[\hat{T} > g(D') \text{ and } f_k(D') + \nu_k \geq \hat{T}] \\
&= \exp(\epsilon) \Pr_{\hat{T},\nu_k}[A' = a]
\end{aligned}$$

where the inequality comes from our bounds on $|\hat{v} - v|$ and $|\hat{t} - t|$ and the form of the pdf of the Laplace distribution. □

Definition 3.9 (Accuracy)**.** We will say that an algorithm which outputs a stream of answers $a_1, \ldots, \in \{\top, \bot\}^*$ in response to a stream of k

queries $f_1, \ldots, f_k$ is (α, β)-accurate with respect to a threshold T if except with probability at most β, the algorithm does not halt before f_k, and for all $a_i = \top$:

$$f_i(D) \geq T - \alpha$$

and for all $a_i = \perp$:

$$f_i(D) \leq T + \alpha.$$

What can go wrong in Algorithm 1? The noisy threshold $\hat{T}$ can be very far from T, say, $|\hat{T} - T| > \alpha$. In addition a small count $f_i(D) < T - \alpha$ can have so much noise added to it that it is reported as above threshold (even when the threshold is close to correct), and a large count $f_i(D) > T + \alpha$ can be reported as below threshold. All of these happen with probability exponentially small in α. In summary, we can have a problem with the choice of the noisy threshold or we can have a problem with one or more of the individual noise values ν_i. Of course, we could have both kinds of errors, so in the analysis below we allocate $\alpha/2$ to each type.

Theorem 3.24. For any sequence of k queries $f_1, \ldots, f_k$ such that $|\{i < k : f_i(D) \geq T - \alpha\}| = 0$ (i.e. the only query close to being above threshold is possibly the last one), AboveThreshold$(D, \{f_i\}, T, \epsilon)$ is (α, β) accurate for:

$$\alpha = \frac{8(\log k + \log(2/\beta))}{\epsilon}.$$

Proof. Observe that the theorem will be proved if we can show that except with probability at most β:

$$\max_{i \in [k]} |\nu_i| + |T - \hat{T}| \leq \alpha$$

If this is the case, then for any $a_i = \top$, we have:

$$f_i(D) + \nu_i \geq \hat{T} \geq T - |T - \hat{T}|$$

or in other words:

$$f_i(D) \geq T - |T - \hat{T}| - |\nu_i| \geq T - \alpha$$

Similarly, for any $a_i = \perp$ we have:

$$f_i(D) < \hat{T} \leq T + |T - \hat{T}| + |\nu_i| \leq T + \alpha$$

We will also have that for any $i < k$: $f_i(D) < T - \alpha < T - |\nu_i| - |T - \hat{T}|$, and so: $f_i(D) + \nu_i \leq \hat{T}$, meaning $a_i = \perp$. Therefore the algorithm does not halt before k queries are answered.

We now complete the proof.

Recall that if $Y \sim \text{Lap}(b)$, then: $\Pr[|Y| \geq t \cdot b] = \exp(-t)$. Therefore we have:

$$\Pr[|T - \hat{T}| \geq \frac{\alpha}{2}] = \exp\left(-\frac{\epsilon\alpha}{4}\right)$$

Setting this quantity to be at most $\beta/2$, we find that we require $\alpha \geq \frac{4\log(2/\beta)}{\epsilon}$

Similarly, by a union bound, we have:

$$\Pr[\max_{i \in [k]} |\nu_i| \geq \alpha/2] \leq k \cdot \exp\left(-\frac{\epsilon\alpha}{8}\right)$$

Setting this quantity to be at most $\beta/2$, we find that we require $\alpha \geq \frac{8(\log(2/\beta) + \log k)}{\epsilon}$ These two claims combine to prove the theorem. □

We now show how to handle multiple "above threshold" queries using composition.

The Sparse algorithm can be thought of as follows: As queries come in, it makes repeated calls to AboveThreshold. Each time an above threshold query is reported, the algorithm simply restarts the remaining stream of queries on a new instantiation of AboveThreshold. It halts after it has restarted AboveThreshold c times (i.e. after c above threshold queries have appeared). Each instantiation of AboveThreshold is $(\epsilon, 0)$-private, and so the composition theorems apply.

Theorem 3.25. Sparse is (ϵ, δ)-differentially private.

Proof. We observe that Sparse is exactly equivalent to the following procedure: We run AboveThreshold$(D, \{f_i\}, T, \epsilon')$ on our stream of queries $\{f_i\}$ setting

$$\epsilon' = \begin{cases} \frac{\epsilon}{c}, & \text{If } \delta = 0; \\ \frac{\epsilon}{\sqrt{8c \ln \frac{1}{\delta}}}, & \text{Otherwise.} \end{cases}$$

Algorithm 2 Input is a private database D, an adaptively chosen stream of sensitivity 1 queries $f_1, \ldots$, a threshold T, and a cutoff point c. Output is a stream of answers $a_1, \ldots$

Sparse$(D, \{f_i\}, T, c, \epsilon, \delta)$

If $\delta = 0$ **Let** $\sigma = \frac{2c}{\epsilon}$. **Else Let** $\sigma = \frac{\sqrt{32c \ln \frac{1}{\delta}}}{\epsilon}$
Let $\hat{T}_0 = T + \text{Lap}(\sigma)$
Let count $= 0$
for Each query i **do**
  Let $\nu_i = \text{Lap}(2\sigma)$
  if $f_i(D) + \nu_i \geq \hat{T}_{\text{count}}$ **then**
    Output $a_i = \top$.
    Let count = count $+1$.
    Let $\hat{T}_{\text{count}} = T + \text{Lap}(\sigma)$
  else
    Output $a_i = \perp$.
  end if
  if count $\geq c$ **then**
    Halt.
  end if
end for

using the answers supplied by AboveThreshold. When AboveThreshold halts (after 1 above threshold query), we simply restart Sparse$(D, \{f_i\}, T, \epsilon')$ on the remaining stream, and continue in this manner until we have restarted AboveThreshold c times. After the c'th restart of AboveThreshold halts, we halt as well. We have already proven that AboveThreshold$(D, \{f_i\}, T, \epsilon')$ is $(\epsilon', 0)$ differentially private. Finally, by the advanced composition theorem (Theorem 3.20), c applications of an $\epsilon' = \frac{\epsilon}{\sqrt{8c \ln \frac{1}{\delta}}}$-differentially private algorithm is (ϵ, δ)-differentially private, and c applications of an $\epsilon' = \epsilon/c$ differentially private algorithm is $(\epsilon, 0)$-private as desired. □

It remains to prove accuracy for Sparse, by again observing that Sparse consists only of c calls to AboveThreshold. We note that if each

of these calls to AboveThreshold is $(\alpha, \beta/c)$-accurate, then Sparse will be (α, β)-accurate.

Theorem 3.26. For any sequence of k queries $f_1, \ldots, f_k$ such that $L(T) \equiv |\{i : f_i(D) \geq T - \alpha\}| \leq c$, if $\delta > 0$, Sparse is (α, β) accurate for:

$$\alpha = \frac{(\ln k + \ln \frac{2c}{\beta})\sqrt{512c \ln \frac{1}{\delta}}}{\epsilon}.$$

If $\delta = 0$, Sparse is (α, β) accurate for:

$$\alpha = \frac{8c(\ln k + \ln(2c/\beta))}{\epsilon}$$

Proof. We simply apply Theorem 3.24 setting β to be β/c, and ϵ to be $\frac{\epsilon}{\sqrt{8c \ln \frac{1}{\delta}}}$ and ϵ/c, depending on whether $\delta > 0$ or $\delta = 0$, respectively. □

Finally, we give a version of Sparse that actually outputs the numeric values of the above threshold queries, which we can do with only a constant factor loss in accuracy. We call this algorithm NumericSparse, and it is simply a composition of Sparse with the Laplace mechanism. Rather than outputting a vector $a \in \{\top, \bot\}^*$, it outputs a vector $a \in (\mathbb{R} \cup \{\bot\})^*$.

We observe that NumericSparse is private:

Theorem 3.27. NumericSparse is (ϵ, δ)-differentially private.

Proof. Observe that if $\delta = 0$, NumericSparse$(D, \{f_i\}, T, c, \epsilon, 0)$ is simply the adaptive composition of Sparse$(D, \{f_i\}, T, c, \frac{8}{9}\epsilon, 0)$, together with the Laplace mechanism with privacy parameters $(\epsilon', \delta) = (\frac{1}{9}\epsilon, 0)$. If $\delta > 0$, then NumericSparse$(D, \{f_i\}, T, c, \epsilon, 0)$ is the composition of Sparse$(D, \{f_i\}, T, c, \frac{\sqrt{512}}{\sqrt{512}+1}\epsilon, \delta/2)$ together with the Laplace mechanism with privacy parameters $(\epsilon', \delta) = (\frac{1}{\sqrt{512}+1}\epsilon, \delta/2)$. Hence the privacy of NumericSparse follows from simple composition. □

To discuss accuracy, we must define what we mean by the accuracy of a mechanism that outputs a stream $a \in (\mathbb{R} \cup \{\bot\})^*$ in response to a sequence of numeric valued queries:

Algorithm 3 Input is a private database D, an adaptively chosen stream of sensitivity 1 queries $f_1, \ldots$, a threshold T, and a cutoff point c. Output is a stream of answers $a_1, \ldots$

NumericSparse$(D, \{f_i\}, T, c, \epsilon, \delta)$

If $\delta = 0$ **Let** $\epsilon_1 \leftarrow \frac{8}{9}\epsilon$, $\epsilon_2 \leftarrow \frac{2}{9}\epsilon$. **Else Let** $\epsilon_1 = \frac{\sqrt{512}}{\sqrt{512}+1}\epsilon, \epsilon_2 = \frac{2}{\sqrt{512}+1}$

If $\delta = 0$ **Let** $\sigma(\epsilon) = \frac{2c}{\epsilon}$. **Else Let** $\sigma(\epsilon) = \frac{\sqrt{32c \ln \frac{2}{\delta}}}{\epsilon}$

Let $\hat{T}_0 = T + \text{Lap}(\sigma(\epsilon_1))$

Let count $= 0$

for Each query i **do**

  Let $\nu_i = \text{Lap}(2\sigma(\epsilon_1))$

  if $f_i(D) + \nu_i \geq \hat{T}_{\text{count}}$ **then**

    Let $\upsilon_i \leftarrow \text{Lap}(\sigma(\epsilon_2))$

    Output $a_i = f_i(D) + \upsilon_i$.

    Let count $=$ count $+1$.

    Let $\hat{T}_{\text{count}} = T + \text{Lap}(\sigma(\epsilon_1))$

  else

    Output $a_i = \perp$.

  end if

  if count $\geq c$ **then**

    Halt.

  end if

end for

Definition 3.10 (Numeric Accuracy)**.** We will say that an algorithm which outputs a stream of answers $a_1, \ldots, \in (\mathbb{R} \cup \{\perp\})^*$ in response to a stream of k queries $f_1, \ldots, f_k$ is (α, β)-accurate with respect to a threshold T if except with probability at most β, the algorithm does not halt before f_k, and for all $a_i \in \mathbb{R}$:

$$|f_i(D) - a_i| \leq \alpha$$

and for all $a_i = \perp$:

$$f_i(D) \leq T + \alpha.$$

Theorem 3.28. For any sequence of k queries $f_1, \ldots, f_k$ such that $L(T) \equiv |\{i : f_i(D) \geq T - \alpha\}| \leq c$, if $\delta > 0$, NumericSparse is (α, β)

accurate for:

$$\alpha = \frac{(\ln k + \ln \frac{4c}{\beta})\sqrt{c \ln \frac{2}{\delta}}(\sqrt{512}+1)}{\epsilon}.$$

If $\delta = 0$, Sparse is (α, β) accurate for:

$$\alpha = \frac{9c(\ln k + \ln(4c/\beta))}{\epsilon}$$

Proof. Accuracy requires two conditions: first, that for all $a_i = \perp$: $f_i(D) \leq T + \alpha$. This holds with probability $1 - \beta/2$ by the accuracy theorem for Sparse. Next, for all $a_i \in \mathbb{R}$, it requires $|f_i(D) - a_i| \leq \alpha$. This holds for with probability $1 - \beta/2$ by the accuracy of the Laplace mechanism. □

What did we show in the end? If we are given a sequence of queries together with a guarantee that only at most c of them have answers above $T - \alpha$, we can answer those queries that are above a given threshold T, up to error α. This accuracy is equal, up to constants and a factor of $\log k$, to the accuracy we would get, given the same privacy guarantee, if we knew the identities of these large above-threshold queries ahead of time, and answered them with the Laplace mechanism. That is, the sparse vector technique allowed us to fish out the identities of these large queries almost "for free", paying only logarithmically for the irrelevant queries. This is the same guarantee that we could have gotten by trying to find the large queries with the exponential mechanism and then answering them with the Laplace mechanism. This algorithm, however, is trivial to run, and crucially, allows us to choose our queries adaptively.

3.7 Bibliographic notes

Randomized Response is due to Warner [84] (predating differential privacy by four decades!). The Laplace mechanism is due to Dwork et al. [23]. The exponential mechanism was invented by McSherry and Talwar [60]. Theorem 3.16 (simple composition) was claimed in [21]; the proof appearing in Appendix B is due to Dwork and Lei [22];

McSherry and Mironov obtained a similar proof. The material in Sections 3.5.1 and 3.5.2 is taken almost verbatim from Dwork et al. [32]. Prior to [32] composition was modeled informally, much as we did for the simple composition bounds. For specific mechanisms applied on a single database, there are "evolution of confidence" arguments due to Dinur, Dwork, and Nissim [18, 31], (which pre-date the definition of differential privacy) showing that the privacy parameter in k-fold composition need only deteriorate like $\sqrt{k}$ if we are willing to tolerate a (negligible) loss in δ (for $k < 1/\varepsilon^2$). Theorem 3.20 generalizes those arguments to arbitrary differentially private mechanisms,

The claim that without coordination in the noise the bounds in the composition theorems are almost tight is due to Dwork, Naor, and Vadhan [29]. The sparse vector technique is an abstraction of a technique that was introduced, by Dwork, Naor, Reingold, Rothblum, and Vadhan [28] (indicator vectors in the proof of Lemma 4.4). It has subsequently found wide use (e.g. by Roth and Roughgarden [74], Dwork, Naor, Pitassi, and Rothblum [26], and Hardt and Rothblum [44]). In our presentation of the technique, the proof of Theorem 3.23 is due to Salil Vadhan.

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.

$\mathbf{MW}(x^t, f_t, v_t)$:

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 $x_i^0 = 1/|\mathcal{X}|$ 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].

5

Generalizations

In this section we generalize the query release algorithms of the previous section. As a result, we get bounds for arbitrary low sensitivity queries (not just linear queries), as well as new bounds for linear queries. These generalizations also shed some light on a connection between query release and machine learning.

The SmallDB offline query release mechanism in Section 4 is a special case of what we call the *net mechanism*. We saw that both mechanisms in that section yield *synthetic databases*, which provide a convenient means for approximating the value of any query in $\mathcal{Q}$ on the private database: just evaluate the query on the synthetic database and take the result as the noisy answer. More generally, a mechanism can produce a *data structure* of arbitrary form, that, together with a fixed, public, algorithm (independent of the database) provides a method for approximating the values of queries.

The Net mechanism is a straightforward generalization of the SmallDB mechanism: First, fix, independent of the actual database, an α-net of data structures such that evaluation of any query in $\mathcal{Q}$ using the released data structure gives a good (within an additive α error) estimate of the value of the query on the private database. Next, apply

the exponential mechanism to choose an element of this net, where the quality function minimizes the maximum error, over the queries in $\mathcal{Q}$, for the elements of the net.

We also generalize the online multiplicative weights algorithm so that we can instantiate it with any other *online learning algorithm* for learning a database with respect to a set of queries. We note that such a mechanism can be run either online, or offline, where the set of queries to be asked to the "online" mechanism is instead selected using a "private distinguisher," which identifies queries on which the current hypothesis of the learner differs substantially from the real database. These are queries that would have yielded an update step in the online algorithm. A "distinguisher" turns out to be equivalent to an agnostic learning algorithm, which sheds light on a source of hardness for efficient query release mechanisms.

In the following sections, we will discuss *data structures* for classes of queries $\mathcal{Q}$.

Definition 5.1. A data structure D drawn from some class of data structures $\mathcal{D}$ for a class of queries $\mathcal{Q}$ is implicitly endowed with an evaluation function $\text{Eval} : \mathcal{D} \times \mathcal{Q} \to \mathbb{R}$ with which we can evaluate any query in $\mathcal{Q}$ on D. However, to avoid being encumbered by notation, we will write simply $f(D)$ to denote $\text{Eval}(D, f)$ when the meaning is clear from context.

5.1 Mechanisms via α-nets

Given a collection of queries $\mathcal{Q}$, we define an α-net as follows:

Definition 5.2 (α-net)**.** An α-net of data structures with respect to a class of queries $\mathcal{Q}$ is a set $\mathcal{N} \subset \mathbb{N}^{|\mathcal{X}|}$ such that for all $x \in \mathbb{N}^{|\mathcal{X}|}$, there exists an element of the α-net $y \in \mathcal{N}$ such that:

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

We write $\mathcal{N}_\alpha(\mathcal{Q})$ to denote an α-net of minimum cardinality among the set of all α-nets for $\mathcal{Q}$.

That is, for every possible database x, there exists a member of the α-net that "looks like" x with respect to all queries in $\mathcal{Q}$, up to an error tolerance of α.

Small α-nets will be useful for us, because when paired with the exponential mechanism, they will lead directly to mechanisms for answering queries with high accuracy. Given a class of functions $\mathcal{Q}$, we will define an instantiation of the exponential mechanism known as the *Net* mechanism. We first observe that the Net mechanism preserves ε-differential privacy.

Algorithm 7 The Net Mechanism

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

Let $\mathcal{R} \leftarrow \mathcal{N}_\alpha(\mathcal{Q})$

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

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

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

Proposition 5.1. The Net mechanism is $(\varepsilon, 0)$ differentially private.

Proof. The Net 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 begin understanding the utility guarantees of the Net mechanism:

Proposition 5.2. Let $\mathcal{Q}$ be any class of sensitivity $1/\|x\|_1$ queries. Let y be the database output by NetMechanism$(x, \mathcal{Q}, \varepsilon, \alpha)$. Then with probability $1 - \beta$:

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

Proof. By applying Theorem 3.11 and noting that $S(q) = \frac{1}{\|x\|_1}$, and that $\mathrm{OPT}_q(D) \leq \alpha$ by the definition of an α-net, we find:

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

Plugging in $t = \log\left(\frac{1}{\beta}\right)$ completes the proof. □

We can therefore see that an upper bound on $|\mathcal{N}_\alpha(\mathcal{Q})|$ for a collection of functions $\mathcal{Q}$ immediately gives an upper bound on the accuracy that a differentially private mechanism can provide simultaneously for *all* functions in the class $\mathcal{Q}$.

This is exactly what we did in Section 4.1, where we saw that the key quantity is the VC-dimension of $\mathcal{Q}$, when $\mathcal{Q}$ is a class of linear queries.

5.2 The iterative construction mechanism

In this section, we derive an offline generalization of the private multiplicative weights algorithm, which can be instantiated with any properly defined learning algorithm. Informally, a database update algorithm maintains a sequence of data structures $D^1, D^2, \ldots$ that give increasingly good approximations to the input database x (in a sense that depends on the database update algorithm). Moreover, these mechanisms produce the next data structure in the sequence by considering only one query f that *distinguishes* the real database in the sense that $f(D^t)$ differs significantly from $f(x)$. The algorithm in this section shows that, up to small factors, solving the query-release problem in a differentially private manner is equivalent to solving the simpler *learning* or *distinguishing* problem in a differentially private manner: given a private distinguishing algorithm and a non-private database update algorithm, we get a corresponding private release algorithm. We can plug in the exponential mechanism as a canonical private distinguisher, and the multiplicative weights algorithm as a generic database update algorithm for the general linear query setting, but more efficient distinguishers are possible in special cases.

Syntactically, we will consider functions of the form $U : \mathcal{D} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{D}$, where $\mathcal{D}$ represents a class of data structures on which queries in $\mathcal{Q}$ can be evaluated. The inputs to U are a data structure in $\mathcal{D}$, which represents the current data structure D^t; a query f, which represents the distinguishing query, and may be restricted to a certain set $\mathcal{Q}$; and also a real number, which estimates $f(x)$. Formally, we define a *database update sequence*, to capture the sequence of inputs to U used to generate the database sequence $D^1, D^2, \ldots$.

Definition 5.3 (Database Update Sequence)**.** Let $x \in \mathbb{N}^{|\mathcal{X}|}$ be any database and let $\{(D^t, f_t, v_t)\}_{t=1,\ldots,L} \in (\mathcal{D} \times \mathcal{Q} \times \mathbb{R})^L$ be a sequence of tuples. We say the sequence is a $(U, x, \mathcal{Q}, \alpha, T)$-*database update sequence* if it satisfies the following properties:

1. $D^1 = U(\perp, \cdot, \cdot)$,
2. for every $t = 1, 2, \ldots, L$, $|f_t(x) - f_t(D^t)| \geq \alpha$,
3. for every $t = 1, 2, \ldots, L$, $|f_t(x) - v_t| < \alpha$,
4. and for every $t = 1, 2, \ldots, L-1$, $D^{t+1} = U(D^t, f_t, v_t)$.

We note that for all of the database update algorithms we consider, the approximate answer v_t is used only to determine the *sign* of $f_t(x) - f_t(D^t)$, which is the motivation for requiring that the estimate of $f_t(x)$ (v_t) have error smaller than α. The main measure of efficiency we're interested in from a database update algorithm is the maximum number of updates we need to perform before the database D^t approximates x well with respect to the queries in $\mathcal{Q}$. To this end we define a database update algorithm as follows:

Definition 5.4 (Database Update Algorithm)**.** Let $U : \mathcal{D} \times \mathcal{Q} \times \mathbb{R} \to \mathcal{D}$ be an update rule and let $T : \mathbb{R} \to \mathbb{R}$ be a function. We say U is a $T(\alpha)$-*database update algorithm for query class* $\mathcal{Q}$ if for every database $x \in \mathbb{N}^{|\mathcal{X}|}$, every $(U, x, \mathcal{Q}, \alpha, L)$-database update sequence satisfies $L \leq T(\alpha)$.

Note that the definition of a $T(\alpha)$-database update algorithm implies that if U is a $T(\alpha)$-database update algorithm, then given any maximal $(U, x, \mathcal{Q}, \alpha, U)$-database update sequence, the final database D^L must satisfy $\max_{f \in \mathcal{Q}} \left| f(x) - f(D^L) \right| \leq \alpha$ or else there would exist

another query satisfying property 2 of Definition 5.3, and thus there would exist a $(U, x, \mathcal{Q}, \alpha, L+1)$-database update sequence, contradicting maximality. That is, the goal of a $T(\alpha)$ database update rule is to generate a maximal database update sequence, and the final data structure in a maximal database update sequence necessarily encodes the approximate answers to every query $f \in \mathcal{Q}$.

Now that we have defined database update algorithms, we can remark that what we really proved in Theorem 4.10 was that the Multiplicative Weights algorithm is a $T(\alpha)$-database update algorithm for $T(\alpha) = 4 \log |\mathcal{X}|/\alpha^2$.

Before we go on, let us build some intuition for what a database update algorithm is. A $T(\alpha)$-database update algorithm begins with some initial guess D^1 about what the true database x looks like. Because this guess is not based on any information, it is quite likely that D^1 and x bear little resemblance, and that there is some $f \in \mathcal{Q}$ that is able to distinguish between these two databases by at least α: that is, that $f(x)$ and $f(D^1)$ differ in value by at least α. What a database update algorithm does is to update its hypothesis D^t given evidence that its current hypothesis D^{t-1} is incorrect: at each stage, it takes as input some query in $\mathcal{Q}$ which distinguishes its current hypothesis from the true database, and then it outputs a new hypothesis. The parameter $T(\alpha)$ is an upper bound on the number of times that the database update algorithm will have to update its hypothesis: it is a promise that after at most $T(\alpha)$ distinguishing queries have been provided, the algorithm will finally have produced a hypothesis that looks like the true database with respect to $\mathcal{Q}$, at least up to error α.[1] For a database update algorithm, smaller bounds $T(\alpha)$ are more desirable.

Database Update Algorithms and Online Learning Algorithms: We remark that database update algorithms are essentially *online learning*

[1] Imagine that the database update algorithm is attempting to sculpt x out of a block of clay. Initially, its sculpture D^1 bears no resemblance to the true database: it is simply a block of clay. However, a helpful distinguisher points out to the sculptor places in which the clay juts out much farther than the true target database: the sculptor dutifully pats down those bumps. If the distinguisher always finds large protrusions, of magnitude at least α, the sculpture will be finished soon, and the distinguisher's time will not be wasted!

algorithms in the *mistake bound model*. In the setting of online learning, unlabeled examples arrive in some arbitrary order, and the learning algorithm must attempt to label them.

Background from Learning Theory. In the *mistake bound model of learning*, labeled examples $(x_i, y_i) \in \mathcal{X} \times \{0, 1\}$ arrive one at a time, in a potentially adversarial order. At time i, the learning algorithm A observes x_i, and must make a prediction $\hat{y}_i$ about the label for x_i. It then sees the true label y_i, and is said to *make a mistake* if its prediction was wrong: i.e., if $y_i \neq \hat{y}_i$. A learning algorithm A for a class of functions C is said to have a mistake bound of M, if for all $f \in C$, and for all adversarially selected sequences of examples $(x_1, f(x_1)), \ldots, (x_i, f(x_i)), \ldots$, A never makes more than M mistakes. Without loss of generality, we can think of such a learning algorithm as maintaining some hypothesis $\hat{f} : \mathcal{X} \to \{0, 1\}$ at all times, and updating it only when it makes a mistake. The adversary in this model is quite powerful — it can choose the sequence of labeled examples adaptively, knowing the current hypothesis of the learning algorithm, and its entire history of predictions. Hence, learning algorithms that have finite mistake bounds can be useful in extremely general settings.

It is not hard to see that mistake bounded online learning algorithms always exist for finite classes of functions C. Consider, for example, the *halving algorithm*. The halving algorithm initially maintains a set S of functions from C consistent with the examples that it has seen so far: Initially $S = C$. Whenever a new unlabeled example arrives, it predicts according to the majority vote of its consistent hypotheses: that is, it predicts label 1 whenever $|\{f \in S : f(x_i) = 1\}| \geq |S|/2$. Whenever it makes a mistake on an example x_i, it updates S by removing any inconsistent function: $S \leftarrow \{f \in S : f(x_i) = y_i\}$. Note that whenever it makes a mistake, the size of S is cut in half! So long as all examples are labeled by *some* function $f \in C$, there is at least one function $f \in C$ that is never removed from S. Hence, the halving algorithm has a mistake bound of $\log |C|$.

Generalizing beyond boolean labels, we can view database update algorithms as online learning algorithms in the mistake bound model:

here, examples that arrive are the queries (which may come in adversarial order). The labels are the approximate values of the queries when evaluated on the database. The database update algorithm hypothesis D^t makes a *mistake* on query f if $|f(D^t) - f(x)| \geq \alpha$, in which case we learn the label of f (that is, v_t) and allow the database update algorithm to update the hypothesis. Saying that an algorithm U is a $T(\alpha)$-database update algorithm is akin to saying that it has a mistake bound of $T(\alpha)$: no adversarially chosen sequence of queries can ever cause it to make more than $T(\alpha)$-mistakes. Indeed, the database update algorithms that we will see are taken from the online learning literature. The multiplicative weights mechanism is based on an online learning algorithm known as *Hedge*, which we have already discussed. The Median Mechanism (later in this section) is based on the *Halving Algorithm*, and the Perceptron algorithm is based (coincidentally) on an algorithm known as *Perceptron*. We won't discuss Perceptron here, but it operates by making *additive* updates, rather than the multiplicative updates used by multiplicative weights.

A database update algorithm for a class $\mathcal{Q}$ will be useful together with a corresponding *distinguisher*, whose job is to output a function that behaves differently on the true database x and the hypothesis D^t, that is, to point out a mistake.

Definition 5.5 ($(F(\varepsilon), \gamma)$-Private Distinguisher)**.** Let $\mathcal{Q}$ be a set of queries, let $\gamma \geq 0$ and let $F(\varepsilon) : \mathbb{R} \to \mathbb{R}$ be a function. An algorithm $\text{Distinguish}_\varepsilon : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{D} \to \mathcal{Q}$ is an $(F(\varepsilon), \gamma)$-Private Distinguisher for $\mathcal{Q}$ if for every setting of the privacy parameter ε, on every pair of inputs $x \in \mathbb{N}^{|\mathcal{X}|}$, $D \in \mathcal{D}$ it is $(\varepsilon, 0)$-differentially private with respect to x and it outputs an $f^* \in \mathcal{Q}$ such that $|f^*(x) - f^*(D)| \geq \max_{f \in \mathcal{Q}} |f(x) - f(D)| - F(\varepsilon)$ with probability at least $1 - \gamma$.

Remark 5.1. In machine learning, the goal is to find a function $f : \mathcal{X} \to \{0, 1\}$ from a class of functions $\mathcal{Q}$ that *best labels* a collection of labeled examples $(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{0, 1\}$. (Examples $(x, 0)$ are known as *negative examples*, and examples $(x, 1)$ are known as *positive examples*). Each example x_i has a *true* label y_i, and a function f *correctly labels* x_i if $f(x_i) = y_i$. An *agnostic learning algorithm* for a class $\mathcal{Q}$ is an algorithm that can find the function in $\mathcal{Q}$ that labels

all of the data points approximately as well as the best function in $\mathcal{Q}$, even if no function in $\mathcal{Q}$ can perfectly label them. Note that equivalently, an agnostic learning algorithm is one that maximizes the number of positive examples labeled 1 minus the number of negative examples labeled 1. Phrased in this way, we can see that a *distinguisher* as defined above is just an agnostic learning algorithm: just imagine that x contains all of the "positive" examples, and that y contains all of the "negative examples." (Note that it is ok of x and y are not disjoint — in the learning problem, the same example can occur with both a positive and a negative label, since agnostic learning does not require that any function perfectly label every example.) Finally, note also that for classes of linear queries $\mathcal{Q}$, a distinguisher is simply an optimization algorithm. Because for linear queries f, $f(x) - f(y) = f(x - y)$, a distinguisher simply seeks to find $\arg\max_{f \in \mathcal{Q}} |f(x - y)|$.

Note that, *a priori*, a differentially private distinguisher is a weaker object than a differentially private release algorithm: A distinguisher merely finds a query in a set $\mathcal{Q}$ with the approximately largest value, whereas a release algorithm must find the answer to every query in $\mathcal{Q}$. In the algorithm that follows, however, we reduce release to optimization.

We will first analyze the IC algorithm, and then instantiate it with a specific distinguisher and database update algorithm. What follows is a formal analysis, but the intuition for the mechanism is simple: we simply run the iterative database construction algorithm to construct a hypothesis that approximately matches x with respect to the queries $\mathcal{Q}$. If at each round our distinguisher succeeds in finding a query that has high discrepancy between the hypothesis database and the true database, then our database update algorithm will output a database that is β-accurate with respect to $\mathcal{Q}$. If the distinguisher ever fails to find such a query, then it must be that there are no such queries, and our database update algorithm has already learned an accurate hypothesis with respect to the queries of interest! This requires at most T iterations, and so we access the data only $2T$ times using $(\varepsilon_0, 0)$-differentially private methods (running the given distinguisher, and then checking its answer with the Laplace mechanism). Privacy will therefore follow from our composition theorems.