equilibrium of the game in question given players' reports. On the other hand, to do so, our suggested route to some player i must depend on the reported location/destination pairs of other players. This tension will pose a problem in terms of incentives: if we compute an equilibrium of the game given the reports of the players, an agent can potentially benefit by misreporting, causing us to compute an equilibrium of the wrong game.

This problem would be largely alleviated, however, if the report of agent i only has a tiny effect on the actions of agents $j \neq i$. In this case, agent i could hardly gain an advantage through his effect on other players. Then, assuming that everyone truthfully reported their type, the mechanism would compute an equilibrium of the correct game, and by definition, each agent i could do no better than follow the suggested equilibrium action. In other words, if we could compute an approximate equilibrium of the game under the constraint of *differential privacy*, then truthful reporting, followed by taking the suggested action of the coordination device would be a Nash equilibrium. A moment's reflection reveals that the goal of privately computing an equilibrium is not possible in small games, in which an agent's utility is a highly sensitive function of the actions (and hence utility functions) of the other agents. But what about in large games?

Formally, suppose we have an n player game with action set $\mathcal{A}$, and each agent with type t_i has a utility function $u_i : \mathcal{A}^n \to [0, 1]$. We say that this game is Δ-large if for all players $i \neq j$, vectors of actions $a \in \mathcal{A}^n$, and pairs of actions $a_j, a'_j \in \mathcal{A}$:

$$\left| u_i(a_j, a_{-j}) - u_i(a'_j, a_{-j}) \right| \leq \Delta.$$

In other words, if some agent j unilaterally changes his action, then his affect on the payoff of any other agent $i \neq j$ is at most Δ. Note that if agent j changes his own action, then his payoff can change arbitrarily. Many games are "large" in this sense. In the commuting example above, if Alice changes her route to work she may substantially increase or decrease her commute time, but will only have a minimal impact on the commute time of any other agent Bob. The results in this section are strongest for $\Delta = O(1/n)$, but hold more generally.

First we might ask whether we need privacy at all— could it be the case that in a large game, any algorithm which computes an equilibrium of a game defined by reported types has the stability property that we want? The answer is no. As a simple example, consider n people who must each choose whether to go to the beach (B) or the mountains (M). People privately know their types— each person's utility depends on his own type, his action, and the fraction of other people p who go to the beach. A Beach type gets a payoff of $10p$ if he visits the beach, and $5(1-p)$ if he visits the mountain. A mountain type gets a payoff $5p$ from visiting the beach, and $10(1-p)$ from visiting the mountain. Note that this is a large (i.e., low sensitivity) game — each player's payoffs are insensitive in the actions of others. Further, note that "everyone visits beach" and "everyone visits mountain" are both equilibria of the game, regardless of the realization of types. Consider the mechanism that attempts to implement the following social choice rule— "if the number of beach types is less than half the population, send everyone to the beach, and vice versa." It should be clear that if mountain types are just in the majority, then each mountain type has an incentive to misreport as a beach type; and vice versa. As a result, even though the game is "large" and agents' actions do not affect others' payoffs significantly, simply computing equilibria from reported type profiles does not in general lead to even approximately truthful mechanisms.

Nevertheless, it turns out to be possible to give a mechanism with the following property: it elicits the type t_i of each agent, and then computes an α-approximate correlated equilibrium of the game defined by the reported types.[2] (In some cases, it is possible to strengthen this result to compute an approximate *Nash equilibrium* of the underlying game.) It draws an action profile $a \in \mathcal{A}^n$ from the correlated equilibrium, and reports action a_i to each agent i. The algorithm has the guarantee that simultaneously for all players i, the joint distribution a_{-i} on reports to all players *other than* i is differentially private in

[2] A correlated equilibrium is defined by a joint distribution on profiles of actions, $\mathcal{A}^n$. For an action profile a drawn from the distribution, if agent i is told only a_i, then playing action a_i is a best response given the induced conditional distribution over a_{-i}. An α-approximate correlated equilibrium is one where deviating improves an agent's utility by at most α.

the reported type of agent i. When the algorithm computes a correlated equilibrium of the underlying game, this guarantee is sufficient for a restricted form of approximate truthfulness: agents who have the option to opt-in or opt-out of the mechanism (but not to misreport their type if they opt-in) have no disincentive to opt-out, because no agent i can substantially change the distribution on actions induced on *the other players* by opting out. Moreover, given that he opts in, no agent has incentive not to follow his suggested action, as his suggestion is part of a correlated equilibrium. When the mechanism computes a Nash equilibrium of the underlying game, then the mechanism becomes truthful even when agents have the ability to mis-report their type to the mechanism when they opt in.

More specifically, when these mechanisms compute an α-approximate Nash equilibrium while satisfying ϵ-differential privacy, every agent following the honest behavior (i.e., first opting in and reporting their true type, then following their suggested action) forms an $(2\epsilon + \alpha)$-approximate Nash equilibrium. This is because, by privacy, reporting your true type is a 2ϵ-approximate dominant strategy, and given that everybody reports their true type, the mechanism computes an α-approximate equilibrium of the true game, and hence by definition, following the suggested action is an α-approximate best response. There exist mechanisms for computing and α-approximate equilibrium in large games with $\alpha = O\left(\frac{1}{\sqrt{n\epsilon}}\right)$. Therefore, by setting $\epsilon = O\left(\frac{1}{n^{1/4}}\right)$, this gives an η-approximately truthful equilibrium selection mechanism for

$$\eta = 2\epsilon + \alpha = O\left(\frac{1}{n^{1/4}}\right).$$

In other words, it gives a mechanism for coordinating equilibrium behavior in large games that is asymptotically truthful in the size of the game, all without the need for monetary transfers.

10.2.3 Obtaining exact truthfulness

So far we have discussed mechanisms that are *asymptotically truthful* in large population games. However, what if we want to insist on mechanisms that are *exactly* dominant strategy truthful, while maintaining

some of the nice properties enjoyed by our mechanisms so far: for example, that the mechanisms do not need to be able to extract monetary payments? Can differential privacy help here? It can—in this section, we discuss a framework which uses differentially private mechanisms as a building block toward designing exactly truthful mechanisms without money.

The basic idea is simple and elegant. As we have seen, the exponential mechanism can often give excellent utility guarantees while preserving differential privacy. This doesn't yield an exactly truthful mechanism, but it gives every agent very little incentive to deviate from truthful behavior. What if we could pair this with a second mechanism which need not have good utility guarantees, but gives each agent a strict positive incentive to report truthfully, i.e., a mechanism that essentially only punishes non-truthful behavior? Then, we could randomize between running the two mechanisms. If we put enough weight on the punishing mechanism, then we inherit its strict-truthfulness properties. The remaining weight that is put on the exponential mechanism contributes to the utility properties of the final mechanism. The hope is that since the exponential mechanism is approximately strategy proof to begin with, the randomized mechanism can put small weight on the strictly truthful punishing mechanism, and therefore will have good utility properties.

To design punishing mechanisms, we will have to work in a slightly non-standard environment. Rather than simply picking an outcome, we can model a mechanism as picking an outcome, and then an agent as choosing a *reaction* to that outcome, which together define his utility. Mechanisms will then have the power to *restrict the reactions allowed by the agent based on his reported type*. Formally, we will work in the following framework:

Definition 10.4 (The Environment)**.** An environment is a set N of n players, a set of types $t_i \in \mathcal{T}$, a finite set $\mathcal{O}$ of outcomes, a set of reactions R and a utility function $u : T \times \mathcal{O} \times R \to [0, 1]$.

We write $r_i(t, s, \hat{R}_i) \in \arg\max_{r \in \hat{R}_i} u_i(t, s, r)$ to denote is optimal reaction among choices $\hat{R}_i \subseteq R$ to alternative s if he is of type t.

A direct revelation mechanism $\mathcal{M}$ defines a game which is played as follows:

1. Each player i reports a type $t'_i \in \mathcal{T}$.
2. The mechanism chooses an alternative $s \in \mathcal{O}$ and a subset $\hat{R}_i \subseteq R$ of reactions, for each player i.
3. Each player i chooses a reaction $r_i \in \hat{R}_i$ and experiences utility $u(t_i, s, r_i)$.

Agents play so as to maximize their own utility. Note that since there is no further interaction after the 3rd step, rational agents will pick $r_i = r_i(t_i, s, \hat{R}_i)$, and so we can ignore this as a strategic step. Let $\mathcal{R} = 2^R$. Then a mechanism is a randomized mapping $\mathcal{M} : \mathcal{T} \to \mathcal{O} \times \mathcal{R}^n$.

Let us consider the utilitarian welfare criterion: $F(t, s, r) = \frac{1}{n}\sum_{i=1}^{n} u(t_i, s, r_i)$, Note that this has sensitivity $\Delta = 1/n$, since each agent's utility lies in the range $[0, 1]$. Hence, if we simply choose an outcome s and allow each agent to play their best response reaction, the exponential mechanism is an ϵ-differentially private mechanism, which by Theorem 3.11, achieves social welfare at least $\mathrm{OPT} - O\left(\frac{\log|\mathcal{O}|}{\epsilon n}\right)$ with high probability. Let us denote this instantiation of the exponential mechanism, with quality score F, range $\mathcal{O}$ and privacy parameter ϵ, as $\mathcal{M}_\epsilon$.

The idea is to randomize between the exponential mechanism (with good social welfare properties) and a strictly truthful mechanism which punishes false reporting (but with poor social welfare properties). If we mix appropriately, then we will get an exactly truthful mechanism with reasonable social welfare guarantees.

Here is one such punishing mechanism which is simple, but not necessarily the best for a given problem:

Definition 10.5. The commitment mechanism $M^P(t')$ selects $s \in \mathcal{O}$ uniformly at random and sets $\hat{R}_i = \{r_i(t'_i, s, R_i)\}$, i.e., it picks a random outcome and forces everyone to react as if their reported type was their true type.

Define the *gap* of an environment as

$$\gamma = \min_{i, t_i \neq t'_i, t_{-i}} \max_{s \in \mathcal{O}} \left(u(t_i, s, r_i(t_i, s, R_i)) - u(t_i, s, r_i(t'_i, s, R_i)) \right),$$

i.e., γ is a lower bound over players and types of the worst-case cost (over s) of mis-reporting. Note that for each player, this worst-case is realized with probability at least $1/|\mathcal{O}|$. Therefore we have the following simple observation:

Lemma 10.2. For all i, t_i, t_i', t_{-i}:

$$u(t_i, \mathcal{M}^P(t_i, t_{-i})) \geq u(t_i, \mathcal{M}^P(t_i', t_{-i})) + \frac{\gamma}{|\mathcal{O}|}.$$

Note that the commitment mechanism is strictly truthful: every individual has at least a $\frac{\gamma}{|\mathcal{O}|}$ incentive not to lie.

This suggests an exactly truthful mechanism with good social welfare guarantees:

Definition 10.6. The punishing exponential mechanism $\mathcal{M}_\epsilon^P(t)$ defined with parameter $0 \leq q \leq 1$ selects the exponential mechanism $\mathcal{M}_\epsilon(t)$ with probability $1-q$ and the punishing mechanism $\mathcal{M}^P(t)$ with complementary probability q.

Observe that by linearity of expectation, we have for all t_i, t_i', t_{-i}:

$$\begin{aligned}
u(t_i, \mathcal{M}_\epsilon^P(t_i, t_{-i})) &= (1-q) \cdot u(t_i, \mathcal{M}_\epsilon(t_i, t_{-i})) + q \cdot u(t_i, \mathcal{M}^P(t_i, t_{-i})) \\
&\geq (1-q)\left(u(t_i, \mathcal{M}_\epsilon(t_i', t_{-i})) - 2\epsilon\right) \\
&\quad + q\left(u(t_i, \mathcal{M}^P(t_i', t_{-i})) + \frac{\gamma}{|\mathcal{O}|}\right) \\
&= u(t_i, \mathcal{M}_\epsilon^P(t_i', t_{-i})) - (1-q)2\epsilon + q\frac{\gamma}{|\mathcal{O}|} \\
&= u(t_i, \mathcal{M}_\epsilon^P(t_i', t_{-i})) - 2\epsilon + q\left(2\epsilon + \frac{\gamma}{|\mathcal{O}|}\right).
\end{aligned}$$

The following two theorems show incentive and social welfare properties of this mechanism.

Theorem 10.3. If $2\epsilon \leq \frac{q\gamma}{|\mathcal{O}|}$ then $\mathcal{M}_\epsilon^P$ is strictly truthful.

Note that we also have utility guarantees for this mechanism. Setting the parameter q so that we have a truthful mechanism:

$$\begin{aligned}
&\mathbb{E}_{s,\hat{R}\sim\mathcal{M}_\epsilon^P}[F(t,s,r(t,s,\hat{R}))] \\
&\quad\geq (1-q)\cdot\mathbb{E}_{s,\hat{R}\sim\mathcal{M}_\epsilon}[F(t,s,r(t,s,\hat{R}))] \\
&\quad= \left(1-\frac{2\epsilon|\mathcal{O}|}{\gamma}\right)\cdot\mathbb{E}_{s,\hat{R}\sim\mathcal{M}_\epsilon}[F(t,s,r(t,s,\hat{R}))] \\
&\quad\geq \left(1-\frac{2\epsilon|\mathcal{O}|}{\gamma}\right)\cdot\left(\max_{t,s,r}F(t,s,r)-O\left(\frac{1}{\epsilon n}\log|\mathcal{O}|\right)\right) \\
&\quad\geq \max_{t,s,r}F(t,s,r)-\frac{2\epsilon|\mathcal{O}|}{\gamma}-O\left(\frac{1}{\epsilon n}\log|\mathcal{O}|\right).
\end{aligned}$$

Setting

$$\epsilon\in O\left(\sqrt{\frac{\log|\mathcal{O}|\gamma}{|\mathcal{O}|n}}\right)$$

we find:

$$\mathbb{E}_{s,\hat{R}\sim\mathcal{M}_\epsilon^P}[F(t,s,r(t,s,\hat{R}))]\geq\max_{t,s,r}F(t,s,r)-O\left(\sqrt{\frac{|\mathcal{O}|\log|\mathcal{O}|}{\gamma n}}\right).$$

Note that in this calculation, we assume that $\epsilon\leq\gamma/(2|\mathcal{O}|)$ so that $q=\frac{2\epsilon|\mathcal{O}|}{\gamma}\leq 1$ and the mechanism is well defined. This is true for sufficiently large n. That is, we have shown:

Theorem 10.4. For sufficiently large n, M_ϵ^P achieves social welfare at least

$$\mathrm{OPT}-O\left(\sqrt{\frac{|\mathcal{O}|\log|\mathcal{O}|}{\gamma n}}\right).$$

Note that this mechanism is truthful without the need for payments!

Let us now consider an application of this framework: the facility location game. Suppose that a city wants to build k hospitals to minimize the average distance between each citizen and their closest hospital. To simplify matters, we make the mild assumption that the city is built on a discretization of the unit line.[3] Formally, let

[3] If this is not the case, we can easily raze and then re-build the city.

$L(m) = \{0, \frac{1}{m}, \frac{2}{m}, \ldots, 1\}$ denote the discrete unit line with step-size $1/m$. $|L(m)| = m+1$. Let $\mathcal{T} = R_i = L(m)$ for all i and let $|\mathcal{O}| = L(m)^k$. Define the utility of agent i to be:

$$u(t_i, s, r_i) = \begin{cases} -|t_i - r_i|, & \text{If } r_i \in s; \\ -1, & \text{otherwise.} \end{cases}$$

In other words, agents are associated with points on the line, and an outcome is an assignment of a location on the line to each of the k facilities. Agents can react to a set of facilities by deciding which one to go to, and their cost for such a decision is the distance between their own location (i.e., their type) and the facility that they have chosen. Note that $r_i(t_i, s)$ is here the closest facility $r_i \in s$.

We can instantiate Theorem 10.4. In this case, we have: $|\mathcal{O}| = (m+1)^k$ and $\gamma = 1/m$, because any two positions $t_i \neq t_i'$ differ by at least $1/m$. Hence, we have:

Theorem 10.5. M_ϵ^P instantiated for the facility location game is strictly truthful and achieves social welfare at least:

$$\text{OPT} - O\left(\sqrt{\frac{km(m+1)^k \log m}{n}}\right).$$

This is already very good for small numbers of facilities k, since we expect that $\text{OPT} = \Omega(1)$.

10.3 Mechanism design for privacy aware agents

In the previous section, we saw that differential privacy can be useful as a tool to design mechanisms, *for agents who care only about the outcome chosen by the mechanism*. We here primarily viewed privacy as a tool to accomplish goals in traditional mechanism design. As a side affect, these mechanisms also preserved the privacy of the reported player types. Is this itself a worthy goal? *Why* might we want our mechanisms to preserve the privacy of agent types?

A bit of reflection reveals that agents might care about privacy. Indeed, basic introspection suggests that in the real world, agents value the ability to keep certain "sensitive" information private, for example,

health information or sexual preferences. In this section, we consider the question of how to model this value for privacy, and various approaches taken in the literature.

Given that agents might have preferences for privacy, it is worth considering the design of mechanisms that preserve privacy *as an additional goal*, even for tasks such as welfare maximization that we can already solve non-privately. As we will see, it is indeed possible to generalize the VCG mechanism to *privately* approximately optimize social welfare in *any* social choice problem, with a smooth trade-off between the privacy parameter and the approximation parameter, all while guaranteeing exact dominant strategy truthfulness.

However, we might wish to go further. In the presence of agents with preferences for privacy, if we wish to design truthful mechanisms, we must somehow model their preferences for privacy in their utility function, and then design mechanisms which are truthful with respect to these new "privacy aware" utility functions. As we have seen with differential privacy, it is most natural to model privacy as a property of the mechanism itself. Thus, our utility functions are not merely functions of the outcome, but functions of the outcome and of the mechanism itself. In almost all models, agent utilities for outcomes are treated as linearly separable, that is, we will have for each agent i,

$$u_i(o, \mathcal{M}, t) \equiv \mu_i(o) - c_i(o, \mathcal{M}, t).$$

Here $\mu_i(o)$ represents agent is utility for outcome o and $c_i(o, \mathcal{M}, t)$ the (privacy) cost that agent i experiences when outcome o is chosen with mechanism $\mathcal{M}$.

We will first consider perhaps the simplest (and most naïve) model for the privacy cost function c_i. Recall that for $\epsilon \ll 1$, differential privacy promises that for each agent i, and for every possible utility function f_i, type vector $t \in \mathcal{T}^n$, and deviation $t'_i \in \mathcal{T}$:

$$|\mathbb{E}_{o \sim M(t_i, t_{-i})}[f_i(o)] - \mathbb{E}_{o \sim M(t'_i, t_{-i})}[f_i(o)]| \leq 2\epsilon \mathbb{E}_{o \sim M(t)}[f_i(o)].$$

If we view f_i as representing the "expected future utility" for agent i, it is therefore natural to model agent i's cost for having his data used in an ϵ-differentially private computation as being linear in ϵ. That is,

we think of agent i as being parameterized by some value $v_i \in \mathbb{R}$, and take:

$$c_i(o, \mathcal{M}, t) = \epsilon v_i,$$

where ϵ is the smallest value such that $\mathcal{M}$ is ϵ-differentially private. Here we imagine v_i to represent a quantity like $\mathbb{E}_{o \sim M(t)}[f_i(o)]$. In this setting, c_i does not depend on the outcome o or the type profile t.

Using this naïve privacy measure, we discuss a basic problem in private data analysis: how to collect the data, when the owners of the data value their privacy and insist on being compensated for it. In this setting, there is no "outcome" that agents value, other than payments, there is only dis-utility for privacy loss. We will then discuss shortcomings of this (and other) measures of the dis-utility for privacy loss, as well as privacy in more general mechanism design settings when agents *do* have utility for the outcome of the mechanism.

10.3.1 A private generalization of the VCG mechanism

Suppose we have a general social choice problem, defined by an outcome space $\mathcal{O}$, and a set of agents N with arbitrary preferences over the outcomes given by $u_i : \mathcal{O} \to [0, 1]$. We might want to choose an outcome $o \in \mathcal{O}$ to maximize the *social welfare* $F(o) = \frac{1}{n} \sum_{i=1}^{n} u_i(o)$. It is well known that in any such setting, the *VCG* mechanism can implement the outcome o^* which exactly maximizes the social welfare, while charging payments that make truth-telling a dominant strategy. What if we want to achieve the same result, while also preserving privacy? How must the privacy parameter ϵ trade off with our approximation to the optimal social welfare?

Recall that we could use the exponential mechanism to choose an outcome $o \in \mathcal{O}$, with quality score F. For privacy parameter ϵ, this would give a distribution $\mathcal{M}_\epsilon$ defined to be $\Pr[\mathcal{M}_\epsilon = o] \propto \exp\left(\frac{\epsilon F(o)}{2n}\right)$. Moreover, this mechanism has good social welfare properties: with probability $1 - \beta$, it selects some o such that: $F(o) \geq F(o^*) - \frac{2}{\epsilon n}\left(\ln \frac{|\mathcal{O}|}{\beta}\right)$. But as we saw, differential privacy only gives ϵ-approximate truthfulness.

However, it can be shown that $\mathcal{M}_\epsilon$ is the solution to the following exact optimization problem:

$$\mathcal{M}_\epsilon = \arg\max_{\mathcal{D}\in\Delta\mathcal{O}} \left(\mathbb{E}_{o\sim\mathcal{D}}[F(o)] + \frac{2}{\epsilon n} H(\mathcal{D}) \right),$$

where H represents the *Shannon Entropy* of the distribution $\mathcal{D}$. In other words, the exponential mechanism is the distribution which exactly maximizes the expected social welfare, *plus* the entropy of the distribution weighted by $2/(\epsilon n)$. This is significant for the following reason: it is known that any mechanism that *exactly* maximizes expected player utilities in any finite range (known as maximal in distributional range mechanisms) can be paired with payments to be made exactly dominant strategy truthful. The exponential mechanism is the distribution that *exactly* maximizes expected social welfare, plus entropy. In other words, if we imagine that we have added a single additional player whose utility is exactly the entropy of the distribution, then the exponential mechanism is maximal in distributional range. Hence, it can be paired with payments that make truthful reporting a dominant strategy for all players — in particular, for the n real players. Moreover, it can be shown how to charge payments in such a way as to preserve privacy. The upshot is that for any social choice problem, the social welfare can be approximated in a manner that both preserves differential privacy, and is exactly truthful.

10.3.2 The sensitive surveyor's problem

In this section, we consider the problem of a data analyst who wishes to conduct a study using the private data of a collection of individuals. However, he must *convince* these individuals to hand over their data! Individuals experience costs for privacy loss. The data analyst can mitigate these costs by guaranteeing differential privacy and compensating them for their loss, while trying to get a representative sample of data.

Consider the following stylized problem of the sensitive surveyor Alice. She is tasked with conducting a survey of a set of n individuals N, to determine what proportion of the individuals $i \in N$ satisfy some property $P(i)$. Her ultimate goal is to discover the true value of this statistic, $s = \frac{1}{n}|\{i \in N : P(i)\}|$, but if that is not possible, she will be

satisfied with some estimate $\hat{s}$ such that the error, $|\hat{s} - s|$, is minimized. We will adopt a notion of accuracy based on large deviation bounds, and say that a surveying mechanism is α-accurate if $\Pr[|\hat{s} - s| \geq \alpha] \leq \frac{1}{3}$. The inevitable catch is that individuals value their privacy and will not participate in the survey for free. Individuals experience some *cost* as a function of their loss in privacy when they interact with Alice, and must be compensated for this loss. To make matters worse, these individuals are rational (i.e., selfish) agents, and are apt to misreport their costs to Alice if doing so will result in a financial gain. This places Alice's problem squarely in the domain of mechanism design, and requires Alice to develop a scheme for trading off statistical accuracy with cost, all while managing the incentives of the individuals.

As an aside, this stylized problem is broadly relevant to any organization that makes use of collections of potentially sensitive data. This includes, for example, the use of search logs to provide search query completion and the use of browsing history to improve search engine ranking, the use of social network data to select display ads and to recommend new links, and the myriad other data-driven services now available on the web. In all of these cases, value is being derived from the statistical properties of a collection of sensitive data in exchange for some payment.[4]

Collecting data in exchange for some fixed price could lead to a biased estimate of population statistics, because such a scheme will result in collecting data only from those individuals who value their privacy less than the price being offered. However, without interacting with the agents, we have no way of knowing what price we can offer so that we will have broad enough participation to guarantee that the answer we collect has only small bias. To obtain an accurate estimate of the statistic, it is therefore natural to consider buying private data using an auction — as a means of discovering this price. There are two obvious obstacles which one must confront when conducting an auction for private data, and an additional obstacle which is less obvious but more insidious. The first obstacle is that one must have a quantitative

[4]The payment need not be explicit and/or dollar denominated — for example, it may be the use of a "free" service.

formalization of "privacy" which can be used to measure agents' costs under various operations on their data. Here, differential privacy provides an obvious tool. For small values of ϵ, because $\exp(\epsilon) \approx (1 + \epsilon)$, and so as discussed earlier, a simple (but possibly naive) first cut at a model is to view each agent as having some *linear* cost for participating in a private study. We here imagine that each agent i has an unknown value for privacy v_i, and experiences a cost $c_i(\epsilon) = \epsilon v_i$ when his private data is used in an ϵ-differentially private manner.[5] The second obstacle is that our objective is to trade off with *statistical accuracy*, and the latter is not well-studied objective in mechanism design.

The final, more insidious obstacle, is that an individual's cost for privacy loss may be highly correlated with his private data itself! Suppose we only know Bob has a high value for privacy of his AIDS status, but do not explicitly know his AIDS status itself. This is already disclosive because Bob's AIDS status is likely correlated with his value for privacy, and knowing that he has a high cost for privacy lets us update our belief about what his private data might be. More to the point, suppose that in the first step of a survey of AIDS prevalence, we ask each individual to report their value for privacy, with the intention of then running an auction to choose which individuals to buy data from. If agents report truthfully, we may find that the reported values naturally form two clusters: low value agents, and high value agents. In this case, we may have learned something about the population statistic even before collecting any data or making any payments— and therefore, the agents will have already experienced a cost. As a result, the agents may misreport their value, which could introduce a bias in the survey results. This phenomenon makes direct revelation mechanisms problematic, and distinguishes this problem from classical mechanism design.

Armed with a means of quantifying an agent i's loss for allowing his data to be used by an ϵ-differentially-private algorithm ($c_i(\epsilon) = \epsilon v_i$), we are almost ready to describe results for the sensitive surveyor's problem. Recall that a differentially private algorithm is some mapping $M : \mathcal{T}^n \to \mathcal{O}$, for a general type space $\mathcal{T}$. It remains to define what

[5] As we will discuss later, this assumption can be problematic.

exactly the type space $\mathcal{T}$ is. We will consider two models. In both models, we will associate with each individual a bit $b_i \in \{0,1\}$ which represents whether they satisfy the sensitive predicate $P(i)$, as well as a value for privacy $v_i \in \mathbb{R}^+$.

1. In the *insensitive value model*, we calculate the ϵ parameter of the private mechanism by letting the type space be $\mathcal{T} = \{0,1\}$: i.e., we measure privacy cost only with respect to how the mechanism treats the sensitive bit b_i, and ignore how it treats the reported values for privacy, v_i.[6]
2. In the *sensitive value model*, we calculate the ϵ parameter of the private mechanism by letting the type space be $\mathcal{T} = (\{0,1\} \times \mathbb{R}^+)$: i.e., we measure privacy with respect to how it treats the pair (b_i, v_i) for each individual.

Intuitively, the insensitive value model treats individuals as ignoring the potential privacy loss due to correlations between their values for privacy and their private bits, whereas the sensitive value model treats individuals as assuming these correlations are worst-case, i.e., their values v_i are just as disclosive as their private bits b_i. It is known that in the insensitive value model, one can derive approximately optimal direct revelation mechanisms that achieve high accuracy and low cost. By contrast, in the *sensitive value model*, no individually rational direct revelation mechanism can achieve any non-trivial accuracy.

This leaves a somewhat unsatisfying state of affairs. The sensitive value model captures the delicate issues that we really want to deal with, and yet there we have an impossibility result! Getting around this result in a satisfying way (e.g., by changing the model, or the powers of the mechanism) remains an intriguing open question.

10.3.3 Better measures for the cost of privacy

In the previous section, we took the naive modeling assumption that the cost experienced by participation in an ϵ-differentially private mechanism M was $c_i(o, \mathcal{M}, t) = \epsilon v_i$ for some numeric value v_i. This measure

[6] That is, the part of the mapping dealing with reported values need not be differentially private.

is problematic for several reasons. First, although differential privacy promises that any agent's loss in utility is *upper bounded* by a quantity that is (approximately) linear in ϵ, there is no reason to believe that agents' costs are *lower bounded* by such a quantity. That is, while taking $c_i(o, \mathcal{M}, t) \leq \epsilon v_i$ is well motivated, there is little support for making the inequality an equality. Second, (it turns out) *any* privacy measure which is a deterministic function only of ϵ (not just a linear function) leads to problematic behavioral predictions.

So how else might we model c_i? One natural measure is the *mutual information* between the reported type of agent i, and the outcome of the mechanism. For this to be well defined, we must be in a world where each agent's type t_i is drawn from a known prior, $t_i \sim \mathcal{T}$. Each agent's strategy is a mapping $\sigma_i : \mathcal{T} \to \mathcal{T}$, determining what type he reports, given his true type. We could then define

$$c_i(o, \mathcal{M}, \sigma) = I(\mathcal{T}; \mathcal{M}(t_{-i}, \sigma(\mathcal{T})),$$

where I is the mutual information between the random variable $\mathcal{T}$ representing the prior on agent is type, and $\mathcal{M}(t_{-i}, \sigma(\mathcal{T}))$, the random variable representing the outcome of the mechanism, given agent is strategy.

This measure has significant appeal, because it represents how "related" the output of the mechanism is to the true type of agent i. However, in addition to requiring a prior over agent types, observe an interesting paradox that results from this measure of privacy loss. Consider a world in which there are two kinds of sandwich breads: Rye (R), and Wheat (W). Moreover, in this world, sandwich preferences are highly embarrassing and held private. The prior on types $\mathcal{T}$ is uniform over R and W, and the mechanism $\mathcal{M}$ simply gives agent i a sandwich of the type that he purports to prefer. Now consider two possible strategies, σ_{truthful} and σ_{random}. σ_{truthful} corresponds to truthfully reporting sandwich preferences (and subsequently leads to eating the preferred sandwich type), while σ_{random} randomly reports independent of true type (and results in the preferred sandwich only half the time). The cost of using the random strategy is $I(\mathcal{T}; \mathcal{M}(t_{-i}, \sigma_{\text{random}}(\mathcal{T})) = 0$, since the output is independent of agent i's type. On the other hand, the cost of truthfully reporting is $I(\mathcal{T}; \mathcal{M}(t_{-i}, \sigma_{\text{truthful}}(\mathcal{T})) = 1$, since

the sandwich outcome is now the identity function on agent is type. However, from the perspective of any outside observer, the two strategies are indistinguishable! In both cases, agent i receives a uniformly random sandwich. Why then should anyone choose the random strategy? So long as an adversary *believes* they are choosing randomly, they should choose the honest strategy.

Another approach, which does not need a prior on agent types, is as follows. We may model agents as having a cost function c_i that satisfies:

$$|c_i(o, \mathcal{M}, t)| = \ln \left(\max_{t_i, t_i' \in \mathcal{T}} \frac{\Pr[\mathcal{M}(t_i, t_{-i}) = o]}{\Pr[\mathcal{M}(t_i', t_{-i}) = o]} \right).$$

Note that if $\mathcal{M}$ is ϵ-differentially private, then

$$\max_{t \in \mathcal{T}^n} \max_{o \in \mathcal{O}} \max_{t_i, t_i' \in \mathcal{T}} \ln \left(\frac{\Pr[\mathcal{M}(t_i, t_{-i}) = o]}{\Pr[\mathcal{M}(t_i', t_{-i}) = o]} \right) \leq \epsilon.$$

That is, we can view differential privacy as bounding the *worst-case* privacy loss over all possible outcomes, whereas the measure proposed here considers only the privacy loss for the outcome o (and type vector t) actually realized. Thus, for any differentially private mechanism $\mathcal{M}$, $|c_i(o, \mathcal{M}, t)| \leq \epsilon$ for all o, t, but it will be important that the cost can vary by outcome.

We can then consider the following allocation rule for maximizing social welfare $F(o) = \sum_{i=1}^{n} u_i(o)$.[7] We discuss the case when $|\mathcal{O}| = 2$ (which does not require payments), but it is possible to analyze the general case (with payments), which privately implements the VCG mechanism for any social choice problem.

1. For each outcome $o \in \mathcal{O}$, choose a random number r_o from the distribution $\Pr[r_o = x] \propto \exp(-\epsilon|x|)$.
2. Output $o^* = \arg\max_{o \in \mathcal{O}}(F(o) + r_o)$.

The above mechanism is ϵ-differentially private, and that it is truthful for privacy aware agents, so long as for each agent i, and for the two outcomes $o, o' \in \mathcal{O}$, $|\mu_i(o) - \mu_i(o')| > 2\epsilon$. Note that this will be true

[7] This allocation rule is extremely similar to, and indeed can be modified to be identical to the exponential mechanism.

for small enough ϵ so long as agent utilities for outcomes are distinct. The analysis proceeds by considering an arbitrary fixed realization of the random variables r_o, and an arbitrary deviation t_i' from truthful reporting for the ith agent. There are two cases: In the first case, the deviation does not change the outcome o of the mechanism. In this case, *neither* the agent's utility for the outcome μ_i, nor his cost for privacy loss c_i change at all, and so the agent does not benefit from deviating. In the second case, if the outcome changes from o to o' when agent i deviates, it must be that $\mu_i(o') < \mu_i(o) - 2\epsilon$. By differential privacy, however, $|c_i(o, \mathcal{M}, t) - c_i(o', \mathcal{M}, t)| \leq 2\epsilon$, and so the change in privacy cost cannot be enough to make it beneficial.

Finally, the most conservative approach to modeling costs for privacy generally considered is as follows. Given an ϵ-differentially private mechanism $\mathcal{M}$, assume only that

$$c_i(o, \mathcal{M}, t) \leq \epsilon v_i,$$

for some number v_i. This is similar to the linear cost functions that we considered earlier, but crucially, here we assume only an upper bound. This assumption is satisfied by all of the other models for privacy cost that we have considered thus far. It can be shown that many mechanisms that combine a differentially private algorithm with a punishing mechanism that has the ability to restrict user choices, like those that we considered in Section 10.2.3, maintain their truthfulness properties in the presence of agents with preferences for privacy, so long as the values v_i are bounded.

10.4 Bibliographical notes

This section is based off of a survey of Pai and Roth [70] and a survey of Roth [73]. The connections between differential privacy and mechanism design were first suggested by Jason Hartline and investigated by McSherry and Talwar in their seminal work, "Mechanism Design via Differential Privacy" [61], where they considered the application of differential privacy to designing approximately truthful digital goods auctions. The best result for exactly truthful mechanisms in the digital goods setting is due to Balcan et al. [2].

The problem of designing exactly truthful mechanisms using differential privacy as a tool was first explored by Nissim, Smorodinsky, and Tennenholtz in [69], who also first posed a criticism as differential privacy (by itself) used as a solution concept. The example in this section of using differential privacy to obtain exactly truthful mechanisms is taken directly from [69]. The sensitive surveyors problem was first considered by Ghosh and Roth [36], and expanded on by [56, 34, 75, 16]. Fleischer and Lyu [34] consider the Bayesian setting discussed in this section, and Ligett and Roth [56] consider the worst-case setting with take-it-or-leave-it offers, both in an attempt to get around the impossibility result of [36]. Ghosh and Ligett consider a related model in which participation decisions (and privacy guarantees) are determined only in equilibrium [35].

The question of conducting mechanism design in the presence of agents who explicitly value privacy as part of their utility function was first raised by the influential work of Xiao [85], who considered (among other measures for privacy cost) the mutual information cost function. Following this, Chen et al. [15] and Nissim et al. [67] showed how in two distinct models, truthful mechanisms can sometimes be designed even for agents who value privacy. Chen Chong, Kash, Moran, and Vadhan considered the outcome-based cost function that we discussed in this section, and Nissim, Orlandi, and Smorodinsky considered the conservative model of only upper bounding each agent's cost by a linear function in $\epsilon>$ The "sandwich paradox" of valuing privacy according to mutual information is due to Nissim, Orlandi, and Smorodinsky.

Huang and Kannan proved that the exponential mechanism could be made exactly truthful with the addition of payments [49]. Kearns Pai, Roth, and Ullman showed how differential privacy could be used to derive asymptotically truthful equilibrium selection mechanisms [54] by privately computing correlated equilibria in large games. These results were strengthened by Rogers and Roth [71], who showed how to privately compute approximate *Nash* equilibria in large congestion games, which leads to stronger incentive properties of the mechanism. Both of these papers use the solution concept of "Joint Differential Privacy,"

which requires that for every player i, the joint distribution on messages sent to *other* players $j \neq i$ be differentially private in is report. This solution concept has also proven useful in other settings of private mechanism design settings, including an algorithm for computing private matchings by Hsu et al. [47].

11

Differential Privacy and Machine Learning

One of the most useful tasks in data analysis is machine learning: the problem of automatically finding a simple rule to accurately predict certain unknown characteristics of never before seen data. Many machine learning tasks can be performed under the constraint of differential privacy. In fact, the constraint of privacy is not necessarily at odds with the goals of machine learning, both of which aim to extract information from the distribution from which the data was drawn, rather than from individual data points. In this section, we survey a few of the most basic results on private machine learning, without attempting to cover this large field completely.

The goal in machine learning is very often similar to the goal in private data analysis. The *learner* typically wishes to learn some simple rule that explains a data set. However, she wishes this rule to generalize — that is, it should be that the rule she learns not only correctly describes the data that she has on hand, but that it should also be able to correctly describe *new* data that is drawn from the same distribution. Generally, this means that she wants to learn a rule that captures distributional information about the data set on hand, in a way that does not depend too specifically on any single data point. Of

course, this is exactly the goal of private data analysis — to reveal *distributional information* about the private data set, without revealing too much about any single individual in the dataset. It should come as no surprise then that machine learning and private data analysis are closely linked. In fact, as we will see, we are often able to perform private machine learning *nearly as accurately, with nearly the same number of examples* as we can perform non-private machine learning.

Let us first briefly define the problem of machine learning. Here, we will follow Valiant's *PAC* (Or *Probably Approximately Correct*) model of machine learning. Let $\mathcal{X} = \{0,1\}^d$ be the domain of "unlabeled examples." Think of each $x \in \mathcal{X}$ as a vector containing d boolean attributes. We will think of vectors $x \in \mathcal{X}$ as being paired with *labels* $y \in \{0,1\}$.

Definition 11.1. A *labeled example* is a pair $(x, y) \in \mathcal{X} \times \{0,1\}$: a vector paired with a label.

A learning problem is defined as a distribution $\mathcal{D}$ over labeled examples. The goal will to be to find a function $f : \mathcal{X} \to \{0,1\}$ that correctly labels almost all of the examples drawn from the distribution.

Definition 11.2. Given a function $f : \mathcal{X} \to \{0,1\}$ and a distribution $\mathcal{D}$ over labeled examples, the *error rate* of f on $\mathcal{D}$ is:

$$\mathrm{err}(f, \mathcal{D}) = \Pr_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$$

We can also define the error rate of f over a finite sample D:

$$\mathrm{err}(f, D) = \frac{1}{|D|}|\{(x, y) \in D : f(x) \neq y\}|.$$

A learning *algorithm* gets to observe some number of labeled examples drawn from $\mathcal{D}$, and has the goal of finding a function f with as small an error rate as possible when measured on $\mathcal{D}$. Two parameters in measuring the quality of a learning algorithm are its running time, and the number of examples it needs to see in order to find a good hypothesis.

Definition 11.3. An algorithm A is said to PAC-learn a class of functions C over d dimensions if for every $\alpha, \beta > 0$, there exists an

$m = \text{poly}(d, 1/\alpha, \log(1/\beta))$ such that for every distribution $\mathcal{D}$ over labeled examples, A takes as input m labeled examples drawn from $\mathcal{D}$ and outputs a hypothesis $f \in C$ such that with probability $1 - \beta$:

$$\text{err}(f, \mathcal{D}) \leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + \alpha$$

If $\min_{f^* \in C} \text{err}(f^*, \mathcal{D}) = 0$, the learner is said to operate in the *realizable* setting (i.e., there exists some function in the class which perfectly labels the data). Otherwise, the learner is said to operate in the *agnostic* setting. If A also has run time that is polynomial in $d, 1/\alpha$, and $\log(1/\beta)$, then the learner is said to be *efficient*. If there is an algorithm which PAC-learns C, then C is said to be PAC-learnable.

The above definition of learning allows the learner to have direct access to labeled examples. It is sometimes also useful to consider models of learning in which the algorithm only has oracle access to some noisy information about $\mathcal{D}$.

Definition 11.4. A *statistical query* is some function $\phi : \mathcal{X} \times \{0, 1\} \to [0, 1]$. A *statistical query oracle* for a distribution over labeled examples $\mathcal{D}$ with tolerance τ is an oracle $\mathcal{O}^\tau_\mathcal{D}$ such that for every statistical query ϕ:

$$\left| \mathcal{O}^\tau_\mathcal{D}(\phi) - \mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x, y)] \right| \leq \tau$$

In other words, an SQ oracle takes as input a statistical query ϕ, and outputs some value that is guaranteed to be within $\pm\tau$ of the expected value of ϕ on examples drawn from $\mathcal{D}$.

The statistical query model of learning was introduced to model the problem of learning in the presence of noise.

Definition 11.5. An algorithm A is said to SQ-learn a class of functions C over d dimensions if for every $\alpha, \beta > 0$ there exists an $m = \text{poly}(d, 1/\alpha, \log(1/\beta))$ such that A makes at most m queries of tolerance $\tau = 1/m$ to $\mathcal{O}^\tau_\mathcal{D}$, and with probability $1-\beta$, outputs a hypothesis $f \in C$ such that:

$$\text{err}(f, \mathcal{D}) \leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + \alpha$$

Note that an SQ learning algorithm does not get any access to $\mathcal{D}$ except through the SQ oracle. As with PAC learning, we can talk about an SQ learning algorithm operating in either the realizable or the agnostic setting, and talk about the computational efficiency of the learning algorithm. We say that a class C is SQ learnable if there exists an SQ learning algorithm for C.

11.1 The sample complexity of differentially private machine learning

Perhaps the first question that one might ask, with respect to the relationship between privacy and learning, is "When is it possible to privately perform machine learning"? In other words, you might ask for a PAC learning algorithm that takes as input a dataset (implicitly assumed to be sampled from some distribution $\mathcal{D}$), and then privately output a hypothesis f that with high probability has low error over the distribution. A more nuanced question might be, "How many *additional* samples are required to privately learn, as compared with the number of samples *already required* to learn without the constraint of differential privacy?" Similarly, "How much additional run-time is necessary to privately learn, as compared with the run-time required to learn non-privately?" We will here briefly sketch known results for $(\varepsilon, 0)$-differential privacy. In general, better results for (ε, δ)-differential privacy will follow from using the advanced composition theorem.

A foundational *information theoretic result* in private machine learning is that private PAC learning is possible with a polynomial number of samples if and only if non-private PAC learning is possible with a polynomial number of samples, even in the agnostic setting. In fact, the increase in sample complexity necessary is relatively small — however, this result does not preserve *computational efficiency*. One way to do this is directly via the exponential mechanism. We can instantiate the exponential mechanism with a range $R = C$, equal to the class of queries to be learned. Given a database D, we can use the quality score $q(f, D) = -\frac{1}{|D|}|\{(x, y) \in D : f(x) \neq y\}|$: i.e., we seek to minimize the fraction of misclassified examples in the private dataset. This is clearly

a $1/n$ sensitive function of the private data, and so we have via our utility theorem for the exponential mechanism that with probability $1-\beta$, this mechanism returns a function $f \in C$ that correctly labels an $\text{OPT} - \frac{2(\log|C|+\log\frac{1}{\beta})}{\varepsilon n}$ fraction of the points in the database correctly. Recall, however, that in the learning setting, we view the database D as consisting of n i.i.d. draws from some distribution over labeled examples $\mathcal{D}$. Recall the discussion of sampling bounds in Lemma 4.3. A Chernoff bound combined with a union bound tells us that with high probability, if D consists of n i.i.d. samples drawn from $\mathcal{D}$, then for all $f \in C$: $|\text{err}(f, D) - \text{err}(f, \mathcal{D})| \leq O(\sqrt{\frac{\log|C|}{n}})$. Hence, if we wish to find a hypothesis that has error within α of the optimal error on the distribution $\mathcal{D}$, it suffices to draw a database D consisting of $n \geq \log|C|/\alpha^2$ samples, and learn the best classifier f^* on D.

Now consider the problem of privately PAC learning, using the exponential mechanism as described above. Recall that, by Theorem 3.11, it is highly unlikely that the exponential mechanism will return a function f with utility score that is inferior to that of the optimal f^* by more than an additive factor of $O((\Delta u/\varepsilon)\log|C|)$, where in this case Δu, the sensitivity of the utility function, is $1/n$. That is, with high probability the exponential mechanism will return a function $f \in C$ such that:

$$\begin{aligned}\text{err}(f, D) &\leq \min_{f^* \in C} \text{err}(f^*, D) + O\left(\frac{(\log|C|)}{\varepsilon n}\right) \\ &\leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + O\left(\sqrt{\frac{\log|C|}{n}}\right) + O\left(\frac{(\log|C|)}{\varepsilon n}\right).\end{aligned}$$

Hence, if we wish to find a hypothesis that has error within α of the optimal error on the distribution $\mathcal{D}$, it suffices to draw a database D consisting of:

$$n \geq O\left(\max\left(\frac{\log|C|}{\varepsilon\alpha}, \frac{\log|C|}{\alpha^2}\right)\right),$$

which is not asymptotically any more than the database size that is required for non-private learning, whenever $\varepsilon \geq \alpha$.

A corollary of this simple calculation[1] is that (ignoring computational efficiency), a class of functions C is PAC learnable if and only if it is privately PAC learnable.

Can we say something stronger about a concept class C that is SQ learnable? Observe that if C is efficiently SQ learnable, then the learning algorithm for C need only access the data through an SQ oracle, which is very amenable to differential privacy: note that an SQ oracle answers an expectation query defined over a predicate $\phi(x, y) \in [0, 1]$, $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x, y)]$, which is only $1/n$ sensitive when estimated on a database D which is a sample of size n from $\mathcal{D}$. Moreover, the learning algorithm does not need to receive the answer exactly, but can be run with any answer a that has the property that: $|\mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x, y)]-a| \leq \tau$: that is, the algorithm can be run using *noisy answers* on *low sensitivity queries*. The benefit of this is that we can answer such queries computationally efficiently, using the Laplace mechanism — but at the expense of requiring a potentially large sample size. Recall that the Laplace mechanism can answer m $1/n$ sensitive queries with $(\varepsilon, 0)$-differential privacy and with expected worst-case error $\alpha = O(\frac{m \log m}{\varepsilon n})$. Therefore, an *SQ* learning algorithm which requires the answers to m queries with accuracy α can be run with a sample size of $n = O(\max(\frac{m \log m}{\varepsilon\alpha}, \frac{\log m}{\alpha^2}))$. Let us compare this to the sample size required for a non-private *SQ* learner. If the *SQ* learner needs to make m queries to tolerance α, then by a Chernoff bound and a union bound, a sample size of $O(\log m/\alpha^2)$ suffices. Note that for $\varepsilon = O(1)$ and error $\alpha = O(1)$, the non-private algorithm potentially requires exponentially fewer samples. However, at the error tolerance $\alpha \leq 1/m$ as allowed in the definition of SQ learning, the sample complexity for private SQ learning is no worse than the sample complexity for non-private SQ learning, for $\epsilon = \Theta(1)$.

The upshot is that *information theoretically*, privacy poses very little hinderance to machine learning. Moreover, for any algorithm that accesses the data only though an SQ oracle,[2] then the reduction to

[1] Together with corresponding lower bounds that show that for general C, it is not possible to non-privately PAC learn using a sample with $o(\log |C|/\alpha^2)$ points.

[2] And in fact, almost every class (with the lone exception of *parity functions*) of functions known to be PAC learnable is also learnable using only an SQ oracle.

private learning is immediate via the Laplace mechanism, and preserves computational efficiency!

11.2 Differentially private online learning

In this section, we consider a slightly different learning problem, known as the problem of *learning from expert advice*. This problem will appear somewhat different from the classification problems that we discussed in the previous section, but in fact, the simple algorithm presented here is extremely versatile, and can be used to perform classification among many other tasks which we will not discuss here.

Imagine that you are betting on horse races, but unfortunately know nothing about horses! Nevertheless, you have access to the opinions of some k *experts*, who every day make a prediction about which horse is going to win. Each day you can choose one of the experts whose advice you will follow, and each day, following your bet, you learn which horse actually won. How should you decide which expert to follow each day, and how should you evaluate your performance? The experts are not perfect (in fact they might not even be any good!), and so it is not reasonable to expect you to make the correct bet all of the time, or even most of the time if none of the experts do so. However, you might have a weaker goal: can you bet on horses in such a way so that you do almost as well as *the best expert, in hindsight*?

Formally, an online learning algorithm A operates in the following environment:

1. Each day $t = 1, \ldots, T$:
   (a) A chooses an expert $a_t \in \{1, \ldots, k\}$
   (b) A observes a loss $\ell_i^t \in [0, 1]$ for each expert $i \in \{1, \ldots, k\}$ and experiences loss $\ell_{a_t}^t$.

For a sequence of losses $\ell^{\leq T} \equiv \{\ell^t\}_{t=1}^T$, we write:

$$L_i(\ell^{\leq T}) = \frac{1}{T} \sum_{t=1}^{T} \ell_i^t$$

to denote the total average loss of expert i over all T rounds, and write

$$L_A(\ell^{\leq T}) = \frac{1}{T}\sum_{t=1}^{T} \ell_{a_t}^t$$

to denote the total average loss of the algorithm.

The *regret* of the algorithm is defined to be the difference between the loss that it actually incurred, and the loss of the *best* expert in hindsight:

$$\text{Regret}(A, \ell^{\leq T}) = L_A(\ell^{\leq T}) - \min_i L_i(\ell^{\leq T}).$$

The goal in online learning is to design algorithms that have the guarantee that for *all possible loss sequences* $\ell^{\leq T}$, even adversarialy chosen, the regret is guaranteed to tend to zero as $T \to \infty$. In fact, this is possible using the multiplicative weights algorithm (known also by many names, e.g., the Randomized Weighted Majority Algorithm, Hedge, Exponentiated Gradient Descent, and multiplicative weights being among the most popular).

Remark 11.1. We have already seen this algorithm before in Section 4 — this is just the multiplicative weights update rule in another guise! In fact, it would have been possible to derive all of the results about the private multiplicative weights mechanism directly from the regret bound we state in Theorem 11.1.

Algorithm 15 The Multiplicative Weights (or Randomized Weighted Majority (RWM)) algorithm, version 1. It takes as input a stream of losses $\ell^1, \ell^2, \ldots$ and outputs a stream of actions $a_1, a_2, \ldots$. It is parameterized by an update parameter η.

RWM(η):

For each $i \in \{1, \ldots, k\}$, let $w_i \leftarrow 1$.
for $t = 1, \ldots$ **do**
  Choose action $a_t = i$ with probability proportional to w_i
  Observe ℓ^t and set $w_i \leftarrow w_i \cdot \exp(-\eta \ell_i^t)$, for each $i \in [k]$
end for

It turns out that this simple algorithm already has a remarkable regret bound.

Theorem 11.1. For any adversarially chosen sequence of losses of length T, $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$ the Randomized Weighted Majority algorithm with update parameter η has the guarantee that:

$$\mathbb{E}[\text{Regret}(\text{RWM}(\eta), \ell^{\leq T})] \leq \eta + \frac{\ln(k)}{\eta T}, \tag{11.1}$$

where k is the number of experts. Choosing $\eta = \sqrt{\frac{\ln k}{T}}$ gives:

$$\mathbb{E}[\text{Regret}(\text{RWM}(\eta), \ell^{\leq T})] \leq 2\sqrt{\frac{\ln k}{T}}.$$

This remarkable theorem states that even faced with an adversarial sequence of losses, the Randomized Weighted Majority algorithm can do as well, on average, as the best expert among k in hindsight, minus only an additional additive term that goes to zero at a rate of $O(\sqrt{\frac{\ln k}{T}})$. In other words, after at most $T \leq 4\frac{\ln k}{\alpha^2}$ rounds, the regret of the randomized weighted majority algorithm is guaranteed to be at most α! Moreover, this bound is the best possible.

Can we achieve something similar, but under the constraint of differential privacy? Before we can ask this question, we must decide *what is the input database*, and at what granularity we would like to protect privacy? Since the input is the collection of loss vectors $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$, it is natural to view $\ell^{\leq T}$ as the database, and to view a neighboring database $\hat{\ell^{\leq T}}$ as one that differs in the entire loss vector in any single timestep: i.e., one in which for some fixed timestep t, $\hat{\ell}^i = \ell^i$ for all $i \neq t$, but in which ℓ^t and $\hat{\ell}^t$ can differ arbitrarily. The output of the algorithm is the sequence of actions that it chooses, $a_1, \ldots, a_T$, and it is this that we wish to be output in a differentially private manner.

Our first observation is that the randomized weighted majority algorithm chooses an action at each day t in a familiar manner! We here rephrase the algorithm in an equivalent way:

It chooses an action a_t with probability proportional to: $\exp(-\eta \sum_{j=1}^{t-1} \ell_i^j)$, which is simply the exponential mechanism with quality score $q(i, \ell^{<T}) = \sum_{j=1}^{t-1} \ell_i^j$, and privacy parameter $\varepsilon = 2\eta$. Note that because each $\ell_i^t \in [0, 1]$, the quality function has sensitivity 1. Thus,

Algorithm 16 The Multiplicative Weights (or Randomized Weighted Majority (RWM)) algorithm, rephrased. It takes as input a stream of losses $\ell^1, \ell^2, \ldots$ and outputs a stream of actions $a_1, a_2, \ldots$. It is parameterized by an update parameter η.

RWM(η):

for $t = 1, \ldots$ **do**

Choose action $a_t = i$ with probability proportional to $\exp(-\eta \sum_{j=1}^{t-1} \ell_i^j)$

Observe ℓ^t

end for

each round t, the randomized weighted majority algorithm chooses an action a_t in a way that preserves 2η differential privacy, so to achieve privacy ε it suffices to set $\eta = \varepsilon/2$.

Moreover, over the course of the run of the algorithm, it will choose an action T times. If we want the *entire* run of the algorithm to be (ε, δ)-differentially private for some ε and δ, we can thus simply apply our composition theorems. Recall that by Theorem 3.20, since there are T steps in total, if each step of the algorithm is $(\varepsilon', 0)$-differentially private for $\varepsilon' = \varepsilon/\sqrt{8T\ln(1/\delta)}$, then the entire algorithm will be (ε, δ) differentially private. Thus, the following theorem is immediate by setting $\eta = \varepsilon'/2$:

Theorem 11.2. For a sequence of losses of length T, the algorithm RWM(η) with $\eta = \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$ is (ε, δ)-differentially private.

Remarkably, we get this theorem *without modifying the original randomized weighted majority algorithm at all*, but rather just by setting η appropriately. In some sense, we are getting privacy for free! We can therefore use Theorem 11.1, the utility theorem for the RWM algorithm, without modification as well:

Theorem 11.3. For any adversarially chosen sequence of losses of length T, $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$ the Randomized Weighted Majority

algorithm with update parameter $\eta = \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$ has the guarantee that:

$$\begin{aligned}\mathbb{E}[\text{Regret}(\text{RWM}(\eta), \ell^{\leq T})] &\leq \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}} + \frac{\sqrt{32\ln(1/\delta)}\ln k}{\varepsilon\sqrt{T}} \\ &\leq \frac{\sqrt{128\ln(1/\delta)}\ln k}{\varepsilon\sqrt{T}},\end{aligned}$$

where k is the number of experts.

Since the per-round loss at each time step t is an independently chosen random variable (over the choices of a_t) with values bounded in $[-1, 1]$, we can also apply a Chernoff bound to get a high probability guarantee:

Theorem 11.4. For any adversarially chosen sequence of losses of length T, $\ell^{\leq T} = (\ell^1, \ldots, \ell^T)$ the Randomized Weighted Majority algorithm with update parameter $\eta = \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$ produces a sequence of actions such that with probability at least $1 - \beta$:

$$\begin{aligned}\text{Regret}(\text{RWM}(\eta), \ell^{\leq T}) &\leq \frac{\sqrt{128\ln(1/\delta)}\ln k}{\varepsilon\sqrt{T}} + \sqrt{\frac{\ln k/\beta}{T}} \\ &= O\left(\frac{\sqrt{\ln(1/\delta)}\ln(k/\beta)}{\varepsilon\sqrt{T}}\right).\end{aligned}$$

This bound is nearly as good as the best possible bound achievable even without privacy (i.e., the RWM bound) — the regret bound is larger only by a factor of $\Omega(\frac{\sqrt{\ln(k)\ln(1/\delta)}}{\varepsilon})$. (We note that by using a different algorithm with a more careful analysis, we can remove this extra factor of $\sqrt{\ln k}$). Since we are in fact using the same algorithm, efficiency is of course preserved as well. Here we have a powerful example in machine learning where privacy is nearly "free." Notably, just as with the non-private algorithm, our utility bound only gets better the longer we run the algorithm, while keeping the privacy guarantee the same.[3]

[3]Of course, we have to set the update parameter appropriately, just as we have to do with the non-private algorithm. This is easy when the number of rounds T is known ahead of time, but can also be done adaptively when the number of rounds is not known ahead of time.

11.3 Empirical risk minimization

In this section, we apply the randomized weighted majority algorithm discussed in the previous section to a special case of the problem of empirical risk minimization to learn a linear function. Rather than assuming an adversarial model, we will assume that *examples* are drawn from some known distribution, and we wish to learn a classifier from some finite number of samples from this distribution so that our loss will be low on *new* samples drawn from the same distribution.

Suppose that we have a distribution $\mathcal{D}$ over *examples* $x \in [-1,1]^d$, and for each such vector $x \in [-1,1]^d$, and for each vector $\theta \in [0,1]^d$ with $\|\theta\|_1 = 1$, we define the loss of θ on example x to be $\text{Loss}(\theta, x) = \langle \theta, x \rangle$. We wish to find a vector θ^* to minimize the *expected* loss over examples drawn from $\mathcal{D}$:

$$\theta^* = \arg \min_{\theta \in [0,1]^d : \|\theta\|_1 = 1} \mathbb{E}_{x \sim \mathcal{D}}[\langle \theta, x \rangle].$$

This problem can be used to model the task of finding a low error linear classifier. Typically our only access to the distribution $\mathcal{D}$ is through some collection of examples $S \subset [-1,1]^d$ drawn i.i.d. from $\mathcal{D}$, which serves as the input to our learning algorithm. We will here think of this sample S as our private database, and will be interested in how well we can privately approximate the error of θ^* as a function of $|S|$ (the *sample complexity* of the learning algorithm).

Our approach will be to reduce the problem to that of learning with expert advice, and apply the private version of the randomized weighted majority algorithm as discussed in the last section:

1. The *experts* will be the d standard basis vectors $\{e_1, \ldots, e_d\}$, where $e_i = (0, \ldots, 0, \underbrace{1}_{i}, 0, \ldots, 0)$.
2. Given an example $x \in [-1,1]^d$, we define a loss vector $\ell(x) \in [-1,1]^d$ by setting $\ell(x)_i = \langle e_i, x \rangle$ for each $i \in \{1, \ldots, d\}$. In other words, we simply set $\ell(x)_i = x_i$.
3. At time t, we choose a loss function ℓ^t by sampling $x \sim \mathcal{D}$ and setting $\ell^t = \ell(x)$.

Note that if we have a sample S from $\mathcal{D}$ of size $|S| = T$, then we can run the RWM algorithm on the sequence of losses as described above for a total of T rounds. This will produce a sequence of outputs $a_1, \ldots, a_T$, and we will define our final classifier to be $\theta^T \equiv \frac{1}{T}\sum_{i=1}^{T} a_i$. (Recall that each a_i is a standard basis vector $a_i \in \{e_1, \ldots, e_d\}$, and so we have $\|\theta^T\|_1 = 1$).

We summarize the algorithm below:

Algorithm 17 An algorithm for learning linear functions. It takes as input a private database of examples $S \subset [-1,1]^d$, $S = (x_1, \ldots, x_T)$, and privacy parameters ε and δ.

LinearLearner(S, ε, δ):

Let $\eta \leftarrow \frac{\varepsilon}{\sqrt{32T\ln(1/\delta)}}$

for $t = 1$ to $T = |S|$ **do**

Choose vector $a_t = e_i$ with probability proportional to $\exp(-\eta\sum_{j=1}^{t-1}\ell_i^j)$

Let loss vector $\ell^t = (\langle e_1, x_t\rangle, \langle e_2, x_t\rangle, \ldots, \langle e_d, x_t\rangle)$.

end for

Output $\theta^T = \frac{1}{T}\sum_{t=1}^{T} a_t$.

We have already seen that LinearLearner is private, since it is simply an instantiation of the randomized weighted majority algorithm with the correct update parameter η:

Theorem 11.5. **LinearLearner**(S, ε, δ) is (ε, δ)-differentially private.

It remains to analyze the classification accuracy of LinearLearner, which amounts to considering the regret bound of the private RWM algorithm.

Theorem 11.6. If S consists of T i.i.d. samples $x \sim \mathcal{D}$, then with probability at least $1-\beta$, LinearLearner outputs a vector θ^T such that:

$$\mathbb{E}_{x\sim\mathcal{D}}[\langle\theta^T, x\rangle] \leq \min_{\theta^*} \mathbb{E}_{x\sim\mathcal{D}}[\langle\theta^*, x\rangle] + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right),$$

where d is the number of experts.

Proof. By Theorem 11.4, we have the following guarantee with probability at least $1 - \beta/2$:

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\langle a_t, x_t\rangle &\leq \min_{i\in\{1,\dots,d\}}\left\langle e_i, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right)\\ &= \min_{\theta^*\in[0,1]^d:\|\theta^*\|_1=1}\left\langle \theta^*, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right).\end{aligned}$$

In the first equality, we use the fact that the minimum of a linear function over the simplex is achieved at a vertex of the simplex. Noting that each $x_t \sim \mathcal{D}$ independently and that each $\langle x_t, e_i\rangle$ is bounded in $[-1, 1]$, we can apply Azuma's inequality twice to bound the two quantities with probability at least $1 - \beta/2$:

$$\begin{aligned}&\left|\frac{1}{T}\sum_{t=1}^{T}\langle a_t, x_t\rangle - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{x\sim\mathcal{D}}\langle a_t, x\rangle\right|\\ &\quad= \left|\frac{1}{T}\sum_{t=1}^{T}\langle a_t, x_t\rangle - \mathbb{E}_{x\sim\mathcal{D}}\langle \theta^T, x\rangle\right| \leq O\left(\sqrt{\frac{\ln(1/\beta)}{T}}\right)\end{aligned}$$

and

$$\max_{i\in\{1,\dots,d\}}\left|\left\langle e_i, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle - \mathbb{E}_{x\sim\mathcal{D}}\langle e_i, x\rangle\right| \leq O\left(\sqrt{\frac{\ln(d/\beta)}{T}}\right).$$

Hence we also have:

$$\max_{\theta^*\in[0,1]^d:\|\theta^*\|_1=1}\left|\left\langle \theta^*, \frac{1}{T}\sum_{t=1}^{T} x_t\right\rangle - \mathbb{E}_{x\sim\mathcal{D}}\langle \theta^*, x\rangle\right| \leq O\left(\sqrt{\frac{\ln d/\beta}{T}}\right).$$

Combining these inequalities gives us our final result about the output of the algorithm θ^T:

$$\mathbb{E}_{x\sim\mathcal{D}}\langle \theta^T, x\rangle \leq \min_{\theta^*\in[0,1]^d:\|\theta^*\|_1=1}\mathbb{E}_{x\sim\mathcal{D}}\langle \theta^*, x\rangle + O\left(\frac{\sqrt{\ln(1/\delta)}\ln(d/\beta)}{\varepsilon\sqrt{T}}\right).$$

□

11.4 Bibliographical notes

The PAC model of machine learning was introduced by Valiant in 1984 [83], and the SQ model was introduced by Kearns [53]. The randomized weighted majority algorithm is originally due to Littlestone and Warmuth [57], and has been studied in many forms. See Blum and Mansour [9] or Arora et al. [1] for a survey. The regret bound that we use for the randomized weighted majority algorithm is given in [1].

Machine learning was one of the first topics studied in differential privacy, beginning with the work of Blum et al. [7], who showed that algorithms that operate in the SQ-learning framework could be converted into privacy preserving algorithms. The sample complexity of differentially private learning was first considered by Kasiviswanathan, Lee, Nissim, Raskhodnikova, and Smith, "What can we Learn Privately?" [52], which characterize the sample complexity of private learning up to polynomial factors. For more refined analysis of the sample complexity of private learning, see [3, 4, 12, 19].

There is also extensive work on efficient machine learning algorithms, including the well known frameworks of SVMs and empirical risk minimizers [13, 55, 76]. Spectral learning techniques, including PCA and low rank matrix approximation have also been studied [7, 14, 33, 42, 43, 51].

Private learning from expert advice was first considered by Dwork et al. [26]. The fact that the randomized weighted majority algorithm is privacy preserving without modification (when the update parameter is set appropriately) is folklore (following from advanced composition [32]) and has been widely used; for example, in [48]. For a more general study of private online learning, see [50], and for a more general study of empirical risk minimization, see [50, 13].

12

Additional Models

So far, we have made some implicit assumptions about the model of private data analysis. For example, we have assumed that there is some trusted curator who has direct access to the private dataset, and we have assumed that the adversary only has access to the output of the algorithm, not to any of its internal state during its execution. But what if this is not the case? What if we trust no one to look at our data, even to perform the privacy preserving data analysis? What if some hacker might gain access to the internal state of the private algorithm while it is running? In this section, we relax some of our previously held assumptions and consider these questions.

In this section we describe some additional computational models that have received attention in the literature.

- The *local model* is a generalization of randomized response (see Section 2), and is motivated by situations in which individuals do not trust the curator with their data. While this lack of trust can be addressed using secure multiparty computation to simulate the role played by the trusted curator, there are also some techniques that do not require cryptography.

The next two models consider streams of *events*, each of which may be associated with an individual. For example, an event may be a search by a particular person on an arbitrary term. In a given event stream, the (potentially many) events associated with a given individual can be arbitrarily interleaved with events associated with other individuals.

- In *pan-privacy* the curator is trusted, but may be subject to compulsory non-private data release, for example, because of a subpoena, or because the entity holding the information is purchased by another, possibly less trustworthy, entity. Thus, in pan-privacy the *internal state* of the algorithm is also differentially private, as is the joint distribution of the internal state and the outputs.
- The *continual observation* model addresses the question of maintaining privacy when the goal is to continually monitor and report statistics about events, such as purchases of over-the-counter medications that might be indicative of an impending epidemic. Some work addresses pan-privacy under continual observation.

12.1 The local model

So far, we have considered a *centralized* model of data privacy, in which there exists a database administrator who has direct access to the private data. What if there is instead no trusted database administrator? Even if there is a suitable trusted party, there are many reasons not to want private data aggregated by some third party. The very existence of an aggregate database of private information raises the possibility that at some *future time*, it will come into the hands of an untrusted party, either maliciously (via data theft), or as a natural result of organizational succession. A superior model — from the perspective of the owners of private data — would be a local model, in which agents could (randomly) answer questions in a differentially private manner about their own data, without ever sharing it with anyone else. In the context of predicate queries, this seems to severely limit the expressivity of a private mechanism's interaction with the data: The mechanism can ask each user whether or not her data satisfies a given predicate, and

the user may flip a coin, answering truthfully only with slightly higher probability than answering falsely. In this model what is possible?

The local privacy model was first introduced in the context of learning. The local privacy model formalizes randomized response: there is no central database of private data. Instead, each individual maintains possession of their own data element (a database of size 1), and answers questions about it only in a differentially private manner. Formally, the database $x \in \mathbb{N}^{|\mathcal{X}|}$ is a collection of n elements from some domain $\mathcal{X}$, and each $x_i \in x$ is held by an individual.

Definition 12.1 (Local Randomizer)**.** An ε-local randomizer $R : \mathcal{X} \to W$ is an ε-differentially private algorithm that takes as input a database of size $n = 1$.

In the local privacy model, algorithms may interact with the database only through a local randomizer oracle:

Definition 12.2 (LR Oracle)**.** An LR oracle $LR_D(\cdot, \cdot)$ takes as input an index $i \in [n]$ and an ε-local randomizer R and outputs a random value $w \in W$ chosen according to the distribution $R(x_i)$, where $x_i \in D$ is the element held by the ith individual in the database.

Definition 12.3 ((Local Algorithm))**.** An algorithm is ε-local if it accesses the database D via the oracle LR_D, with the following restriction: If $LR_D(i, R_1), \ldots, LR_D(i, R_k)$ are the algorithm's invocations of LR_D on index i, where each R_J is an ε_j-local randomizer, then $\varepsilon_1 + \cdots + \varepsilon_k \leq \varepsilon$.

Because differential privacy is composable, it is easy to see that ε-local algorithms are ε-differentially private.

Observation 12.1. ε-local algorithms are ε-differentially private.

That is to say, an ε-local algorithm interacts with the data using only a sequence of ε-differentially private algorithms, each of which computes only on a database of size 1. Because nobody other than its owner ever touches any piece of private data, the local setting is far more secure: it does not require a trusted party, and there is no central party who might be subject to hacking. Because even the algorithm

never sees private data, the internal state of the algorithm is always differentially private as well (i.e., local privacy implies pan privacy, described in the next section). A natural question is how restrictive the local privacy model is. In this section, we merely informally discuss results. The interested reader can follow the bibliographic references at the end of this section for more information. We note that an alternative name for the local privacy model is the *fully distributed* model.

We recall the definition of the statistical query (SQ) model, introduced in Section 11. Roughly speaking, given a database x of size n, the statistical query model allows an algorithm to access this database by making a polynomial (in n) number of noisy linear queries to the database, where the error in the query answers is some inverse polynomial in n. Formally:

Definition 12.4. A *statistical query* is some function $\phi : \mathcal{X} \times \{0,1\} \to [0,1]$. A *statistical query oracle* for a distribution over labeled examples $\mathcal{D}$ with tolerance τ is an oracle $\mathcal{O}^\tau_{\mathcal{D}}$ such that for every statistical query ϕ:

$$\left|\mathcal{O}^\tau_{\mathcal{D}}(\phi) - \mathbb{E}_{(x,y)\sim\mathcal{D}}[\phi(x,y)]\right| \leq \tau$$

In other words, an SQ oracle takes as input a statistical query ϕ, and outputs some value that is guaranteed to be within $\pm\tau$ of the expected value of ϕ on examples drawn from $\mathcal{D}$.

Definition 12.5. An algorithm A is said to SQ-learn a class of functions C if for every $\alpha, \beta > 0$ there exists an $m = \text{poly}(d, 1/\alpha, \log(1/\beta))$ such that A makes at most m queries of tolerance $\tau = 1/m$ to $\mathcal{O}^\tau_{\mathcal{D}}$, and with probability $1 - \beta$, outputs a hypothesis $f \in C$ such that:

$$\text{err}(f, \mathcal{D}) \leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + \alpha$$

More generally, we can talk about an algorithm (for performing any computation) as operating in the SQ model if it accesses the data only through an SQ oracle:

Definition 12.6. An algorithm A is said to operate in the SQ model if there exists an m such that A makes at most m queries of tolerance $\tau = 1/m$ to $\mathcal{O}^\tau_D$, and does not have any other access to the database. A is efficient if m is polynomial in the size of the database, D.

It turns out that up to polynomial factors in the size of the database and in the number of queries, any algorithm that can be implemented in the SQ model can be implemented and analyzed for privacy in the local privacy model, and vice versa. We note that there is a distinction between an algorithm being implemented in the SQ model, and its privacy analysis being carried out in the local model: almost all of the algorithms that we have presented in the end access the data using noisy linear queries, and so can be thought of as acting in the SQ model. However, their privacy guarantees are analyzed in the centralized model of data privacy (i.e., because of some "global" part of the analysis, as in the sparse vector algorithm).

In the following summary, we will also recall the definition of PAC learning, also introduced in Section 11:

Definition 12.7. An algorithm A is said to PAC-learn a class of functions C if for every $\alpha, \beta > 0$, there exists an $m = \text{poly}(d, 1/\alpha, \log(1/\beta))$ such that for every distribution $\mathcal{D}$ over labeled examples, A takes as input m labeled examples drawn from $\mathcal{D}$ and outputs a hypothesis $f \in C$ such that with probability $1 - \beta$:

$$\text{err}(f, \mathcal{D}) \leq \min_{f^* \in C} \text{err}(f^*, \mathcal{D}) + \alpha$$

If $\min_{f^* \in C} \text{err}(f^*, \mathcal{D}) = 0$, the learner is said to operate in the *realizable* setting (i.e., there exists some function in the class which perfectly labels the data). Otherwise, the learner is said to operate in the *agnostic* setting. If A also has run time that is polynomial in $d, 1/\alpha$, and $\log(1/\beta)$, then the learner is said to be *efficient*. If there is an algorithm which PAC-learns C, then C is said to be PAC-learnable. Note that the main distinction between an SQ learning algorithm and a PAC learning algorithm, is that the PAC learning algorithm gets direct access to the database of examples, whereas the SQ learning algorithm only has access to the data through a noisy SQ oracle.

What follows is some of our understanding of the limitations of the SQ model and problems which separate it from the centralized model of data privacy.

1. A single sensitivity-1 query can be answered to error $O(1)$ in the centralized model of data privacy using the Laplace mechanism, but requires error $\Theta(\sqrt{n})$ in the local data privacy model.
2. The set of function classes that we can (properly) learn in the local privacy model is exactly the set of function classes that we can properly learn in the SQ model (up to polynomial factors in the database size and query complexity of the algorithm). In contrast, the set of things we can (properly or agnostically) learn in the centralized model corresponds to the set of things we can learn in the PAC model. SQ learning is strictly weaker, but this is not a huge handicap, since parity functions are essentially the only interesting class that is PAC learnable but not SQ learnable. We remark that we refer explicitly to proper learning here (meaning the setting in which there is some function in the class which perfectly labels the data). In the PAC model there is no information theoretic difference between proper and agnostic learning, but in the SQ model the difference is large: see the next point.
3. The set of queries that we can release in the local privacy model are exactly those queries that we can agnostically learn in the SQ model. In contrast, the set of things we can release in the centralized model corresponds to the set of things we can agnostically learn in the PAC model. This is a much bigger handicap — even conjunctions (i.e., marginals) are not agnostically learnable in the SQ model. This follows from the information theoretic reduction from agnostic learning (i.e., *distinguishing*) to query release that we saw in Section 5 using the iterative construction mechanism.

We note that if we are only concerned about computationally bounded adversaries, then in principle distributed agents can use *secure multiparty computation* to simulate private algorithms in the centralized setting. While this does not actually give a differential privacy guarantee, the result of such simulations will be indistinguishable from the result of differentially private computations, from the point of view of a computationally bounded adversary. However, general secure multiparty computation protocols typically require huge amounts of message passing (and hence sometimes have unreasonably large run times),

whereas algorithms in the local privacy model tend to be extremely simple.

12.2 Pan-private streaming model

The goal of a pan-private algorithm is to remain differentially private even against an adversary that can, on rare occasions, observe the algorithm's internal state. Intrusions can occur for many reasons, including hacking, subpoena, or *mission creep*, when data collected for one purpose are used for a different purpose ("Think of the children!"). Pan-private streaming algorithms provide protection against all of these. Note that ordinary streaming algorithms do *not* necessarily provide privacy against intrusions, as even a low-memory streaming algorithm can hold a small number of data items in memory, which would be completely exposed in an intrusion. On the technical side, intrusions can be *known* to the curator (subpoena) or unknown (hacking). These can have very different effects, as a curator aware of an intrusion can take protective measures, such as re-randomizing certain variables.

12.2.1 Definitions

We assume a data stream of unbounded length composed of elements in a universe $\mathcal{X}$. It may be helpful to keep in mind as motivation data analysis on a query stream, in which queries are accompanied by the IP address of the issuer. For now, we ignore the query text itself; the universe $\mathcal{X}$ is the universe of potential IP addresses. Thus, intuitively, *user-level* privacy protects the presence or absence of an IP address in the stream, indpendent of the number of times it arises, should it actually be present at all. In contrast, *event-level* privacy merely protects the privacy of individual accesses. For now, we focus on user-level privacy.

As usual in differentially private algorithms, the adversary can have arbitrary control of the input stream, and may have arbitrary auxiliary knowledge obtained from other sources. It can also have arbitrary computational power.

We assume the algorithm runs until it receives a special signal, at which point it produces (observable) outputs. The algorithm may optionally continue to run and produce additional outputs later, again in response to a special signal. Since outputs are obvservable we do not provide privacy for the special signals.

A streaming algorithm experiences a sequence of internal states. and produces a (possibly unbounded) sequence of outputs. Let I denote the set of possible internal states of the algorithm, and σ the set of possible output sequences. We assume that the adversary can only observe internal states and the output sequence; it cannot see the data in the stream (although it may have auxiliary knowledge about some of these data) and it has no access to the *length* of the input sequence.

Definition 12.8 ($\mathcal{X}$-Adjacent Data Streams)**.** We think of data streams as being of unbounded length; *prefixes* have finite length. Data streams S and S' are $\mathcal{X}$-adjacent if they differ only in the presence or absence of *all* occurrences of a single element $u \in \mathcal{X}$. We define $\mathcal{X}$-adjacency for stream prefixes analogously.

User-Level Pan-Privacy. An algorithm **Alg** mapping data stream prefixes to the range $\mathrm{I} \times \sigma$, is *pan-private against a single intrusion* if for all sets $\mathrm{I}' \subseteq \mathrm{I}$ of internal states and $\sigma' \subseteq \sigma$ of output sequences, and for all pairs of adjacent data stream prefixes S, S'

$$\Pr[\mathbf{Alg}(S) \in (\mathrm{I}', \sigma')] \leq e^{\varepsilon} \Pr[\mathbf{Alg}(S') \in (\mathrm{I}', \sigma')],$$

where the probability spaces are over the coin flips of the algorithm **Alg**.

This definition speaks only of a single intrusion. For multiple intrusions we must consider interleavings of observations of internal states and outputs.

The relaxation to *event-level privacy* is obtained by modifying the notion of adjacency so that, roughly speaking, two streams are event-adjacent if they differ in a single instance of a single element in $\mathcal{X}$; that is, one instance of one element is deleted/added. Clearly, event-level privacy is a much weaker guarantee than user-level privacy.

Remark 12.1. If we assume the existence of a very small amount of secret storage, not visible to the adversary, then many problems for which we have been unable to obtain pan-private solutions have (non-pan-) private streaming solutions. However, the *amount* of secret storage is not so important as its *existence*, since secret storage is vulnerable to the social pressures against which pan-privacy seeks to protect the data (and the curator).

Pan-Private Density Estimation. Quite surprisingly, pan-privacy can be achieved even for *user-level* privacy of many common streaming computations. As an example, consider the problem of *density estimation*: given a universe $\mathcal{X}$ of data elements and a stream σ, the goal is to estimate the fraction of $\mathcal{X}$ that acutally appears in the stream. For example, the universe consists of all teenagers in a given community (represented by IP addresses), and the goal is to understand what fraction visit the Planned Parenthood website.

Standard low-memory streaming solutions for density estimation involve recording the results of deterministic computations of at least some input items, an approach that is inherently not pan-private. Here is a simple, albeit high-memory, solution inspired by randomized response. The algorithm maintains a bit b_a for each IP address a (which may appear any number of times in the stream), initialized uniformly at random. The stream is processed one element at a time. On input a the algorithm flips a bit biased to 1; that is, the biased bit will take value 0 with probability $1/2 - \varepsilon$, and value 1 with probability $1/2 + \varepsilon$. The algorithm follows this procedure independent of the number of times IP address a appears in the data stream. This algorithm is $(\varepsilon, 0)$-differentially private. As with randomized response, we can estimate the fraction of "real" 1's by $z = 2(y - |\mathcal{X}|/2)/|\mathcal{X}|$, where y is the actual number of 1's in the table after the stream is processed. To ensure pan-privacy, the algorithm publishes a noisy version of z. As with randomized response, the error will be on the order of $1/\sqrt{|\mathcal{X}|}$, yielding meaningful results when the density is high.

Other problems enjoying user-level pan-private algorithms include:

- Estimating, for any t, the fraction of elements appearing exactly t times;

- Estimating the *t-cropped mean*: roughly, the average, over all elements, of the minimum of t and the number of occurrences of the element in the data stream;
- Estimating the fraction of k-heavy hitters (elements of $\mathcal{X}$ that appear at least k times in the data stream).

Variants of these problems can also be defined for *fully dynamic* data, in which counts can be decremented as well as incremented. For example, density estimation (what fraction appeared in the stream?) becomes "How many (or what fraction) of elements have a (net) count equal to zero?" These, too, can be solved with user-level pan-privacy, using differentially private variations of *sketching* techniques from the streaming literature.

12.3 Continual observation

Many applications of data analysis involve repeated computations, either because the entire goal is one of monitoring of, for example, traffic conditions, search trends, or incidence of influenza. In such applications the system is required to continually produce outputs. We therefore need techniques for achieving *differential privacy under continual observation*.

As usual, differential privacy will require having essentially the same distribution on outputs for each pair of adjacent databases, but how should we define adjacency in this setting? Let us consider two example scenarios.

Suppose the goal is to monitor public health by analyzing statistics from an H1N1 self-assessment Web site.[1] Individuals can interact with the site to learn whether symptoms they are experiencing may be indicative of the H1N1 flu. The user fills in some demographic data (age, zipcode, sex), and responds to queries about his symptoms (fever over 100.4°F?, sore throat?, duration of symptoms?). We would expect a given individual to interact very few times with the H1N1 self-assessment site (say, if we restrict our attention to a six-month

[1] https://h1n1.cloudapp.net provided such a service during the winter of 2010; user-supplied data were stored for analysis with the user's consent.

period). For simplicity, let us say this is just once. In such a setting, it is sufficient to ensure *event-level* privacy, in which the privacy goal is to hide the presence or absence of a single event (interaction of one user with the self-assessment site).

Suppose again that the goal is to monitor public health, this time by analyzing search terms submitted to a medical search engine. Here it may no longer be safe to assume an individual has few interactions with the Web site, even if we restrict attention to a relatively short period of time. In this case we would want *user-level* privacy, ensuring that the entire set of a user's search terms is protected simultaneously.

We think of continual observation algorithms as taking steps at discrete time intervals; at each step the algorithm receives an input, computes, and produces output. We model the data as arriving in a stream, at most one data element in each time interval. To capture the fact that, in real life, there are periods of time in which nothing happens, null events are modeled by a special symbol in the data stream. Thus, the intuitive notion of "t time periods" corresponds to processing a sequence of t elements in the stream.

For example, the motivation behind the counter primitive below is to count the number of times that something has occurred since the algorithm was started (the counter is very general; we don't specify *a priori* what it is counting). This is modeled by an input stream over $\{0, 1\}$. Here, "0" means "nothing happened," "1" means the event of interest occurred, and for $t = 1, 2, \ldots, T$ the algorithm outputs an approximation to the number of 1s seen in the length t prefix of the stream.

There are three natural options:

1. Use randomized response for each time period and add this randomized value to the counter;
2. Add noise distributed according to $\mathrm{Lap}(1/\varepsilon)$ to the true value for each time step and add this perturbed value to the counter;
3. Compute the true count at each time step, add noise distributed according to $\mathrm{Lap}(T/\varepsilon)$ to the count, and release this noisy count.

All of these options result in noise on the order of at least $\Omega(\sqrt{T}/\varepsilon)$. The hope is to do much better by exploiting structure of the query set.

Let $\mathcal{X}$ be the universe of possible input symbols. Let S and S' be stream prefixes (i.e., finite streams) of symbols drawn from $\mathcal{X}$. Then $\mathrm{Adj}(S, S')$ ("S is adjacent to S'") if and only if there exist $a, b \in \mathcal{X}$ so that if we change some of the instances of a in S to instances of b, then we get S'. More formally, $\mathrm{Adj}(S, S')$ iff $\exists a, b \in \mathcal{X}$ and $\exists R \subseteq [|S|]$, such that $S|_{R:a\to b} = S'$. Here, R is a set of indices in the stream prefix S, and $S|_{R:a\to b}$ is the result of replacing all the occurrences of a at these indices with b. Note that adjacent prefixes are always of the same length.

To capture event-level privacy, we restrict the definition of adjacency to the case $|R| \leq 1$. To capture user-level privacy we do not constrain the size of R in the definition of adjacency.

As noted above, one option is to publish a noisy count at each time step; the count published at time t reflects the approximate number of 1s in the length t prefix of the stream. The privacy challenge is that early items in the stream are subject to nearly T statistics, so for $(\varepsilon, 0)$-differential privacy we would be adding noise scaled to T/ε, which is unacceptable. In addition, since the 1s are the "interesting" elements of the stream, we would like that the distortion be scaled to the number of 1s seen in the stream, rather than to the length of the stream. This rules out applying randomized response to each item in the stream independently.

The algorithm below follows a classical approach for converting static algorithms to dynamic algorithms.

Assume T is a power of 2. The intervals are the natural ones corresponding to the labels on a complete binary tree with T leaves, where the leaves are labeled, from left to right, with the intervals $[0, 0], [1, 1], \ldots, [T-1, T-1]$ and each parent is labeled with the interval that is the union of the intervals labeling its children. The idea is to compute and release a noisy count for each label $[s, t]$; that is, the released value corresponding to the label $[s, t]$ is a noisy count of the number of 1s in positions $s, s+1, \ldots, t$ of the input stream. To learn the approximate cumulative count at time $t \in [0, T-1]$ the analyst uses the binary representation of t to determine a set of at most $\log_2 T$

Counter (T, ε)

Initialization. Initialize $\xi = \log_2 T/\varepsilon$, and sample Counter $\sim \mathrm{Lap}(\xi)$.

Intervals. For $i \in \{1, \ldots, \log T\}$, associate with each string $s \in \{0,1\}^i$ the time interval S of $2^{\log T - i}$ time periods $\{s \circ 0^{\log T - i}, \ldots s \circ 1^{\log T - i}\}$. The interval *begins in time* $s \circ 0^{\log T - i}$ and *ends in time* $s \circ 1^{\log T - i}$.

Processing. In time period $t \in \{0, 1, \ldots, T-1\}$, let $x_t \in \{0,1\}$ be the t-th input bit:

1. For every interval I beginning at time t, initialize c_I to an independent random draw: $c_I \leftarrow \mathrm{Lap}((\log_2 T)/\varepsilon)$;
2. For every interval I containing t, add x_t to c_I: $c_I \leftarrow c_I + x_t$;
3. For every interval I that ends in time t, output c_I.

Figure 12.1: Event-level private counter algorithm (not pan-private).

disjoint intervals whose union is $[0, t]$, and computes the sum of the corresponding released noisy counts.[2] See Figure 12.1.

Each stream position $t \in [0, T-1]$ appears in at most $1 + \log_2 T$ intervals (because the height of the tree is $\log_2 T$), and so each element in the stream affects at most $1 + \log_2 T$ released noisy counts. Thus, adding noise to each interval count distributed according to $\mathrm{Lap}((1 + \log_2 T)/\varepsilon)$ ensures $(\varepsilon, 0)$-differential privacy. As for accuracy, since the binary representation of any index $t \in [0, T-1]$ yields a disjoint set of at most $\log_2 T$ intervals whose union is $[0, t]$ we can apply Lemma 12.2 below to conclude that the expected error is tightly concentrated around $(\log_2 T)^{3/2}$. The maximum expected error, over all times t, is on the order of $(\log_2 T)^{5/3}$.

Lemma 12.2. Let Let $Y_1, \ldots, Y_k$ be independent variables with distribution $\mathrm{Lap}(b_i)$. Let $Y = \sum_i Y_i$ and $b_{\max} = \max_i b_i$. Let $\nu \geq \sqrt{\sum_i (b_i)^2}$, and $0 < \lambda < \frac{2\sqrt{2}\nu^2}{b_{\max}}$. Then

$$\Pr[Y > \lambda] \leq \exp\left(-\frac{\lambda^2}{8\nu^2}\right).$$

[2] This algorithm can be optimized slightly (for example, we never use the count corresponding to the root, eliminating one level from the tree), and it can be modified to handle the case in which T is not a power of 2 and, more interestingly, when T is not known *a priori*.

Proof. The moment generating function of Y_i is $\mathbb{E}[\exp(hY_i)] = 1/(1 - h^2 b_i^2)$, where $|h| < 1/b_i$. Using the inequality $(1-x)^{-1} \leq 1 + 2x \leq \exp(2x)$ for $0 \leq x < 1/2$, we have $\mathbb{E}[\exp(hY_i)] \leq \exp(2h^2 b_i^2)$, if $|h| < 1/2b_i$. We now calculate, for $0 < h < 1/\sqrt{2}b_{\max}$:

$$\begin{aligned}
\Pr[Y > \lambda] &= \Pr[\exp(hY) > \exp(h\lambda)] \\
&\leq \exp(-h\lambda)\mathbb{E}[\exp(hY)] \\
&= \exp(-h\lambda) \prod_i \mathbb{E}[\exp(hY_i)] \\
&\leq \exp(-h\lambda + 2h^2\nu^2).
\end{aligned}$$

By assumption, $0 < \lambda < \frac{2\sqrt{2}\nu^2}{b_{\max}}$. We complete the proof by setting $h = \lambda/4\nu^2 < 1/\sqrt{2}b_{\max}$. □

Corollary 12.3. Let $Y, \nu, \{b_i\}_i, b_{\max}$ be as in Lemma 12.2. For $\delta \in (0,1)$ and $\nu > \max\{\sqrt{\sum_i b_i^2}, b_{\max}\sqrt{\ln(2/\delta)}\}$, we have that $\Pr[|Y| > \nu\sqrt{8\ln(2/\delta)}] \leq \delta$.

In our case, all the b_i's are the same (e.g., $b = (\log_2 T)/\varepsilon$). Taking $\nu = \sqrt{k}b$ we have the following corollary:

Corollary 12.4. For all $\lambda < \alpha(\sqrt{k}b) < 2\sqrt{2}kb = 2\sqrt{2k}\nu$,

$$\Pr[Y > \lambda] \leq e^{-\alpha^2/8}.$$

Note that we have taken the unusual step of adding noise to the count *before* counting, rather than after. In terms of the outputs it makes no difference (addition is commutative). However, it has an interesting effect on the algorithm's internal states: they are differentially private! That is, suppose the intrusion occurs at time t, and consider any $i \in [0,t]$. Since there are at most $\log_2 T$ intervals containing step i (in the algorithm we abolished the interval corresponding to the root), x_i affects at most $\log_2 T$ of the noisy counts, and so x_i is protected against the intrusion for exactly the same reason that it is protected in the algorithm's outputs. Nevertheless, the algorithm in Figure 12.1 is *not* pan-private even against a single intrusion. This is because, while its internal state and its outputs are each independently differentially private, the joint distribution does not ensure ε-differential privacy. To

see why this is so, consider an intruder that sees the internal state at time t and knows the entire data stream except x_{t+1}, and let $I = [a, b]$ be an interval containing both t and $t + 1$. Since the adversary knows $x_{[0,t]}$, it can subtract from c_I the contribution from the stream occurring up through time t (that is, it subtracts off from the observed c_I at time t the values $x_a, x_{a+1}, \ldots, x_t$, all of which it knows). From this the intruder learns the value of the Laplace draw to which c_I was initialized. When c_I is published at the end of step b, the adversary subtracts from the published value this initial draw, together with the contributions of all elements in $x_{[a,b]}$ except x_{t+1}, which it does not know. What remains is the unknown x_{t+1}.

12.3.1 Pan-private counting

Although the algorithm in Figure 12.1 is easily modified to ensure *event-level pan-privacy against a single intrusion*, we give a different algorithm here in order to introduce a powerful *bijection* technique which has proved useful in other applications. This algorithm maintains in its internal state a single noisy counter, or accumulator, as well as noise values for each interval. The output at any given time period t is the sum of the accumulator and the noise values for the intervals containing t. When an interval I ends, its associated noise value, η_I, is erased from memory.

Theorem 12.5. The counter algorithm of Figure 12.2, when run with parameters T, ε, and suffering at most one intrusion, yields an $(\varepsilon, 0)$-pan-private counter that, with probability at least $1 - \beta$ has maximum error, over its T outputs, of $O(\log(1/\beta) \cdot \log^{2.5} T/\varepsilon)$. We note also that in every round *individually* (rather than in all rounds simultaneously), with all but β probability, the error has magnitude at most $O(\log(1/\beta) \cdot \log^{1.5} T/\varepsilon)$.

Proof. The proof of accuracy is the same as that for the algorithm in Figure 12.1, relying on Corollary 12.4. We focus here on the proof of pan-privacy.

During an intrusion between atomic steps t^* and $t^* + 1$, that is, immediately following the processing of element t^* in the input stream

Pan-Private Counter (T, ε)

Initialization. Initialize $\xi = (1 + \log T)/\varepsilon$, and sample Counter $\sim \text{Lap}(\xi)$.

Intervals. For $i \in \{1, \ldots, \log T\}$, associate with each string $s \in \{0,1\}^i$ the time interval S of $2^{\log T - i}$ time periods $\{s \circ 0^{\log T - i}, \ldots s \circ 1^{\log T - i}\}$. The interval *begins in time* $s \circ 0^{\log T - i}$ and *ends in time* $s \circ 1^{\log T - i}$.

Processing. In time period $t \in \{0, 1, \ldots, T-1\}$, let $x_t \in \{0,1\}$ be the t-th input bit:

1. Counter $\leftarrow$ Counter $+ x_t$;
2. For every interval I which begins in time t, sample noise $\eta_I \sim \text{Lap}(\xi)$;
3. Let $I_1, \ldots, I_{\log T}$ be the $\log T$ intervals that contain t. Output Counter $+ \sum_{i=1}^{\log T} \eta_{I_i}$.
4. For every interval I that ends in time t, erase η_I.

Figure 12.2: Event-level pan-private counter algorithm.

(recall that we begin numbering the elements with 0), the view of the adversary consists of (1) the noisy cumulative count (in the variable "count"), (2) the interval noise values η_S in memory when the intrusion occurs, and (3) the complete sequence of all of the algorithm's outputs in rounds $0, 1, \ldots, t$. Consider adjacent databases x and x', which differ in time t, say, without loss of generality, $x_t = 1$ and $x'_t = 0$, and an intrusion immediately following time period $t^* \geq t$ (we will discuss the case $t^* < t$ below). We will describe a bijection between the vector of noise values used in executions on x and executions on x', such that corresponding noise values induce identical adversary views on x and x', and the probabilities of adjacent noise values differ only by an e^{ε} multiplicative factor. This implies ε-differential pan-privacy.

By assumption, the true count just after the time period $t^* \geq t$ is larger when the input is x than it is when the input is x'. Fix an arbitrary execution E_x when the input stream is x. This amounts to fixing the randomness of the algorithm, which in turn fixes the noise values generated. We will describe the corresponding execution $E_{x'}$ by describing how its noise values differ from those in E_x.

The program variable Counter was initialized with Laplace noise. By increasing this noise by 1 in $E_{x'}$ the value of Counter just after step t^* is identical in $E_{x'}$ and E_x. The noise variables in memory immediately following period t^* are independent of the input; these will be

unchanged in $E_{x'}$. We will make the sequence of outputs in $E_{x'}$ *identical* to those in E_x by changing a collection of $\log T$ interval noise values η_S that are *not* in memory when the adversary intrudes, so that the sum of *all* noise values in all rounds up through $t-1$ is unchanged, but the sum from round t on is larger by 1 for database x' than for x. Since we *increased* the initialization noise for Counter, we now need to *decrease* the sum of interval noise values for periods $0, \ldots, t-1$ by 1, and leave unchanged the sum of interval noise values from period t.

To do this, we find a collection of disjoint intervals whose union is $\{0, \ldots, t-1\}$. There is always such a collection, and it is always of size at most $\log T$. We can construct it iteratively by, for i decreasing from $\lfloor \log(t-1) \rfloor$ to 0, choosing the interval of size 2^i that is contained in $\{0, \ldots, t-1\}$ and is not contained in a previously chosen interval (if such an interval exists). Given this set of disjoint intervals, we notice also that they all end by time $t-1 < t \leq t^*$, and so their noises are not in memory when the adversary intrudes (just following period t^*). In total (taking into account also changing the initial noise value for Counter), the complete view seen by the adversary is identical and the probabilities of the (collection of) noise values used for x and x' differ by at most an e^{ε} multiplicative factor.

Note that we assumed $t^* \geq t$. If $t^* < t$ then the initial noise added to Counter in $E_{x'}$ will be the same as in E_x, and we need to add 1 to the sum of interval noises in every time period from t through T (the sum of interval noises before time t remains unchanged). This is done as above, by finding a disjoint collection of at most $\log T$ intervals that exactly covers $\{t, \ldots, T-1\}$. The noise values for these intervals are not yet in memory when the intrusion occurs in time $t^* < t$, and the proof follows similarly. □

12.3.2 A logarithmic (in T) lower bound

Given the upper bound of Theorem 12.5, where the error depends only poly-logarithmically on T, it is natural to ask whether *any* dependence is inherent. In this section we show that a logarithmic dependence on T is indeed inherent.

Theorem 12.6. Any differentially private event-level algorithm for counting over T rounds must have error $\Omega(\log T)$ (even with $\varepsilon = 1$).

Proof. Let $\varepsilon = 1$. Suppose for the sake of contradiction that there exists a differentially private event-level counter for streams of length T that guarantees that with probability at least $2/3$, its count at all time periods is accurate up to a maximum error of $(\log_2 T)/4$. Let $k = (\log_2 T)/4$. We construct a set S of T/k inputs as follows. Divide the T time periods into T/k consecutive phases, each of length k (except, possibly, the last one). For $i = 1, \ldots, T/k$, the i-th input $x^i \in S$ has 0 input bits everywhere except during the ith phase. That is, $x^i = 0^{k \cdot i} \circ 1^k \circ 0^{k \cdot ((T/k) - (i+1))}$.

Fo $1 \leq i \leq T/k$, we say an output *matches* i if just before the ith phase the output is less than $k/2$ and at the end of the ith phase the output is at least $k/2$. By accuracy, on input x^i the output should match i with probability at least $2/3$. By ε differential privacy, this means that for every $i, j \in [T/k]$ such that $i \neq j$, the output on input x^i should match j with probability at least

$$\begin{aligned} e^{-2\varepsilon \cdot k} &= e^{-\varepsilon \log(T^{1/2})} \\ &= e^{-\log(T^{1/2})} = 1/\sqrt{T}. \end{aligned}$$

This is a contradiction, because the events that the output matches j are disjoint for different j, and yet the sum of their probabilities on input x^i exceeds 1. □

12.4 Average case error for query release

In Sections 4 and 5, we considered various mechanisms for solving the private query release problem, where we were interested in *worst case error*. That is, given a class of queries $\mathcal{Q}$, of size $|\mathcal{Q}| = k$, we wished to recover a vector of answers $\hat{a} \in \mathbb{R}^k$ such that for *each* query $f_i \in \mathcal{Q}$, $|f_i(x) - \hat{a}_i| \leq \alpha$ for some worst-case error rate α. In other words, if we let $a \in \mathbb{R}^k$ denote the vector of *true* answers, with $a_i \equiv f_i(x)$, then we require a bound of the form: $\|a - \hat{a}\|_\infty \leq \alpha$. In this section, we consider a weakened utility guarantee, on the ℓ_2 (rather than ℓ_∞) error: a bound of the form $\|a - \hat{a}\|_2 \leq \alpha$. A bound of this form does not guarantee

that we have low error for *every* query, but it does guarantee that on average, we have small error.

Although this sort of bound is weaker than worst-case error, the mechanism is particularly simple, and it makes use of an elegant geometric view of the query release problem that we have not seen until now.

Recall that we can view the database x as a vector $x \in \mathbb{N}^{|\mathcal{X}|}$ with $\|x\|_1 = n$. We can similarly also view the queries $f_i \in \mathcal{Q}$ as vectors $f_i \in \mathbb{N}^{|\mathcal{X}|}$, such that $f_i(x) = \langle f_i, x \rangle$. It will therefore be helpful to view our class of queries $\mathcal{Q}$ as a matrix $A \in \mathbb{R}^{k \times |\mathcal{X}|}$, with the ith row of A being the vector f_i. We can then see that our answer vector $a \in \mathbb{R}^k$ is, in matrix notation:

$$A \cdot x = a.$$

Let's consider the domain and range of A when viewed as a linear map. Write $B_1 = \{x \in \mathbb{R}^{|\mathcal{X}|} : \|x\|_1 = 1\}$ denote the unit ℓ_1 ball in $|\mathcal{X}|$ dimensional space. Observe that $x \in nB_1$, since $\|x\|_1 = n$. We will refer to nB_1 as "Database Space." Write $K = AB_1$. Note similarly that for all $x \in nB_1$, $a = A \cdot x \in nK$. We will refer to nK as "answer space." We make a couple of observations about K: Note that because B_1 is centrally symmetric, so is K — that is, $K = -K$. Note also that $K \subset \mathbb{R}^k$ is a convex polytope with vertices $\pm A^1, \ldots, \pm A^{|\mathcal{X}|}$ equal to the columns of A, together with their negations.

The following algorithm is extremely simple: it simply answers every query independently with the Laplace mechanism, and then *projects back into answer space*. In other words, it adds independent Laplace noise to every query, which as we have seen, by itself leads to distortion that is linear in k (or at least $\sqrt{k}$, if we relax to (ε, δ)-differential privacy). However, the resulting vector $\tilde{a}$ of answers is likely not consistent with *any* database $y \in nB_1$ in database space. Therefore, rather than returning $\tilde{a}$, it instead returns some consistent answer vector $\hat{a} \in nK$ that is as close to $\tilde{a}$ as possible. As we will see, this projection step improves the accuracy of the mechanism, while having no effect on privacy (since it is just post-processing!)

We first observe that Project is differentially private.

Theorem 12.7. For any $A \in [0,1]^{k \times |\mathcal{X}|}$, **Project**$(x, A, \varepsilon)$ preserves (ε, δ)-differential privacy.

Algorithm 18 The K-Projected Laplace Mechanism. It takes as input a matrix $A \in [0,1]^{k \times |\mathcal{X}|}$, a database $x \in nB_1$, and a privacy parameters ε and δ.

Project$(x, A, \varepsilon, \delta)$:

Let $a = A \cdot x$

For each $i \in [k]$, sample $\nu_i \sim \text{Lap}(\sqrt{8k \ln(1/\delta)}/\varepsilon)$, and let $\tilde{a} = a + \nu$.

Output $\hat{a} = \arg\min_{\hat{a} \in nK} \|\hat{a} - \tilde{a}\|_2^2$.

Proof. We simply note that $\tilde{a}$ is the output of the Laplace mechanism on k sensitivity 1 queries, which is (ε, δ)-differentially private by Theorems 3.6 and 3.20 . Finally, since $\hat{a}$ is derived from $\tilde{a}$ without any further access to the private data, the release of $\hat{a}$ is differentially private by the post-processing guarantee of differential privacy, Proposition 2.1. □

Theorem 12.8. For any class of linear queries A and database x, let $a = A \cdot x$ denote the true answer vector. Let $\hat{a}$ denote the output of the mechanism **Project**: $\hat{a} =$ **Project**(x, A, ε). With probability at least $1 - \beta$:

$$\|a - \hat{a}\|_2^2 \leq \frac{kn\sqrt{192 \ln(1/\delta) \ln(2|\mathcal{X}|/\beta)}}{\varepsilon}.$$

To prove this theorem, we will introduce a couple of simple concepts from convex geometry. For a convex body $K \subset \mathbb{R}^k$, its *polar body* is K° defined to be $K^\circ = \{y \in \mathbb{R}^k : \langle y, x \rangle \leq 1 \text{ for all } x \in K\}$. The *Minkowski Norm* defined by a convex body K is

$$\|x\|_K \equiv \min\{r \in \mathbb{R} \text{ such that } x \in rK\}.$$

The *dual norm* of $\|x\|_K$ is the Minkowski norm induced by the polar body of K, i.e., $\|x\|_{K^\circ}$. This norm also has the following form:

$$\|x\|_{K^\circ} = \max_{y \in K} \langle x, y \rangle.$$

The key fact we will use is *Holder's Inequality*, which is satisfied by all centrally symmetric convex bodies K:

$$|\langle x, y \rangle| \leq \|x\|_K \|y\|_{K^\circ}.$$

Proof of Theorem 12.8. The proof will proceed in two steps. First we will show that: $\|a - \hat{a}\|_2^2 \leq 2\langle \hat{a} - a, \tilde{a} - a\rangle$, and then we will use Holder's inequality to bound this second quantity.

Lemma 12.9.

$$\|a - \hat{a}\|_2^2 \leq 2\langle \hat{a} - a, \tilde{a} - a\rangle$$

Proof. We calculate:

$$\begin{aligned}\|\hat{a} - a\|_2^2 &= \langle \hat{a} - a, \hat{a} - a\rangle \\ &= \langle \hat{a} - a, \tilde{a} - a\rangle + \langle \hat{a} - a, \hat{a} - \tilde{a}\rangle \\ &\leq 2\langle \hat{a} - a, \tilde{a} - a\rangle.\end{aligned}$$

The inequality follows from calculating:

$$\begin{aligned}\langle \hat{a} - a, \tilde{a} - a\rangle &= \|\tilde{a} - a\|_2^2 + \langle \hat{a} - \tilde{a}, \tilde{a} - a\rangle \\ &\geq \|\hat{a} - \tilde{a}\|_2^2 + \langle \hat{a} - \tilde{a}, \tilde{a} - a\rangle \\ &= \langle \hat{a} - \tilde{a}, \hat{a} - a\rangle,\end{aligned}$$

Where the final inequality follows because by choice of $\hat{a}$, for all $a' \in nK$: $\|\tilde{a} - \hat{a}\|_2^2 \leq \|\tilde{a} - a'\|_2^2$. □

We can now complete the proof. Recall that by definition, $\tilde{a} - a = \nu$, the vector of i.i.d. Laplace noise added by the Laplace mechanism. By Lemma 12.9 and Holder's inequality, we have:

$$\begin{aligned}\|a - \hat{a}\|_2^2 &\leq 2\langle \hat{a} - a, \nu\rangle \\ &\leq 2\|\hat{a} - a\|_K \|\nu\|_{K^\circ}.\end{aligned}$$

We bound these two terms separately. Since by definition $\hat{a}, a \in nK$, we have $\max(\|\hat{a}\|_K, \|a\|_K) \leq n$, and so by the triangle inequality, $\|\hat{a} - a\|_K \leq 2n$.

Next, observe that since $\|\nu\|_{K^\circ} = \max_{y \in K}\langle y, \nu\rangle$, and since the maximum of a linear function taken over a polytope is attained at a vertex, we have: $\|\nu\|_{K^\circ} = \max_{i \in [|\mathcal{X}|]} |\langle A^i, \nu\rangle|$.

Because each $A^i \in \mathbb{R}^k$ is such that $\|A^i\|_\infty \leq 1$, and recalling that for any scalar q, if $Z \sim \text{Lap}(b)$, then $qZ \sim \text{Lap}(qb)$, we can apply Lemma by

Lemma 12.2 to bound the weighted sums of Laplace random variables $\langle A^i, \nu \rangle$. Doing so, we have that with probability at least $1 - \beta$:

$$\max_{i \in [|\mathcal{X}|]} |\langle A^i, \nu \rangle| \leq \frac{8k\sqrt{\ln(1/\delta)\ln(|\mathcal{X}|/\beta)}}{\epsilon}.$$

Combining all of the above bounds, we get that with probability $1 - \beta$:

$$\|a - \hat{a}\|_2^2 \leq \frac{16nk\sqrt{\ln(1/\delta)\ln(|\mathcal{X}|/\beta)}}{\epsilon}. \qquad \square$$

Let's interpret this bound. Observe that $\|a - \hat{a}\|_2^2 = \sum_{i=1}^k (a_i - \hat{a}_i)^2$, and so this is a bound on the sum of squared errors over all queries. Hence, the *average* per-query squared error of this mechanism is only:

$$\frac{1}{k}\sum_{i=1}^{k}(a_i - \hat{a}_i)^2 \leq \frac{16n\sqrt{\ln(1/\delta)\ln(|\mathcal{X}|/\beta)}}{\epsilon}.$$

In contrast, the private multiplicative weights mechanism guarantees that $\max_{i \in [k]} |a_i - \hat{a}_i| \leq \tilde{O}(\sqrt{n} \log |\mathcal{X}|^{1/4}/\varepsilon^{1/2})$, and so matches the average squared error guarantee of the projected Laplace mechanism, with a bound of: $\tilde{O}(n\sqrt{\log |\mathcal{X}|}/\varepsilon)$. However, the multiplicative weights mechanism (and especially its privacy analysis) its much more complex than the Projected Laplace mechanism! In particular, the *private* part of the K-Projected Laplace mechanism is simply the Laplace mechanism itself, and requires no coordination between queries. Interestingly — and, it turns out, necessarily — coordination occurs in the projection phase. Since projection is in post-precessing, it incurs no further privacy loss; indeed, it can be carrie out (online, if necessary) by the data analyst himself.

12.5 Bibliographical notes

The local model of data privacy has its roots in randomized response, as first proposed by Warner in 1965 [84]. The local model was formalized by Kasiviswanathan et al. [52] in the context of learning, who proved that private learning in the local modal is equivalent to non-private

learning in the statistical query (SQ) model. The set of queries which can be *released* in the local model was shown to be exactly equal to the set of queries that can be *agnostically learned* in the SQ model by Gupta et al. [38].

Pan-Privacy was introduced by Dwork et al. [27], and further explored by Mir et al. [62]. The pan-private density estimation, as well as a low-memory variant using hashing, appear in [27].

Privacy under continual observation was introduced by Dwork et al. [26]; our algorithm for counting under continual observation is from that paper, as is the lower bound on error. Similar algorithms were given by Chan et al. [11]. The proof of concentration of measure inequality for the sums of Laplace random variables given in Lemma 12.2 is from [11].

The Projected Laplace mechanism for achieving low average error was given by Nikolov et al.[66], who also give *instance optimal* algorithms for the (average error) query release problem for any class of queries. This work extends a line of work on the connections between differential privacy and geometry started by Hardt and Talwar [45], and extended by Bhaskara et al. [5] and Dwork et al. [30].

Dwork, Naor, and Vadhan proved an exponential gap between the number of queries that can be answered (with non-trivial error) by stateless and stateful differentially private mechanisms [29]. The lesson learned — that coordination is essential for accurately and privately answering very large numbers of queries — seems to rule out the independent noise addition in the Projected Laplace mechanism. The statefulness of that algorithm appears in the projection step, resolving the paradox.

13

Reflections

13.1 Toward practicing privacy

Differential Privacy was designed with internet-scale data sets in mind. Reconstruction attacks along the lines of those in Section 8 can be carried out by a *polynomial time* bounded adversary asking only $O(n)$ queries on databases of size n. When n is on the order of hundreds of millions, and each query requires a linear amount of computation, such an attack is unrealistic, even thought the queries can be parallelized. This observation led to the earliest steps toward differential privacy: If the adversary is restricted to a *sublinear* number of counting queries, then $o(\sqrt{n})$ noise per query — less than the sampling error! — is sufficient for preserving privacy (Corollary 3.21).

To what extent can differential privacy be brought to bear on smaller data sets, or even targeted attacks that isolate a small subset of a much larger database, without destroying statistical utility? First, an analysis may require a number of queries that begins to look something like the size of this smaller set. Second, letting n now denote the size of the smaller set or small database, and letting k be the number of queries, fractional errors on the order of $\sqrt{k}/n$ are harder to ignore when n is small. Third, the $\sqrt{\ln(1/\delta)}/\varepsilon$ factor in the advanced

composition theorem becomes significant. Keeping in mind the reconstruction attacks when noise is $o(\sqrt{n})$, there appears to be little room to maneuver for arbitrary sets of $k \approx n$ low-sensitivity queries.

There are several promising lines of research for addressing these concerns.

The Query Errors Don't Tell the Whole Story. As an example of this phenomenon, consider the problem of linear regression. The input is a collection of labeled data points of the form (x, y), where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, for arbitrary dimension d. The goal is to find $\theta \in \mathbb{R}^d$ that "predicts" y "as well as possible," given x, under the assumption that the relationship is linear. If the goal is simply to "explain" the given data set, differential privacy may well introduce unacceptable error. Certainly the specific algorithm that simply computes

$$\text{argmin}_\theta |\sum_{i=1}^{n} \theta \cdot x_i - y_i|^2$$

and adds appropriately scaled Laplace noise independently to each coordinate of θ may produce a $\tilde{\theta}$ that differs substantially from θ. But if the goal is to learn a predictor that will do well for *future, unseen* inputs (x, y) then a slightly different computation is used to avoid overfitting, and the (possibly large) difference between the private and non-private coefficient vectors does *not* translate into a gap in classification error! A similar phenomenon has been observed in model fitting.

Less Can Be More. Many analyses ask for more than they actually use. Exploitation of this principle is at the heart of Report Noisy Max, where for the accuracy "price" of one measurement we learn one of the largest of many measurements. By asking for "less" (that is, not requiring that all noisy measurements be released, but rather only asking for the largest one), we obtain "more" (better accuracy). A familiar principle in privacy is to *minimize* collection and reporting. Here we see this play out in the realm of what must be *revealed*, rather than what must be used in the computation.

Quit When You are NOT Ahead. This is the philosophy behind Propose-Test-Release, in which we test in a privacy-preserving way

that small noise is sufficient for a particular intended computation on the given data set.

Algorithms with Data-Dependent Accuracy Bounds. This can be viewed as a generalization of Quit When You are Not Ahead. Algorithms with data-dependent accuracy bounds can deliver excellent results on "good" data sets, as in Propose-Test-Release, and the accuracy can degrade gradually as the "goodness" decreases, an improvement over Propose-Test-Release.

Exploit "Nice" Query Sets. When (potentially large) sets of linear queries are presented as a batch it is possible, by analyzing the geometry of the query *matrix* to obtain higher quality answers than would be obtained were the queries answered independently[1].

Further Relaxation of Differential Privacy We have seen that (ϵ, δ)-differential privacy is a meaningful relaxation of differential privacy that can provide substantially improved accuracy bounds. Moreover, such a relaxation can be essential to these improvements. For example, Propose-Test-Release algorithms can only offer (ε, δ)-differential privacy for $\delta > 0$. What about other, but still meaningful, relaxations of differential privacy? *Concentrated Differential Privacy* is such a relaxation that is incomparable to (ε, δ)-differential privacy and that permits better accuracy. Roughly speaking, it ensures that large privacy loss happens with very small probability; for example, for all k the probability of privacy loss $k\varepsilon$ falls exponentially in k^2. In contrast, (ε, δ)-differential privacy is consistent with having *infinite* privacy loss with probability δ; on the other hand, privacy lost 2ε can happen in concentrated differential privacy with constant probability, while in (ε, δ)-differential privacy it will only occur with probability bounded by δ, which we typically take to be cryptographically small.

Why might we feel comfortable with this relaxation? The answer lies in behavior under composition. As an individual's data participate

[1]More accurately, the analysis is of the object $K = AB_1^k$, where A is the query matrix and B_1^k is the k-dimensional L_1 ball; note that K is the feasible region in answer space when the database has one element.

in many databases and many different computations, perhaps the real worry is the combined threat of multiple exposures. This is captured by privacy under composition. Concentrated differential privacy permits better accuracy while yielding the same behavior under composition as (ε, δ) (and $(\varepsilon, 0)$) differential privacy.

Differential privacy also faces a number of cultural challenges. One of the most significant is non-algorithmic thinking. Differential privacy is a property of an algorithm. However, many people who work with data describe their interactions with the data in fundamentally non-algorithmic terms, such as, "First, I *look at* the data." Similarly, data cleaning is often described in non-algorithmic terms. If data are reasonably plentiful, and the analysts are energetic, then the "Raw Data" application of the Subsample and Aggregate methodology described in Example 7.3 suggests a path toward enabling non-algorithmic, interactions by trusted analysts who will follow directions. In general, it seems plausible that on high-dimensional and on internet-scale data sets non-algorithmic interactions will be the exception.

What about ε? In Example 3.7 we applied Theorem 3.20 to conclude that to bound the cumulative lifetime privacy loss at $\varepsilon = 1$ with probability $1 - e^{-32}$, over participation in $10,000$ databases, it is sufficient that each database be $(1/801, 0)$-differentially private. While $k = 10,000$ may be an overestimate, the dependence on k is fairly weak ($\sqrt{k}$), and in the worst case these bounds are tight, ruling out a more relaxed bound than $\varepsilon_0 = 1/801$ for each database *over the lifetime of the database*. This is simply too strict a requirement in practice.

Perhaps we can ask a different question: Fix ε, say, $\varepsilon = 1$ or $\varepsilon = 1/10$; now ask: How can multiple ε's be apportioned? Permitting ε privacy loss *per query* is too weak, and ε loss over the lifetime of the database is too strong. Something in between, say, ε per study or ε per researcher, may make sense, although this raises the questions of who is a "researcher" and what constitutes a "study." This affords substantially more protection against accidental and intentional privacy compromise than do current practices, from enclaves to confidentiality contracts.

A different proposal is less prescriptive. This proposal draws from second-generation regulatory approaches to reducing environmental

degradation, in particular pollution release registries such as the Toxic Release Inventory that have been found to encourage better practices through transparency. Perhaps a similar effect could arise with private data analysis: an Epsilon Registry describing data uses, granularity of privacy protection, a "burn rate" of privacy loss per unit time, and a cap on total privacy loss permitted before data are retired, when accompanied with a financial penalty for infinite (or very large) loss, can lead to innovation and competition, deploying the talents and resources of a larger set of researchers and privacy professionals in the search for differentially private algorithms.

13.2 The differential privacy lens

An online etymological dictionary describes the original 18th century meaning of the term of the word "statistics" as "science dealing with data about the condition of a state or community." This resonates with differential privacy in the breach: if the presence or absence of the data of a small number of individuals changes the outcome of an analysis then in some sense the outcome is "about" these few individuals, and is not describing the condition of the community as a whole. Put differently, stability to small perturbations in the data is both the hallmark of differential privacy and the essence of a common conception of the term "statistical." Differential privacy is enabled by stability (Section 7) and ensures stability (by definition). In some sense it forces all queries to be statistical in nature. As stability is also increasingly understood to be a key necessary and sufficient condition for learnability, we observe a tantalizing moral equivalence between learnability, differential privacy, and stability.

With this in mind, it is not surprising that differential privacy is also a means to ends other than privacy, and indeed we saw this with game theory in Section 10. The power of differential privacy comes from its amenability to composition. Just as composition allows us to build complex differentially private algorithms from smaller differentially private building blocks, it provides a programming language for constructing stable algorithms for complex analytical tasks. Consider, for example, the problem of eliciting a set of bidder values, and using them to price

a collection of goods that are for sale. Informally, *Walrasian equilibrium prices* are prices such that every individual can simultaneously purchase their *favorite* bundle of goods *given the prices*, while ensuring that demand exactly equals the supply of each good. It would seem at first blush, then, that simply computing these prices, and assigning each person their favorite bundle of goods given the prices would yield a mechanism in which agents were incentivized to tell the truth about their valuation function — since how could any agent do better than receiving their favorite bundle of goods? However, this argument fails — because in a Walrasian equilibrium, agents receive their favorite bundle of goods *given the prices*, but the prices are computed as a function of the reported valuations, so an industrious but dishonest agent could potentially gain by manipulating the computed prices. However, this problem is solved (and an approximately truthful mechanism results) if the equilibrium prices are computed using a differentially private algorithm — precisely because individual agents have almost no effect on the distribution of prices computed. Note that this application is made possible by the use of the tools of differential privacy, but is completely orthogonal to privacy concerns. More generally, this connection is more fundamental: computing *equilibria* of various sorts using algorithms that have the stability property guaranteed by differential privacy leads to approximately truthful mechanisms implementing these equilibrium outcomes.

Differential privacy also helps in ensuring generalizability in adaptive data analysis. Adaptivity means that the questions asked and hypotheses tested depend on outcomes of earlier questions. Generalizability means that the outcome of a computation or a test on the data set is close to the ground truth of the distribution from which the data are sampled. It is known that the naive paradigm of answering queries with the exact empirical values on a fixed data set fails to generalize even under a limited amount of adaptive questioning. Remarkably, answering with differential privacy not only ensures privacy, but with high probability it ensures generalizability even for exponentially many adaptively chosen queries. Thus, the deliberate introduction of noise using the techniques of differential privacy has profound and promising implications for the validity of traditional scientific inquiry.

Appendices

A

The Gaussian Mechanism

Let $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^d$ be an arbitrary d-dimensional function, and define its ℓ_2 sensitivity to be $\Delta_2 f = \max_{\text{adjacent}x,y} \|f(x) - f(y)\|_2$. The *Gaussian Mechanism with parameter* σ adds noise scaled to $\mathcal{N}(0, \sigma^2)$ to each of the d components of the output.

Theorem A.1. 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.

Proof. There is a database D and a query f, and the mechanism will return $f(D)+\eta$, where the noise is normally distributed. We are adding noise $\mathcal{N}(0, \sigma^2)$. For now, assume we are talking about real-valued functions, so

$$\Delta f = \Delta_1 f = \Delta_2 f.$$

We are looking at

$$\left| \ln \frac{e^{(-1/2\sigma^2)x^2}}{e^{(-1/2\sigma^2)(x+\Delta f)^2}} \right|. \tag{A.1}$$

We are investigating the probability, given that the database is D, of observing an output that occurs with a very different probability

under D than under an adjacent database D', where the probability space is the noise generation algorithm. The numerator in the ratio above describes the probability of seeing $f(D) + x$ when the database is D, the denominator corresponds the probability of seeing *this same value* when the database is D'. This is a ratio of probabilities, so it is always positive, but the logarithm of the ratio may be negative. Our random variable of interest — the privacy loss — is

$$\ln \frac{e^{(-1/2\sigma^2)x^2}}{e^{(-1/2\sigma^2)(x+\Delta f)^2}}$$

and we are looking at its absolute value.

$$\begin{aligned}\left|\ln \frac{e^{(-1/2\sigma^2)x^2}}{e^{(-1/2\sigma^2)(x+\Delta f)^2}}\right| &= |\ln e^{(-1/2\sigma^2)[x^2-(x+\Delta f)^2]}| \\ &= |-\frac{1}{2\sigma^2}[x^2-(x^2+2x\Delta f+\Delta f^2)]| \\ &= |\frac{1}{2\sigma^2}(2x\Delta f+(\Delta f)^2)|. \end{aligned} \tag{A.2}$$

This quantity is bounded by ε whenever $x < \sigma^2\varepsilon/\Delta f - \Delta f/2$. To ensure privacy loss bounded by ε with probability at least $1-\delta$, we require

$$\Pr[|x| \geq \sigma^2\varepsilon/\Delta f - \Delta f/2] < \delta,$$

and because we are concerned with $|x|$ we will find σ such that

$$\Pr[x \geq \sigma^2\varepsilon/\Delta f - \Delta f/2] < \delta/2.$$

We will assume throughout that $\varepsilon \leq 1 \leq \Delta f$.

We will use the tail bound

$$\Pr[x > t] \leq \frac{\sigma}{\sqrt{2\pi}} e^{-t^2/2\sigma^2}.$$

We require:

$$\begin{aligned} &\frac{\sigma}{\sqrt{2\pi}}\frac{1}{t}e^{-t^2/2\sigma^2} < \delta/2 \\ \Leftrightarrow\ &\sigma\frac{1}{t}e^{-t^2/2\sigma^2} < \sqrt{2\pi}\delta/2 \\ \Leftrightarrow\ &\frac{t}{\sigma}e^{t^2/2\sigma^2} > 2/\sqrt{2\pi}\delta \\ \Leftrightarrow\ &\ln(t/\sigma) + t^2/2\sigma^2 > \ln(2/\sqrt{2\pi}\delta). \end{aligned}$$

Taking $t = \sigma^2\varepsilon/\Delta f - \Delta f/2$, we get

$$\ln((\sigma^2\varepsilon/\Delta f - \Delta f/2)/\sigma) + (\sigma^2\varepsilon/\Delta f - \Delta f/2)^2/2\sigma^2 > \ln(2/\sqrt{2\pi}\delta)$$
$$= \ln\left(\sqrt{\frac{2}{\pi}}\frac{1}{\delta}\right).$$

Let us write $\sigma = c\Delta f/\varepsilon$; we wish to bound c. We begin by finding the conditions under which the first term is non-negative.

$$\begin{aligned}\frac{1}{\sigma}\left(\sigma^2\frac{\varepsilon}{\Delta f} - \frac{\Delta f}{2}\right) &= \frac{1}{\sigma}\left[\left(c^2\frac{(\Delta f)^2}{\varepsilon^2}\right)\frac{\varepsilon}{\Delta f} - \frac{\Delta f}{2}\right] \\ &= \frac{1}{\sigma}\left[c^2\left(\frac{\Delta f}{\varepsilon}\right) - \frac{\Delta f}{2}\right] \\ &= \frac{\varepsilon}{c\Delta f}\left[c^2\left(\frac{\Delta f}{\varepsilon}\right) - \frac{\Delta f}{2}\right] \\ &= c - \frac{\varepsilon}{2c}.\end{aligned}$$

Since $\varepsilon \leq 1$ and $c \geq 1$, we have $c - \varepsilon/(2c) \geq c - 1/2$. So $\ln(\frac{1}{\sigma}(\sigma^2\frac{\varepsilon}{\Delta f} - \frac{\Delta f}{2})) > 0$ provided $c \geq 3/2$. We can therefore focus on the t^2/σ^2 term.

$$\begin{aligned}\left(\frac{1}{2\sigma^2}\frac{\sigma^2\varepsilon}{\Delta f} - \frac{\Delta f}{2}\right)^2 &= \frac{1}{2\sigma^2}\left[\Delta f\left(\frac{c^2}{\varepsilon} - \frac{1}{2}\right)\right]^2 \\ &= \left[(\Delta f)^2\left(\frac{c^2}{\varepsilon} - \frac{1}{2}\right)\right]^2\left[\frac{\varepsilon^2}{c^2(\Delta f)^2}\right]\frac{1}{2} \\ &= \frac{1}{2}\left(\frac{c^2}{\varepsilon} - \frac{1}{2}\right)^2\frac{\varepsilon^2}{c^2} \\ &= \frac{1}{2}(c^2 - \varepsilon + \varepsilon^2/4c^2).\end{aligned}$$

Since $\varepsilon \leq 1$ the derivative of $(c^2 - \varepsilon + \varepsilon^2/4c^2)$ with respect to c is positive in the range we are considering ($c \geq 3/2$), so $c^2 - \varepsilon + \varepsilon^2/4c^2 \geq c^2 - 8/9$ and it suffices to ensure

$$c^2 - 8/9 > 2\ln\left(\sqrt{\frac{2}{\pi}}\frac{1}{\delta}\right).$$

In other words, we need that

$$c^2 > 2\ln(\sqrt{2/\pi}) + 2\ln(1/\delta) + \ln(e^{8/9}) = \ln(2/\pi) + \ln(e^{8/9}) + 2\ln(1/\delta),$$

which, since $(2/\pi)e^{8/9} < 1.55$, is satisfied whenever $c^2 > 2\ln(1.25/\delta)$.

Let us parition $\mathbb{R}$ as $\mathbb{R} = R_1 \cup R_2$, where $R_1 = \{x \in \mathbb{R} : |x| \leq c\Delta f/\varepsilon\}$ and $R_2 = \{x \in \mathbb{R} : |x| > c\Delta f/\varepsilon\}$. Fix any subset $S \subseteq \mathbb{R}$, and define

$$S_1 = \{f(x) + x \,|\, x \in R_1\}$$
$$S_2 = \{f(x) + x \,|\, x \in R_2\}.$$

We have

$$\begin{aligned}
\Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x) + x \in S] &= \Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x) + x \in S_1] \\
&\quad + \Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x) + x \in S_2] \\
&\leq \Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(x) + x \in S_1] + \delta \\
&\leq e^{\varepsilon}\left(\Pr_{x\sim\mathcal{N}(0,\sigma^2)}[f(y) + x \in S_1]\right) + \delta,
\end{aligned}$$

yielding (ε, δ)-differential privacy for the Gaussian mechanism in one dimension.

High Dimension. To extend this to functions in R^m, define $\Delta f = \Delta_2 f$. We can now repeat the argument, using Euclidean norms. Let v be any vector satisfying $\|v\| \leq \Delta f$. For a fixed pair of databases x, y we are interested in $v = f(x) - f(y)$, since this is what our noise must obscure. As in the one dimensional, case we seek conditions on σ under which the privacy loss

$$\left|\ln \frac{e^{(-1/2\sigma^2)\|x-\mu\|^2}}{e^{(-1/2\sigma^2)\|x+v-\mu\|^2}}\right|$$

is bounded by ε; here x is chosen from $\mathcal{N}(0, \Sigma)$, where (Σ) is a diagonal matrix with entries σ^2, whence $\mu = (0, \ldots, 0)$.

$$\left| \ln \frac{e^{(-1/2\sigma^2)\|x-\mu\|^2}}{e^{(-1/2\sigma^2)\|x+v-\mu\|^2}} \right| = |\ln e^{(-1/2\sigma^2)[\|x-\mu\|^2 - \|x+v-\mu\|^2]}|$$
$$= \left| \frac{1}{2\sigma^2}(\|x\|^2 - \|x+v\|^2)) \right|.$$

We will use the fact that the distribution of a spherically symmetric normal is independent of the orthogonal basis from which its constituent normals are drawn, so we may work in a basis that is alligned with v. Fix such a basis $b_1, \ldots, b_m$, and draw x by first drawing signed lengths $\lambda_i \sim \mathcal{N}(0, \sigma^2)$, for $i \in [m]$, then defining $x^{[i]} = \lambda_i b_i$, and finally letting $x = \sum_{i=1}^m x^{[i]}$. Assume without loss of generality that b_1 is parallel to v. We are interested in $|\, \|x\|^2 - \|x+v\|^2 \,|$.

Consider the right triangle with base $v + x^{[1]}$ and edge $\sum_{i=2}^m x^{[i]}$ orthogonal to v. The hypotenuse of this triangle is $x + v$.

$$\|x+v\|^2 = \|v + x^{[1]}\|^2 + \sum_{i=2}^m \|x^{[i]}\|^2$$
$$\|x\|^2 = \sum_{i=1}^m \|x^{[i]}\|^2.$$

Since v is parallel to $x^{[1]}$ we have $\|v + x^{[1]}\|^2 = (\|v\| + \lambda_1)^2$. Thus, $\|x+v\|^2 - \|x\|^2 = \|v\|^2 + 2\lambda_1 \cdot \|v\|$. Recall that $\|v\| \leq \Delta f$, and $\lambda \sim \mathcal{N}(0, \sigma)$, so we are now exactly back in the one-dimensional case, writing λ_1 instead of x in Equation (A.2):

$$\left| \frac{1}{2\sigma^2}(\|x\|^2 - \|x+v\|^2)) \right| \leq \left| \frac{1}{2\sigma^2}(2\lambda_1 \Delta f - (\Delta f)^2) \right|$$

and the rest of the argument proceeds as above. □

The argument for the high dimensional case highlights a weakness of (ε, δ)-differential privacy that does not exist for $(\varepsilon, 0)$-differential privacy. Fix a database x. In the $(\varepsilon, 0)$-case, the guarantee of indistinguishability holds for all adjacent databases *simultaneously*. In the

(ε, δ) case indistinguishability only holds "prospectively," i.e., for any fixed y adjacent to x, the probability that the mechanism will allow the adversary to distinguish x from y is small. In the proof above, this is manifested by the fact that we fixed $v = f(x) - f(y)$; we did not have to argue about all possible directions of v simultaneously, and indeed we cannot, as once we have fixed our noise vector $x \sim \mathcal{N}(0, \Sigma)$, so that the output on x is $o = f(x) + x$, there may exist an adjacent y such that output $o = f(x) + x$ is much more likely when the database is y than it is on x.

A.1 Bibliographic notes

Theorem A.1 is folklore initially observed by the authors of [23]. A generalization to non-spherical gaussian noise appears in [66].

B

Composition Theorems for (ε, δ)-DP

B.1 Extension of Theorem 3.16

Theorem B.1. Let $T_1(D) : D \mapsto T_1(D) \in \mathcal{C}_1$ be an (ϵ, δ)-d.p. function, and for any $s_1 \in \mathcal{C}_1$, $T_2(D, s_1) : (D, s_1) \mapsto T_2(D, s_1) \in \mathcal{C}_2$ be an (ϵ, δ)-d.p. function given the second input s_1. Then we show that for any neighboring D, D', for any $S \subseteq \mathcal{C}_2 \times \mathcal{C}_1$, we have, using the notation in our paper

$$P((T_2, T_1) \in S) \leq e^{2\epsilon} P'((T_2, T_1) \in S) + 2\delta. \tag{B.1}$$

Proof. For any $C_1 \subseteq \mathcal{C}_1$, define

$$\mu(C_1) = \left(P(T_1 \in C_1) - e^{\epsilon} P'(T_1 \in C_1)\right)_+,$$

then μ is a measure on $\mathcal{C}_1$ and $\mu(\mathcal{C}_1) \leq \delta$ since T_1 is (ϵ, δ)-d.p. As a result, we have for all $s_1 \in \mathcal{C}_1$,

$$P(T_1 \in ds_1) \leq e^{\epsilon} P'(T_1 \in ds_1) + \mu(ds_1). \tag{B.2}$$

Also note that by the definition of (ϵ, δ)-d.p., for any $s_1 \in \mathcal{C}_1$,

$$\begin{aligned} P((T_2, s_1) \in S) &\leq \left(e^{\epsilon} P'((T_2, s_1) \in S) + \delta\right) \wedge 1 \\ &\leq \left(e^{\epsilon} P'((T_2, s_1) \in S)\right) \wedge 1 + \delta. \end{aligned} \tag{B.3}$$

Then (B.2) and (B.3) give (B.1):

$$\begin{aligned}
P((T_2, T_1) \in S) &\leq \int_{S_1} P((T_2, s_1) \in S) P(T_1 \in ds_1) \\
&\leq \int_{S_1} \left((e^{\epsilon} P'((T_2, s_1) \in S)) \wedge 1 + \delta \right) P(T_1 \in ds_1) \\
&\leq \int_{S_1} \left((e^{\epsilon} P'((T_2, s_1) \in S)) \wedge 1 \right) P(T_1 \in ds_1) + \delta \\
&\leq \int_{S_1} \left((e^{\epsilon} P'((T_2, s_1) \in S)) \wedge 1 \right) \\
&\quad \times \left(e^{\epsilon} P'(T_1 \in ds_1) + \mu(ds_1) \right) + \delta \\
&\leq e^{2\epsilon} \int_{S_1} P'((T_2, s_1) \in S) P'(T_1 \in ds_1) + \mu(S_1) + \delta \\
&\leq e^{2\epsilon} P'((T_2, T_1) \in S) + 2\delta. \qquad \text{(B.4)}
\end{aligned}$$

In the equations above, S_1 denotes the projection of S onto $\mathcal{C}_1$. The event $\{(T_2, s_1) \in S\}$ refers to $\{(T_2(D, s_1), s_1) \in S\}$ (or $\{(T_2(D', s_1), s_1) \in S\}$). □

Using induction, we have:

Corollary B.2 (general composition theorem for (ϵ, δ)-d.p. algorithms)**.** Let $T_1 : D \mapsto T_1(D)$ be (ϵ, δ)-d.p., and for $k \geq 2$, $T_k : (D, s_1, \ldots, s_{k-1}) \mapsto T_k(D, s_1, \ldots, s_{k-1}) \in \mathcal{C}_k$ be (ϵ, δ)-d.p., for all given $(s_{k-1}, \ldots, s_1) \in \bigotimes_{j=1}^{k-1} \mathcal{C}_j$. Then for all neighboring D, D' and all $S \subseteq \bigotimes_{j=1}^{k} \mathcal{C}_j$

$$P((T_1, \ldots, T_k) \in S) \leq e^{k\epsilon} P'((T_1, \ldots, T_k) \in S) + k\delta.$$

Acknowledgments

We would like to thank many people for providing careful comments and corrections on early drafts of this book, including Vitaly Feldman, Justin Hsu, Simson Garfinkel, Katrina Ligett, Dong Lin, David Parkes, Ryan Rogers, Guy Rothblum, Ian Schmutte, Jon Ullman, Salil Vadhan, Zhiwei Steven Wu, and the anonymous referees. This book was used in a course taught by Salil Vadhan and Jon Ullman, whose students also provided careful feedback. This book has also benefited from conversations with many other colleagues, including Moritz Hardt, Ilya Mironov, Sasho Nikolov, Kobbi Nissim, Mallesh Pai, Benjamin Pierce, Adam Smith, Abhradeep Thakurta, Abhishek Bhowmick, Kunal Talwar, and Li Zhang. We are grateful to Madhu Sudan for proposing this monograph.

References

[1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[2] M.-F. Balcan, A. Blum, J. D. Hartline, and Y. Mansour. Mechanism design via machine learning. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 605–614. IEEE, 2005.

[3] A. Beimel, S. P. Kasiviswanathan, and K. Nissim. Bounds on the sample complexity for private learning and private data release. In *Theory of Cryptography*, pages 437–454. Springer, 2010.

[4] A. Beimel, K. Nissim, and U. Stemmer. Characterizing the sample complexity of private learners. In *Proceedings of the Conference on Innovations in Theoretical Computer Science*, pages 97–110. Association for Computing Machinery, 2013.

[5] A. Bhaskara, D. Dadush, R. Krishnaswamy, and K. Talwar. Unconditional differentially private mechanisms for linear queries. In H. J. Karloff and T. Pitassi, editors, *Proceedings of the Symposium on Theory of Computing Conference, Symposium on Theory of Computing, New York, NY, USA, May 19–22, 2012*, pages 1269–1284. 2012.

[6] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In Chen Li, editor, *Principles of Database Systems*, pages 128–138. ACM, 2005.

[7] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *Principles of Database Systems*. 2005.

[8] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In Cynthia Dwork, editor, *Symposium on Theory of Computing*, pages 609–618. Association for Computing Machinery, 2008.

[9] A. Blum and Y. Monsour. Learning, regret minimization, and equilibria, 2007.

[10] J. L. Casti. *Five Golden Rules: Great Theories of 20th-Century Mathematics and Why They Matter*. Wiley, 1996.

[11] T. H. Hubert Chan, E. Shi, and D. Song. Private and continual release of statistics. In *Automata, Languages and Programming*, pages 405–417. Springer, 2010.

[12] K. Chaudhuri and D. Hsu. Sample complexity bounds for differentially private learning. In *Proceedings of the Annual Conference on Learning Theory (COLT 2011)*. 2011.

[13] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of machine learning research: JMLR*, 12:1069, 2011.

[14] K. Chaudhuri, A. Sarwate, and K. Sinha. Near-optimal differentially private principal components. In *Advances in Neural Information Processing Systems 25*, pages 998–1006. 2012.

[15] Y. Chen, S. Chong, I. A. Kash, T. Moran, and S. P. Vadhan. Truthful mechanisms for agents that value privacy. *Association for Computing Machinery Conference on Electronic Commerce*, 2013.

[16] P. Dandekar, N. Fawaz, and S. Ioannidis. Privacy auctions for recommender systems. In *Internet and Network Economics*, pages 309–322. Springer, 2012.

[17] A. De. Lower bounds in differential privacy. In *Theory of Cryptography Conference*, pages 321–338. 2012.

[18] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the Association for Computing Machinery SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 202–210. 2003.

[19] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. *arXiv preprint arXiv:1302.3203*, 2013.

[20] C. Dwork. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)(2)*, pages 1–12. 2006.

[21] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503. 2006.

[22] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proceedings of the 2009 International Association for Computing Machinery Symposium on Theory of Computing (STOC)*. 2009.

[23] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference '06*, pages 265–284. 2006.

[24] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of lp decoding. In *Proceedings of the Association for Computing Machinery Symposium on Theory of Computing*, pages 85–94. 2007.

[25] C. Dwork and M. Naor. On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, 2010.

[26] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Association for Computing Machinery Symposium on Theory of Computing*, pages 715–724. Association for Computing Machinery, 2010.

[27] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and Sergey Yekhanin. Pan-private streaming algorithms. In *Proceedings of International Conference on Super Computing*. 2010.

[28] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. P. Vadhan. On the complexity of differentially private data release: Efficient algorithms and hardness results. In *Symposium on Theory of Computing '09*, pages 381–390. 2009.

[29] C. Dwork, M. Naor, and S. Vadhan. The privacy of the analyst and the power of the state. In *Foundations of Computer Science*. 2012.

[30] C. Dwork, A. Nikolov, and K. Talwar. Efficient algorithms for privately releasing marginals via convex relaxations. In *Proceedings of the Annual Symposium on Computational Geometry (SoCG)*. 2014.

[31] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Proceedings of Cryptology 2004*, vol. 3152, pages 528–544. 2004.

[32] C. Dwork, G. N. Rothblum, and S. P. Vadhan. Boosting and differential privacy. In *Foundations of Computer Science*, pages 51–60. 2010.

[33] C. Dwork, K. Talwar, A. Thakurta, and L. Zhang. Analyze gauss: Optimal bounds for privacy-preserving pca. In *Symposium on Theory of Computing*. 2014.

[34] L. Fleischer and Y.-H. Lyu. Approximately optimal auctions for selling privacy when costs are correlated with data. In *Association for Computing Machinery Conference on Electronic Commerce*, pages 568–585. 2012.

[35] A. Ghosh and K. Ligett. Privacy and coordination: Computing on databases with endogenous participation. In *Proceedings of the fourteenth ACM conference on Electronic commerce (EC)*, pages 543–560, 2013.

[36] A. Ghosh and A. Roth. Selling privacy at auction. In *Association for Computing Machinery Conference on Electronic Commerce*, pages 199–208. 2011.

[37] A. Groce, J. Katz, and A. Yerukhimovich. Limits of computational differential privacy in the client/server setting. In *Proceedings of the Theory of Cryptography Conference*. 2011.

[38] A. Gupta, M. Hardt, A. Roth, and J. Ullman. Privately releasing conjunctions and the statistical query barrier. In *Symposium on Theory of Computing '11*, pages 803–812. 2011.

[39] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *Theory of Cryptography Conference*, pages 339–356. 2012.

[40] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28, 1999.

[41] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems 25*, pages 2348–2356. 2012.

[42] M. Hardt and A. Roth. Beating randomized response on incoherent matrices. In *Proceedings of the Symposium on Theory of Computing*, pages 1255–1268. Association for Computing Machinery, 2012.

[43] M. Hardt and A. Roth. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the Symposium on Theory of Computing*. 2013.

[44] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Foundations of Computer Science*, pages 61–70. IEEE Computer Society, 2010.

[45] M. Hardt and K. Talwar. On the geometry of differential privacy. In *Proceedings of the Association for Computing Machinery Symposium on Theory of Computing*, pages 705–714. Association for Computing Machinery, 2010.

[46] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. Pearson, D. Stephan, S. Nelson, and D. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4, 2008.

[47] J. Hsu, Z. Huang, A. Roth, T. Roughgarden, and Z. S. Wu. Private matchings and allocations. arXiv preprint arXiv:1311.2828, 2013.

[48] J. Hsu, A. Roth, and J. Ullman. Differential privacy for the analyst via private equilibrium computation. In *Proceedings of the Association for Computing Machinery Symposium on Theory of Computing (STOC)*, pages 341–350, 2013.

[49] Z. Huang and S. Kannan. The exponential mechanism for social welfare: Private, truthful, and nearly optimal. In *IEEE Annual Symposium on the Foundations of Computer Science (FOCS)*, pages 140–149. 2012.

[50] P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. *Journal of Machine Learning Research — Proceedings Track*, 23:24.1–24.34, 2012.

[51] M. Kapralov and K. Talwar. On differentially private low rank approximation. In Sanjeev Khanna, editor, *Symposium on Discrete Algorthims*, pages 1395–1414. SIAM, 2013.

[52] S. P. Kasiviswanathan, H. K. Lee, Kobbi Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[53] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the Association for Computing Machinery (JAssociation for Computing Machinery)*, 45(6):983–1006, 1998.

[54] M. Kearns, M. Pai, A. Roth, and J. Ullman. Mechanism design in large games: Incentives and privacy. In *Proceedings of the 5th conference on Innovations in theoretical computer science (ITCS)*, 2014.

[55] D. Kifer, A. Smith, and A. Thakurta. Private convex empirical risk minimization and high-dimensional regression. *Journal of Machine Learning Research*, 1:41, 2012.

[56] K. Ligett and A. Roth. Take it or leave it: Running a survey when privacy comes at a cost. In *Internet and Network Economics*, pages 378–391. Springer, 2012.

[57] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Annual Symposium on Foundations of Computer Science, 1989*, pages 256–261. IEEE, 1989.

[58] A. McGregor, I. Mironov, T. Pitassi, O. Reingold, K. Talwar, and S. P. Vadhan. The limits of two-party differential privacy. In *Foundations of Computer Science*, pages 81–90. IEEE Computer Society, 2010.

[59] F. McSherry. Privacy integrated queries (codebase). Available on Microsoft Research downloads website. See also the Proceedings of SIGMOD 2009.

[60] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science*, pages 94–103. 2007.

[61] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Foundations of Computer Science*, pages 94–103. 2007.

[62] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the Association for Computing Machinery SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 37–48. Association for Computing Machinery, 2011.

[63] I. Mironov. On significance of the least significant bits for differential privacy. In T. Yu, G. Danezis, and V. D. Gligor, editors, *Association for Computing Machinery Conference on Computer and Communications Security*, pages 650–661. Association for Computing Machinery, 2012.

[64] I. Mironov, O. Pandey, O. Reingold, and S. P. Vadhan. Computational differential privacy. In *Proceedings of CRYPTOLOGY*, pages 126–142. 2009.

[65] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets (how to break anonymity of the netflix prize dataset). In *Proceedings of IEEE Symposium on Security and Privacy*. 2008.

[66] A. Nikolov, K. Talwar, and L. Zhang. The geometry of differential privacy: the sparse and approximate cases. *Symposium on Theory of Computing*, 2013.

[67] K. Nissim, C. Orlandi, and R. Smorodinsky. Privacy-aware mechanism design. In *Association for Computing Machinery Conference on Electronic Commerce*, pages 774–789. 2012.

[68] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Association for Computing Machinery Symposium on Theory of Computing*, pages 75–84. 2007.

[69] K. Nissim, R. Smorodinsky, and M. Tennenholtz. Approximately optimal mechanism design via differential privacy. In *Innovations in Theoretical Computer Science*, pages 203–213. 2012.

[70] M. Pai and A. Roth. Privacy and mechanism design. *SIGecom Exchanges*, 2013.

[71] R. Rogers and A. Roth. Asymptotically truthful equilibrium selection in large congestion games. arXiv preprint arXiv:1311.2625, 2013.

[72] A. Roth. Differential privacy and the fat-shattering dimension of linear queries. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques*, pages 683–695. Springer, 2010.

[73] A. Roth. Buying private data at auction: the sensitive surveyor's problem. *Association for Computing Machinery SIGecom Exchanges*, 11(1):1–8, 2012.

[74] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Symposium on Theory of Computing '10*, pages 765–774. 2010.

[75] A. Roth and G. Schoenebeck. Conducting truthful surveys, cheaply. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 826–843. 2012.

[76] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. arXiv preprint arXiv:0911.5708, 2009.

[77] R. Schapire. The boosting approach to machine learning: An overview. In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

[78] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 39:297–336, 1999.

[79] R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012.

[80] A. Smith and A. G. Thakurta. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Proceedings of Conference on Learning Theory*. 2013.

[81] L. Sweeney. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicines Ethics*, 25:98–110, 1997.

[82] J. Ullman. Answering $n^{\{2+o(1)\}}$ counting queries with differential privacy is hard. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *Symposium on Theory of Computing*, pages 361–370. Association for Computing Machinery, 2013.

[83] L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

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

[85] D. Xiao. Is privacy compatible with truthfulness? In *Proceedings of the Conference on Innovations in Theoretical Computer Science*, pages 67–86. 2013.