7

When Worst-Case Sensitivity is Atypical

In this section, we briefly describe two general techniques, both enjoying unconditional privacy guarantees, that can often make life easier for the data analyst, especially when dealing with a function that has arbitrary, or difficult to analyze, worst-case sensitivity. These algorithms are most useful in computing functions that, for some exogenous reason, the analyst has reason to believe are "usually" insensitive in practice.

7.1 Subsample and aggregate

The Subsample and Aggregate technique yields a method for "forcing" the computation of a function $f(x)$ to be insensitive, even for an *arbitrary* function f. Proving privacy will be trivial. Accuracy depends on properties of the function f and the specific data set x; in particular, if $f(x)$ can be accurately estimated, with high probability, on $f(S)$, where S is a random subset of the elements in x, then accuracy should be good. Many maximum likelihood statistical estimators enjoy this property on "typical" data sets — this is why these estimators are employed in practice.

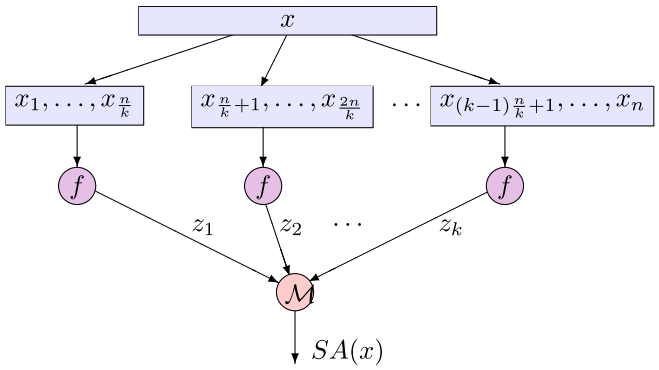

Figure 7.1: Subsample and Aggregate with a generic differentially private aggregation algorithm $\mathcal{M}$.

In Subsample and Aggregate, the n rows of the database x are randomly partitioned into m blocks $B_1, \ldots, B_m$, each of size n/m. The function f is computed *exactly, without noise*, independently on each block. The intermediate outcomes $f(B_1), \ldots, f(B_m)$ are then combined via a differentially private aggregation mechanism — typical examples include standard aggregations, such as the α-trimmed mean,[1] the Winsorized mean,[2] and the median, but there are no restrictions — and then adding Laplace noise scaled to the sensitivity of the aggregation function in question; see Figure 7.1.

The key observation in Subsample and Aggregate is that any single element can affect at most one block, and therefore the value of just a single $f(B_i)$. Thus, changing the data of any individual can change at most a single input to the aggregation function. Even if f is arbitrary, the analyst chooses the aggregation function, and so is free to choose one that is insensitive, *provided that choice is independent of the database!* Privacy is therefore immediate: For any $\delta \geq 0$ and any function f, if the aggregation mechanism $\mathcal{M}$ is (ε, δ)-differentially private

[1]The α-trimmed mean is the mean after the top and bottom α fraction of the inputs have been discarded.

[2]The Winsorized mean is similar to the α-trimmed mean except that, rather than being discarded, the top and bottom α fraction are replaced with the most extreme remaining values.

then so is the Subsample and Aggregate technique when instantiated with f and $\mathcal{M}$.[3]

Utility is a different story, and it is frustratingly difficult to argue even for the case in which data are plentiful and large random subsets are very likely to give similar results. For example, the data may be labeled training points in high dimensional space and the function is logistic regression, which produces a vector v and labels a point p with $+1$ if and only if $p \cdot v \geq T$ for some (say, fixed) threshold T. Intuitively, if the samples are sufficiently plentiful and typical then all blocks should yield similar vectors v. The difficulty comes in getting a good bound on the worst-case sensitivity of the aggregation function — we may need to use the size of the range as a fallback. Nonetheless, some nice applications are known, especially in the realm of statistical estimators, where, for example, it can be shown that, under the assumption of "generic normality," privacy can be achieved at *no* additional cost in statistical efficiency (roughly, accuracy as the number of samples grows). We do not define generic normality here, but note that estimators fitting these assumptions include the maximum likelihood estimator for "nice" parametric families of distributions such as gaussians, and maximum-likelihood estimators for linear regression and logistic regression.

Suppose the function f has a *discrete* range of cardinality m, say, $[m]$. In this case Subsample and Aggregate will need to aggregate a set of b elements drawn from $[m]$, and we can use Report Noisy Arg-Max to find the most popular outcome. This approach to aggregation requires $b \geq \log m$ to obtain meaningful results even when the intermediate outcomes are unanimous. We will see an alternative below with no such requirement.

Example 7.1 (Choosing a Model)**.** Much work in statistics and machine learning addresses the problem of *model selection*: Given a data set and a discrete collection of "models," each of which is a family of probability distributions, the goal is to determine the model that best "fits"

[3]The choice of aggregation function can even depend on the database, but the selection must be made in a differentially private fashion. The privacy cost is then the cost of composing the choice operation with the aggregation function.

the data. For example, given a set of labeled d-dimensional data, the collection of models might be all subsets of at most $s \ll d$ features, and the goal is to find the set of features that best permits prediction of the labels. The function f might be choosing the best model from the given set of of m models, a process known as *model fitting*, via an arbitrary learning algorithm. Aggregation to find the most popular value could be done via Report Noisy Max, which also yields an estimate of its popularity.

Example 7.2 (Significant Features)**.** This is a special case of model fitting. The data are a collection of points in $\mathbb{R}^d$ and the function is the very popular LASSO, which yields as output a list $L \in [d]^s$ of at most $s \ll d$ significant features. We can aggregate the output in two ways: feature by feature — equivalent to running d executions of Subsample and Aggregate, one for each feature, each with a range of size 2 — or on the set as a whole, in which case the cardinality of the range is $\binom{d}{s}$.

7.2 Propose-test-Release

At this point one might ask: what is the meaning of the aggregation if there is not substantial agreement among the blocks? More generally, for any reasonably large-scale statistical analysis in real life, we expect the results to be fairly stable, independent of the presence or absence of any single individual. Indeed, this is the entire intuition behind the significance of a statistic and underlying the utility of differential privacy. We can even go further, and argue that if a statistic is not stable, we should have no interest in computing it. Often, our database will in fact be a sample from a larger population, and our true goal is not to compute the value of the statistic on the database itself, but rather estimate it for the underlying population. Implicitly, therefore, when computing a statistic we are already assuming that the statistic is stable under subsampling!

Everything we have seen so far has provided privacy even on very "idiosyncratic" datasets, for which "typically" stable algorithms my be highly unstable. In this section we introduce a methodology, Propose-Test-Release, which is motivated by the philosophy that if there is

insufficient stability then the analysis can be abandoned because the results are not in fact meaningful. That is, the methodology allows the analyst to check that, *on the given dataset*, the function satisfies some "robustness" or "stability" criterion and, if it does not, to halt the analysis.

The goal of our first application of Propose-Test-Release is to come up with a variant of the Laplace mechanism that adds noise scaled to something strictly smaller than the sensitivity of a function. This leads to the notion of *local sensitivity*, which is defined for a (function, database) pair, say, (f, x). Quite simply, the local sensitivity of f with respect to x is the amount by which the $f(y)$ can differ from $f(x)$ for any y adjacent to x.

Definition 7.1 (Local Sensitivity)**.** The local sensitivity of a function $f : \mathcal{X}^n \to \mathbb{R}^k$ with respect to a database x is:

$$\max_{y \text{ adjacent to } x} \|f(x) - f(y)\|_1.$$

The Propose-Test-Release approach is to first *propose* a bound, say b, on local sensitivity — typically the data analyst has some idea of what this should be — and then run a differentially private *test* to ensure that the database is "far" from any database for which this bound fails to hold. If the test is passed, then the sensitivity is assumed to be bounded by b, and a differentially private mechanism such as, for example, the Laplace mechanism with parameter b/ϵ, is used to *release* the (slightly) noisy response to the query.

Note that we can view this approach as a two-party algorithm where one party plays an honest data analyst and the other is the Laplace mechanism. There is an interplay between the honest analyst and the mechanism in which the algorithm asks for an estimate of the sensitivity and then "instructs" the mechanism to use this estimated sensitivity in responding to subsequent queries. Why does it need to be so complicated? Why can't the mechanism simply add noise scaled to the local sensitivity without playing this private estimation game? The reason is that the local sensitivity may *itself* be sensitive. This fact, combined with some auxiliary information about the database, can lead to privacy problems: the adversary may know that the database is one of x,

which has very low local sensitivity for the computation in question, and a neighboring y, for which the function has very high local sensitivity. In this case the adversary may be able to guess rather accurately which of x and y is the true database. For example, if $f(x) = f(y) = s$ and the reponse is far from s, then the adversary would guess y.

This is captured by the math of differential privacy. There are neighboring instances of the median function which have the same median, say, m, but arbitrarily large gaps in the local sensitivity. Suppose the response R to the median query is computed via the Laplace mechanism with noise scaled to the local sensitivity. When the database is x the probability mass is close to m, because the sensitivity is small, but when the database is y the mass is far flung, because the sensitivity is large. As an extreme case, suppose the local sensitivity on x is exactly zero, for example, $\mathcal{X} = \{0, 10^6\}$, n is even, and x, which has size $n + 1$, contains $1 + n/2$ zeros. Then the median of x is zero and the local sensitivity of the median, when the database is x, is 0. In contrast, the neighboring database y has size n, contains $n/2$ zeros, has median zero (we have defined median to break ties in favor of the smaller value), and the local sensitivity of the median, when the database is y, is 10^6. On x all the mass of the Laplace mechanism (with parameter $0/\varepsilon = 0$) is concentrated on the single point 0; but on y the probability distribution has standard deviation $\sqrt{2} \cdot 10^6$. This destroys all hope of differential privacy.

To test that the database is "far" from one with local sensitivity greater than the proposed bound b, we may pose the query: "What is the distance of the true database to the closest one with local sensitivity exceeding b?" Distance to a fixed set of databases is a (global) sensitivity 1 query, so this test can be run in a differentially private fashion by adding noise $\mathrm{Lap}(1/\varepsilon)$ to the true answer. To err on the side of privacy, the algorithm can compare this noisy distance to a conservative threshold — one that is only negligibly likely to be exceeded due to a freak event of very large magnitude Laplace noise. For example, if the threshold used is, say, $\ln^2 n$, the probability of a false positive (passing the test when the local sensitivity in fact exceeds b) is at most $O(n^{-\varepsilon \ln n})$, by the properties of the Laplace distribution. Because of the negligible probability of a false positive, the technique cannot yield $(\varepsilon, 0)$-differential privacy for any ε.

To apply this methodology to consensus on blocks, as in our discussion of Subsample and Aggregate, view the intermediate results $f(B_1), \ldots, f(B_m)$ as a data set and consider some measure of the concentration of these values. Intuitively, if the values are tightly concentrated then we have consensus among the blocks. Of course, we still need to find the correct notion of concentration, one that is meaningful and that has a differentially private instantiation. In a later section we will define and weave together two notions of stability that seem relevant to Subsample and Aggregate: insensitivity (to the removal or addition of a few data points) and stability under subsampling, capturing the notion that a subsample should yield similar results to the full data set.

7.2.1 Example: the scale of a dataset

Given a dataset, a natural question to ask is, "What is the scale, or dispersion, of the dataset?" This is a different question from *data location*, which might be captured by the median or the mean. The data scale is more often captured by the variance or an interquantile range. We will focus on the *interquartile range (IQR)*, a well-known robust estimator for the scale of the data. We begin with some rough intuition. Suppose the data are *i.i.d.* samples drawn from a distribution with cumulative distribution function F. Then $\mathrm{IQR}(F)$, defined as $F^{-1}(3/4) - F^{-1}(1/4)$, is a constant, depending only on F. It might be very large, or very tiny, but either way, if the density of F is sufficiently high at the two quartiles, then, given enough samples from F, the empirical (that is, sample) interquartile distance should be close to $\mathrm{IQR}(F)$.

Our Propose-Test-Release algorithm for the interquartile distance first tests how many database points need to be changed to obtain a data set with a "sufficiently different" interquartile distance. Only if the (noisy) reply is "sufficiently large" will the algorithm release an approximation to the interquartile range of the dataset.

The definition of "sufficiently different" is multiplicative, as an additive notion for difference of scale makes no sense — what would be the right scale for the additive amount? The algorithm therefore works with the logarithm of the scale, which leads to a multiplicative noise

on the IQR. To see this, suppose that, as in what might be the typical case, the sample interquartile distance cannot change by a factor of 2 by modifying a single point. Then the logarithm (base 2) of the sample interquartile has local sensitivity bounded by 1. This lets us privately release an approximation to *the logarithm of* the sample interquartile range by adding to this value a random draw from $\mathrm{Lap}(1/\varepsilon)$.

Let $\mathrm{IQR}(x)$ denote the sample interquartile range when the the data set is x. The algorithm is (implicitly) *proposing* to add noise drawn from $\mathrm{Lap}(1/\varepsilon)$ to the value $\log_2(\mathrm{IQR}(x))$. To *test* whether this magnitude of noise is sufficient for differential privacy, we discretize $\mathbb{R}$ into disjoint bins $\{[k \ln 2, (k+1) \ln 2)\}_{k \in \mathbf{Z}}$ and ask how many data points must be modified in order to obtain a new database, the logarithm (base 2) of whose interquartile range is in a different bin than that of $\log_2(\mathrm{IQR}(x))$. If the answer is at least two then the local sensitivity (of the logarithm of the interquartile range) is bounded by the bin width. We now give more details.

To understand the choice of bin size, we write

$$\log_2(\mathrm{IQR}(x)) = \frac{\ln \mathrm{IQR}(x)}{\ln 2} = \frac{c \ln 2}{\ln 2},$$

whence we find that looking at $\ln(\mathrm{IQR}(x))$ on the scale of $\ln 2$ is equivalent to looking at $\log_2(\mathrm{IQR}(x))$ on the scale of 1. Thus we have scaled bins which are intervals whose endpoints are a pair of adjacent integers: $B_k = [k, k+1)$, $k \in \mathbf{Z}$, and we let $k_1 = \lfloor \log_2(\mathrm{IQR}(x)) \rfloor$, so $\log_2(\mathrm{IQR}(x)) \in [k_1, k_1 + 1)$ and we say informally that the logarithm of the IQR is in bin k_1. Consider the following testing query:

> **$\mathbf{Q_0}$** : How many data points need to change in order to get a new database z such that $\log_2(\mathrm{IQR}(z)) \notin B_{k_1}$?

Let $A_0(x)$ be the true answer to $\mathbf{Q_0}$ when the database is x. If $A_0(x) \geq 2$, then neighbors y of x satisfy $|\log_2(\mathrm{IQR}(y)) - \log_2(\mathrm{IQR}(x))| \leq 1$. That is, they are close to each other. This is not equivalent to being in the same interval in the discretization: $\log_2(\mathrm{IQR}(x))$ may lie close to one of the endpoints of the interval $[k_1, k_1 + 1)$ and $\log_2(\mathrm{IQR}(y))$ may lie just on the other side of the endpoint. Letting $R_0 = A_0(x) + \mathrm{Lap}(1/\varepsilon)$, a small R_0, even when the

draw from the Laplace distribution has small magnitude, might not actually indicate high sensitivity of the interquartile range. To cope with the case that the local sensitivity is very small, but $\log_2(\mathrm{IQR}(x))$ is very close to the boundary, we consider a second discretization $\{B_k^{(2)} = [k-0.5, k+0.5)\}_{k\in\mathbf{Z}}$. We denote the two discretizations by $B^{(1)}$ and $B^{(2)}$ respectively. The value $\log_2(\mathrm{IQR}(x))$ — indeed, any value — cannot be close to a boundary in both discretizations. The test is passed if R_0 is large in at least one discretization.

The **Scale** algorithm (Algorithm 12) below for computing database scale assumes that n, the size of the database, is known, and the distance query ("How far to a database whose interquartile range has sensitivity exceeding b?") is asking how many points must be *moved* to reach a database with high sensitivity of the IQR. We can avoid this assumption by having the algorithm first ask the (sensitivity 1) query: "How many data points are in x?" We remark that, for technical reasons, to cope with the case $IQR(x) = 0$, we define $\log 0 = -\infty$, $\lfloor -\infty \rfloor = -\infty$, and let $[-\infty, -\infty) = \{-\infty\}$.

Algorithm 12 The **Scale** Algorithm (releasing the interquartile range)

Require: dataset: $x \in \mathcal{X}^*$, privacy parameters: $\epsilon, \delta > 0$

for the jth discretization ($j = 1, 2$) **do**
  Compute $R_0(x) = A_0(x) + z_0$, where $z_0 \in_R \mathrm{Lap}(1/\varepsilon)$.
  if $R_0 \leq 1 + \ln(1/\delta)$ **then**
    Let $s^{(j)} = \perp$.
  else
    Let $s^{(j)} = IQR(x) \times 2^{z_s^{(j)}}$, where $z_s^{(j)} \sim \mathrm{Lap}(1/\varepsilon)$.
  end if
end for
if $s^{(1)} \neq \perp$ **then**
  Return $s^{(1)}$.
else
  Return $s^{(2)}$.
end if

Note that the algorithm is efficient: let $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ denote the n database points *after sorting*, and let $x(m)$ denote the median, so $m = \lfloor (n+1)/2 \rfloor$. Then the local sensitivity of the median is $\max\{x(m) - x(m-1), x(m+1) - x(m)\}$ and, more importantly, one can compute $A_0(x)$ by considering $O(n)$ sliding intervals with width 2^{k_1} and 2^{k_1+1}, each having one endpoint in x. The computational cost for each interval is constant.

We will not prove convergence bounds for this algorithm because, for the sake of simplicity, we have used a base for the logarithm that is far from optimal (a better base is $1 + 1/\ln n$). We briefly outline the steps in the proof of privacy.

Theorem 7.1. Algorithm **Scale** (Algorithm 12) is $(4\varepsilon, \delta)$-differentially private.

Proof. (Sketch.) Letting s be shorthand for the result obtained with a single discretization, and defining $\mathcal{D}_0 = \{x : A_0(x) \geq 2\}$, the proof shows:

1. The worst-case sensitivity of query $\mathbf{Q_0}$ is at most 1.
2. Neighboring databases are almost equally likely to result in $\perp$: For all neighboring database x, y:
$$\Pr[s = \perp | x] \leq e^{\varepsilon} \Pr[s = \perp | y].$$
3. Databases not in $\mathcal{D}_0$ are unlikely to pass the test:
$$\forall x \notin \mathcal{D}_0 : \Pr[s \neq \perp | x] \leq \frac{\delta}{2}.$$
4. $\forall C \in \mathbb{R}^+, x \in \mathcal{D}_0$ and all neighbors y of x:
$$\Pr[s \in C | x] \leq e^{2\varepsilon} \Pr[s \in C | y].$$

Thus, we get $(2\varepsilon, \delta/2)$-differential privacy for each discretization. Applying Theorem 3.16 (Appendix B), which says that "the epsilons and the deltas add up," yields $(4\varepsilon, \delta)$-differential privacy. □

7.3 Stability and privacy

7.3.1 Two notions of stability

We begin by making a distinction between the two notions of stability intertwined in this section: stability under subsampling, which yields similar results under random subsamples of the data, and perturbation stability, or low local sensitivity, for a given dataset. In this section we will define and make use of extreme versions of both of these.

- *Subsampling stability*: We say f is q-subsampling stable on x if $f(\hat{x}) = f(x)$ with probability at least $3/4$ when $\hat{x}$ is a random subsample from x which includes each entry independently with probability q. We will use this notion in Algorithm $\mathcal{A}_{\text{samp}}$, a variant of Sample and Aggregate.

- *Perturbation Stability*: We say that f is *stable* on x if f takes the value $f(x)$ on all of the neighbors of x (and *unstable* otherwise). In other words, f is stable on x if the local sensitivity of f on x is zero. We will use this notion (implemented in Algorithm $\mathcal{A}_{\text{dist}}$ below) for the aggregation step of $\mathcal{A}_{\text{samp}}$.

At the heart of Algorithm $\mathcal{A}_{\text{samp}}$ is a relaxed version of perturbation stability, where instead of requiring that the value be unchanged on neighboring databases — a notion that makes sense for arbitrary ranges, including arbitrary discrete ranges — we required only that the value be "close" on neighboring databases — a notion that requires a metric on the range.

Functions f with arbitrary ranges, and in particular the problem of aggregating outputs in Subsample and Aggregate, motivate the next algorithm, $\mathcal{A}_{\text{dist}}$. On input f, x, $\mathcal{A}_{\text{dist}}$ outputs $f(x)$ with high probability if x is at distance at least $\frac{2\log(1/\delta)}{\varepsilon}$ from the nearest *un*stable data set. The algorithm is conceptually trivial: compute the distance to the nearest unstable data set, add Laplace noise $\text{Lap}(1/\varepsilon)$, and check that this noisy distance is at least $\frac{2\log(1/\delta)}{\varepsilon}$. If so, release $f(x)$, otherwise output $\perp$. We now make this a little more formal.

We begin by defining a quantitative measure of perturbation stability.

Definition 7.2. A function $f : \mathcal{X}^* \to \mathcal{R}$ is *k-stable on input x* if adding or removing any k elements from x does not change the value of f, that is, $f(x) = f(y)$ for all y such that $|x \triangle y| \leq k$. We say f is *stable* on x if it is (at least) 1-stable on x, and *unstable* otherwise.

Definition 7.3. The *distance to instability* of a data set $x \in \mathcal{X}^*$ with respect to a function f is the number of elements that must be added to or removed from y to reach a data set that is not stable under f.

Note that f is k-stable on x if and only if the distance of x to instability is at least k.

Algorithm $\mathcal{A}_{\text{dist}}$, an instantiation of Propose-Test-Release for discrete-valued functions g, appears in Figure 13.

Algorithm 13 $\mathcal{A}_{\text{dist}}$ (releasing $g(x)$ based on distance to instability)

Require: dataset: $x \in \mathcal{X}^*$, privacy parameters: $\epsilon, \delta > 0$, function $g : \mathcal{X}^* \to \mathbb{R}$

$d \leftarrow$ distance from x to nearest unstable instance
$\hat{d} \leftarrow d + \text{Lap}(1/\varepsilon)$
if $\hat{d} > \frac{\log(1/\delta)}{\varepsilon}$ **then**
  Output $g(x)$
else
  Output $\perp$
end if

The proof of the following propostion is immediate from the properties of the Laplace distribution.

Proposition 7.2. For every function g:

1. $\mathcal{A}_{\text{dist}}$ is (ε, δ)-differentially private.
2. For all $\beta > 0$: if g is $\frac{\ln(1/\delta)+\ln(1/\beta)}{\varepsilon}$-stable on x, then $\mathcal{A}_{\text{dist}}(x) = g(x)$ with probability at least $1 - \beta$, where the probability space is the coin flips of $\mathcal{A}_{\text{dist}}$.

This distance-based result is the best possible, in the following sense: if there are two data sets x and y for which $\mathcal{A}_{\text{dist}}$ outputs different

values $g(x)$ and $g(y)$, respectively, with at least constant probability, then the distance from x to y must be $\Omega(\log(1/\delta)/\varepsilon)$.

Distance to instability can be difficult to compute, or even to lower bound, so this is not in general a practical solution. Two examples where distance to instability turns out to be easy to bound are the median and the mode (most frequently occurring value).

$\mathcal{A}_{\text{dist}}$ may also be unsatisfactory if the function, say f, is not stable on the specific datasets of interest. For example, suppose f is not stable because of the presence of a few outliers in x. Instances of the average behave this way, although for this function there are well know robust alternatives such as the winsorized mean, the trimmed mean, and the median. By what about for general functions f? Is there a method of "forcing" an arbitrary f to be stable on a database x?

This will be the goal of $\mathcal{A}_{\text{samp}}$, a variant of Subsample and Aggregate that outputs $f(x)$ with high probability (over its own random choices) whenever f is subsampling stable on x.

7.3.2 Algorithm $\mathcal{A}_{\text{samp}}$

In $\mathcal{A}_{\text{samp}}$, the blocks $B_1, \ldots, B_m$ are chosen *with replacement*, so that each block has the same distribution as the inputs (although now an element of x may appear in multiple blocks). We will call these subsampled datasets $\hat{x}_1, \ldots, \hat{x}_m$. The intermediate outputs $z = \{f(\hat{x}_1), \ldots, f(\hat{x}_m)\}$ are then aggregated via $\mathcal{A}_{\text{dist}}$ with function $g = \text{mode}$. The distance measure used to estimate the stability of the mode on z is a scaled version of the difference between the popularity of the mode and that of the second most frequent value. Algorithm $\mathcal{A}_{\text{samp}}$, appears in Figure 14. Its running time is dominated by running f about $1/q^2$ times; hence it is efficient whenever f is.

The key property of Algorithm $\mathcal{A}_{\text{samp}}$ is that, on input f, x, it outputs $f(x)$ with high probability, over its own random choices, whenever f is q-subsampling stable on x for $q = \frac{\varepsilon}{64\log(1/\delta)}$. This result has an important statistical interpretation. Recall the discussion of model selection from Example 7.1. Given a collection of models, the *sample complexity* of model selection is the number of samples from a distribution in one of the models necessary to select the correct model

with probability at least 2/3. The result says that *differentially private* model selection increases the sample complexity of (non-private) model selection by a problem-independent (and range-independent) factor of $O(\log(1/\delta)/\varepsilon)$.

Algorithm 14 $\mathcal{A}_{\text{samp}}$: Bootstrapping for Subsampling-Stable f

Require: dataset: x, function $f : \mathcal{X}^* \to \mathbb{R}$, privacy parameters $\epsilon, \delta > 0$.

$q \leftarrow \frac{\epsilon}{64\ln(1/\delta)}$, $m \leftarrow \frac{\log(n/\delta)}{q^2}$.
Subsample m data sets $\hat{x}_1, ..., \hat{x}_m$ from x, where $\hat{x}_i$ includes each position of x independently with probability q.
if some element of x appears in more than $2mq$ sets $\hat{x}_i$ **then**
  Halt and output $\perp$.
else
  $z \leftarrow \{f(\hat{x}_1), \cdots, f(\hat{x}_m)\}$.
  For each $r \in \mathbb{R}$, let $count(r) = \#\{i : f(\hat{x}_i) = r\}$.
  Let $count_{(i)}$ denote the the ith largest count, $i = 1, 2$.
  $d \leftarrow (count_{(1)} - count_{(2)})/(4mq) - 1$
  Comment Now run $\mathcal{A}_{\text{dist}}(g, z)$ using d to estimate distance to instability:
  $\hat{d} \leftarrow d + \text{Lap}(\frac{1}{\epsilon})$.
  if $\hat{d} > \ln(1/\delta)/\varepsilon$ **then**
    Output $g(z) = mode(z)$.
  else
    Output $\perp$.
  end if
end if

Theorem 7.3.

1. Algorithm $\mathcal{A}_{\text{samp}}$ is (ε, δ)-differentially private.
2. If f is q-subsampling stable on input x where $q = \frac{\varepsilon}{64\ln(1/\delta)}$, then algorithm $\mathcal{A}_{\text{samp}}(x)$ outputs $f(x)$ with probability at least $1 - 3\delta$.
3. If f can be computed in time $T(n)$ on inputs of length n, then $\mathcal{A}_{\text{samp}}$ runs in expected time $O(\frac{\log n}{q^2})(T(qn) + n)$.

Note that the utility statement here is an input-by-input guarantee; f need not be q-subsampling stable on all inputs. Importantly, there is no dependence on the size of the range $\mathcal{R}$. In the context of model selection, this means that one can efficiently satisfy differential privacy with a modest blowup in sample complexity (about $\log(1/\delta)/\varepsilon$) whenever there is a particular model that gets selected with reasonable probability.

The proof of privacy comes from the insensitivity of the computation of d, the privacy of the Propose-Test-Release technique, and the privacy of Subsample and Aggregate, modified slightly to allow for the fact that this algorithm performs sampling with replacement and thus the aggregator has higher sensitivity, since any individual might affect up to $2mq$ blocks. The main observation for analyzing the utility of this approach is that the stability of the *mode* is a function of the difference between the frequency of the mode and that of the next most popular element. The next lemma says that if f is subsampling stable on x, then x is far from unstable with respect to the mode $g(z) = g(f(\hat{x}_1), \ldots, f(\hat{x}_m))$ (but not necessarily with respect to f), and moreover one can estimate the distance to instability of x *efficiently* and privately.

Lemma 7.4. Fix $q \in (0,1)$. Given $f : \mathcal{X}^* \to \mathcal{R}$, let $\hat{f} : \mathcal{X}^* \to \mathcal{R}$ be the function $\hat{f} = \text{mode}(f(\hat{x}_1), ..., f(\hat{x}_m))$ where each $\hat{x}_i$ includes each element of x independently with probability q and $m = \ln(n/\delta)/q^2$. Let $d(z) = (\text{count}_{(1)} - \text{count}_{(2)})/(4mq) - 1$; that is, given a "database" z of values, $d(z)+1$ isa scaled difference between the number of occurrences of the two most popular values. Fix a data set x. Let E be the event that no position of x is included in more than $2mq$ of the subsets $\hat{x}_i$. Then, when $q \leq \varepsilon/64\ln(1/\delta)$ we have:

1. E occurs with probability at least $1 - \delta$.
2. Conditioned on E, d lower bounds the stability of $\hat{f}$ on x, and d has global sensitivity 1.
3. If f is q-subsampling stable on x, then with probability at least $1 - \delta$ over the choice of subsamples, we have $\hat{f}(x) = f(x)$, and, conditioned on this event, the final test will be passed with

probability at least $1 - \delta$, where the probability is over the draw from $\text{Lap}(1/\varepsilon)$.

The events in Parts 2 and 3 occur simultaenously with probability at least $1 - 2\delta$.

Proof. Part 1 follows from the Chernoff bound. To prove Part 2, notice that, conditioned on the event E, adding or removing one entry in the original data set changes any of the counts $count_{(r)}$ by at most $2mq$. Therefore, $count_{(1)} - count_{(2)}$ changes by at most $4mq$. This in turn means that $d(f(\hat{x}_1), \ldots, f(\hat{x}_m))$ changes by at most one for any x and hence has global sensitivity of one. This also implies that d lower bounds the stability of $\hat{f}$ on x.

We now turn to part 3. We want to argue two facts:

1. If f is q-subsampling stable on x, then there is likely to be a large gap between the counts of the two most popular bins. Specifically, we want to show that with high probability $count_{(1)} - count_{(2)} \geq m/4$. Note that if the most popular bin has count at least $5m/8$ then the second most popular bin can have count at most $3m/8$, with a difference of $m/4$. By definition of subsampling stability the most popular bin has an expected count of at least $3m/4$ and hence, by the Chernoff bound, taking $\alpha = 1/8$, has probability at most $e^{-2m\alpha^2} = e^{-m/32}$ of having a count less than $5m/8$. (All the probabilities are over the subsampling.)

2. When the gap between the counts of the two most popular bins is large, then the algorithm is unlikely to fail; that is, the test is likely to succeed. The worry is that the draw from $\text{Lap}(\frac{1}{\varepsilon})$ will be negative and have large absolute value, so that $\hat{d}$ falls below the threshold $(\ln(1/\delta)/\varepsilon)$ even when d is large. To ensure this happens with probability at most δ it suffices that $d > 2\ln(1/\delta)/\varepsilon$.
   By definition, $d = (count_{(1)} - count_{(2)})/(4mq) - 1$, and, assuming we are in the high probability case just described, this implies

$$d \geq \frac{m/4}{4mq} - 1 = \frac{1}{16q} - 1$$

so it is enough to have

$$\frac{1}{16q} > 2\ln(1/\delta)/\varepsilon.$$

Taking $q \leq \varepsilon/64\ln(1/\delta)$ suffices.

Finally, note that with these values of q and m we have $e^{-m/32} < \delta$. □

Example 7.3. [The Raw Data Problem] Suppose we have an analyst whom we can trust to follow instructions and only publish information obtained according to these instructions. Better yet, suppose we have b such analysts, and we can trust them not to communicate among themselves. The analysts do not need to be identical, but they do need to be considering a common set of *options*. For example, these options might be different statistics in a fixed set S of possible statistics, and in this first step the analyst's goal is to choose, for eventual publication, the most significant statistic in S. Later, the chosen statistic will be recomputed in a differentially private fashion, and the result can be published.

As described the procedure is not private at all: the *choice* of statistic made in the first step may depend on the data of a single individual! Nonetheless, we can use the Subsample-and-Aggregate framework to carry out the first step, with the ith analyst receiving a subsample of the data points and applying to this smaller database the function f_i to obtain an option. The options are then aggregated as in algorithm $\mathcal{A}_{\text{samp}}$; if there is a clear winner this is overwhelmingly likely to be the selected statistic. This was *chosen* in a differentially private manner, and in the second step it will be *computed* with differential privacy.

Bibliographic Notes

Subsample and Aggregate was invented by Nissim, Raskhodnikova, and Smith [68], who were the first to define and exploit low local sensitivity. Propose-Test-Release is due to Dwork and Lei [22], as is the algorithm for releasing the interquartile range. The discussion of stability and privacy, and Algorithm $\mathcal{A}_{samp}$ which blends these two techniques, is due to Smith and Thakurta [80]. This paper demonstrates the power of

$\mathcal{A}_{samp}$ by analyzing the subsampling stability conditions of the famous LASSO algorithm and showing that differential privacy can be obtained "for free," via (a generalization of $\mathcal{A}_{samp}$), precisely under the (fixed data as well as distributional) conditions for which LASSO is known to have good explanatory power.