# 6

## Boosting for Queries

In the previous sections, we have focused on the problem of private query release in which we insist on bounding the worst-case error over all queries. Would our problem be easier if we instead asked only for low error on average, given some distribution over the queries? In this section, we see that the answer is no: given a mechanism which is able to solve the query release problem with low average error given any distribution on queries, we can "boost" it into a mechanism which solves the query release problem to worst-case error. This both sheds light on the difficulty of private query release, and gives us a new tool for designing private query release algorithms.

Boosting is a general and widely used method for improving the accuracy of learning algorithms. Given a set of labeled training examples

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\},$$

where each $x_i$ is drawn from an underlying distribution $\mathcal{D}$ on a universe $\mathcal{U}$, and each $y_i \in \{+1, -1\}$, a learning algorithm produces a hypothesis $h : \mathcal{U} \to \{+1, -1\}$. Ideally, $h$ will not just "describe" the labeling on the given samples, but will also *generalize*, providing a reasonably accurate method of classifying other elements drawn from the underlying

distribution. The goal of boosting is to convert a weak *base* learner, which produces a hypothesis that may do just a little better than random guessing, into a strong learner, which yields a very accurate predictor for samples drawn according to $\mathcal{D}$. Many boosting algorithms share the following basic structure. First, an initial (typically uniform) probability distribution is imposed on the sample set. Computation then proceeds in rounds. In each round $t$:

1. The base learner is run on the current distribution, denoted $\mathcal{D}_t$, producing a classification hypothesis $h_t$; and
2. The hypotheses $h_1, \ldots, h_t$ are used to re-weight the samples, defining a new distribution $\mathcal{D}_{t+1}$.

The process halts either after a predetermined number of rounds or when an appropriate combining of the hypotheses is determined to be sufficiently accurate. Thus, given a base learner, the design decisions for a boosting algorithm are (1) are how to modify the probability distribution from one round to the next, and (2) how to combine the hypotheses $\{h_t\}_{t=1,\ldots,T}$ to form a final output hypothesis.

In this section we will use boosting on queries — that is, for the purposes of the boosting algorithm the universe $\mathcal{U}$ is a set of queries $\mathcal{Q}$ — to obtain an offline algorithm for answering large numbers of arbitrary low-sensitivity queries. This algorithm requires less space than the median mechanism, and, depending on the base learner, is potentially more time efficient as well.

The algorithm revolves around a somewhat magical fact (Lemma 6.5): if we can find a synopsis that provides accurate answers on a few selected queries, then in fact this synopsis provides accurate answers on *most* queries! We apply this fact to the base learner, which samples from a distribution on $\mathcal{Q}$ and produces as output a "weak" synopsis that yields "good" answers for a majority of the weight in $\mathcal{Q}$, boosting, in a differentially private fashion, to obtain a synopsis that is good for all of $\mathcal{Q}$.

Although the boosting is performed over the queries, the privacy is still for the rows of the database. The privacy challenge in boosting for queries comes from the fact that each row in the database affects the

answers to all the queries. This will manifest in the reweighting of the queries: adjacent databases could cause radically different reweightings, which will be observable in the generated $h_t$ that, collectively, will form the synopsis.

The running time of the boosting procedure depends quasi-linearly on the number $|\mathcal{Q}|$ of queries and on the running time of the base synopsis generator, independent of the data universe size $|\mathcal{X}|$. This yields a new avenue for constructing efficient and accurate privacy-preserving mechanisms, analogous to the approach enabled by boosting in the machine learning literature: an algorithm designer can tackle the (potentially much easier) task of constructing a weak privacy-preserving base synopsis generator, and automatically obtain a stronger mechanism.

## 6.1 The boosting for queries algorithm

We will use the *row representation* for databases, outlined in Section 2, where we think of the database as a multiset of rows, or elements of $\mathcal{X}$. Fix a database size $n$, a data universe $\mathcal{X}$, and a query set $\mathcal{Q} = \{q : \mathcal{X}^* \to \mathbb{R}\}$ of real-valued queries of sensitivity at most $\rho$.

We assume the existence of a *base synopsis generator* (in Section 6.2 we will see how to construct these). The property we will need of the base generator, formulated next, is that, for any distribution $\mathcal{D}$ on the query set $\mathcal{Q}$, the output of base generator can be used for computing accurate answers for a *large fraction* of the queries, where the "large fraction" is defined in terms of the weights given by $\mathcal{D}$. The base generator is parameterized by $k$, the number of queries to be sampled; $\lambda$, an accuracy requirement for its outputs; $\eta$, a measurement of "large" describing what we mean by a large fraction of the queries, and $\beta$, a failure probability.

**Definition 6.1** (($k, \lambda, \eta, \beta$)-base synopsis generator)**.** For a fixed database size $n$, data universe $\mathcal{X}$ and query set $\mathcal{Q}$, consider a synopsis generator $\mathcal{M}$, that samples $k$ queries independently from a distribution $\mathcal{D}$ on $\mathcal{Q}$ and outputs a synopsis. We say that $\mathcal{M}$ is a $(k, \lambda, \eta, \beta)$-*base synopsis generator* if for any distribution $\mathcal{D}$ on $\mathcal{Q}$, with all but $\beta$ probability

over the coin flips of $\mathcal{M}$, the synopsis $\mathcal{S}$ that $\mathcal{M}$ outputs is $\lambda$-accurate for a $(1/2+\eta)$-fraction of the mass of $\mathcal{Q}$ as weighted by $\mathcal{D}$:

$$\Pr_{q\sim\mathcal{D}}[|q(\mathcal{S})-q(x)|\leq\lambda]\geq 1/2+\eta. \tag{6.1}$$

The query-boosting algorithm can be used for any class of queries and any differentially private base synopsis generator. The running time is inherited from the base synopsis generator. The booster invests additional time that is quasi-linear in $|\mathcal{Q}|$, and in particular its running time does not depend directly on the size of the data universe.

To specify the boosting algorithm we will need to specify a stopping condition, an aggregation mechanism, and an algorithm for updating the current distribution on $\mathcal{Q}$.

**Stopping Condition.** We will run the algorithm for a fixed number $T$ of rounds — this will be our stopping condition. $T$ will be selected so as to ensure sufficient accuracy (with very high probability); as we will see, $\log|\mathcal{Q}|/\eta^2$ rounds will suffice.

**Updating the Distribution.** Although the distributions are never directly revealed in the outputs, the base synopses $\mathcal{A}_1,\mathcal{A}_2,\ldots,\mathcal{A}_T$ are revealed, and each $\mathcal{A}_i$ can in principle leak information about the queries chosen, from $\mathcal{D}_i$, in constructing $\mathcal{A}_i$. We therefore need to constrain the max-divergence between the probability distributions obtained on neighboring databases. This is technically challenging because, given $\mathcal{A}_i$, the database is very heavily involved in constructing $\mathcal{D}_{i+1}$.

The initial distribution, $\mathcal{D}_1$, will be uniform over $\mathcal{Q}$. A standard method for updating $\mathcal{D}_t$ is to increase the weight of poorly handled elements, in our case, queries for which $|q(x)-q(A_t)|>\lambda$, by a fixed factor, say, $e$, and decrease the weight of well-handled elements by the same factor. (The weights are then normalized so as to sum to 1.) To get a feel for the difficulty, let $x=y\cup\{\xi\}$, and suppose that all queries $q$ are handled well by $\mathcal{A}_t$ when the database is $y$, but the addition of $\xi$ causes this to fail for, say, a $1/10$ fraction of the queries; that is, $|q(y)-q(A_t)|\leq\lambda$ for all queries $q$, but $|q(x)-q(A_t)|>\lambda$ for some $|\mathcal{Q}|/10$ queries. Note that, since $\mathcal{A}_t$ "does well" on $9/10$ of the queries even

when the database is $x$, it could be returned from the base sanitizer no matter which of $x, y$ is the true data set. Our concern is with the effects of the updating: when the database is $y$ all queries are well handled and there is no reweighting (after normalization), but when the database is $x$ there is a reweighting: one tenth of the queries have their weights increased, the remaining nine tenths have their weights decreased. This difference in reweighting may be detected in the next iteration via $\mathcal{A}_{t+1}$, which is observable, and which will be built from samples drawn from rather different distributions depending on whether the database is $x$ or $y$.

For example, suppose we start from the uniform distribution $\mathcal{D}_1$. Then $\mathcal{D}_2^{(y)} = \mathcal{D}_1^{(y)}$, where by $\mathcal{D}_i^{(z)}$ we mean the distribution at round $i$ when the database is $z$. This is because the weight of every query is decreased by a factor of $e$, which disappears in the normalization. So each $q \in \mathcal{Q}$ is assigned weight $1/|\mathcal{Q}|$ in $\mathcal{D}_2^{(y)}$. In contrast, when the database is $x$ the "unhappy" queries have normalized weight

$$\frac{\frac{e}{|\mathcal{Q}|}}{\frac{9}{10}\frac{1}{|\mathcal{Q}|}\frac{1}{e} + \frac{1}{10}\frac{e}{|\mathcal{Q}|}}.$$

Consider any such unhappy query $q$. The *ratio* $\mathcal{D}_2^{(x)}(q)/\mathcal{D}_2^{(y)}(q)$ is given by

$$\begin{aligned}\frac{\mathcal{D}_2^{(x)}(q)}{\mathcal{D}_2^{(y)}(q)} &= \frac{\frac{\frac{e}{|\mathcal{Q}|}}{\frac{9}{10}\frac{1}{|\mathcal{Q}|}\frac{1}{e} + \frac{1}{10}\frac{e}{|\mathcal{Q}|}}}{\frac{1}{|\mathcal{Q}|}} \\ &= \frac{10}{1+\frac{9}{e^2}} \stackrel{\text{def}}{=} F \approx 4.5085.\end{aligned}$$

Now, $\ln F \approx 1.506$, and even though the choice of queries used in round 2 by the base generator are not explicitly made public, they may be detectable from the resulting $\mathcal{A}_2$, which is made public. Thus, there is a potential privacy loss of up to 1.506 per query (of course, we expect cancellations; we are simply trying to explain the source of the difficulty). This is partially addressed by ensuring that the number of samples used by the base generator is relatively small, although we still have the problem that, over multiple iterations, the distributions $\mathcal{D}_t$ may evolve very differently even on neighboring databases.

The solution will be to attenuate the re-weighting procedure. Instead of always using a fixed ratio either for increasing the weight (when the answer is "accurate") or decreasing it (when it is not), we set separate thresholds for "accuracy" ($\lambda$) and "inaccuracy" ($\lambda + \mu$, for an appropriately chosen $\mu$ that scales with the *bit size* of the output of the base generator; see Lemma 6.5 below). Queries for which the error is below or above these thresholds have their weight decreased or increased, respectively, by a factor of $e$. For queries whose error lies between these two thresholds, we scale the natural logarithm of the weight change linearly: $1 - 2(|q(x) - q(A_t)| - \lambda)/\mu$, so queries with errors of magnitude exceeding $\lambda + \mu/2$ increase in weight, and those with errors of magnitude less than $\lambda + \mu/2$ decrease in weight.

The attenuated scaling reduces the effect of any individual on the re-weighting of any query. This is because an individual can only affect the true answer to a query — and thus also the accuracy of the base synopsis generator's output $q(A_t)$ — by a small amount, and the attenuation divides this amount by a parameter $\mu$ which will be chosen to compensate for the $kT$ samples chosen (total) from the $T$ distributions obtained over the course of the execution of the boosting algorithm. This helps to ensure privacy. Intuitively, we view each of these $kT$ samples as a "mini-mechanism." We first bound the privacy loss of sampling at any round (Claim 6.4) and then bound the cumulative loss via the composition theorem.

The larger the gap ($\mu$) between the thresholds for "accurate" and "inaccurate," the smaller the effect of each individual on a query's weight can be. This means that larger gaps are better for privacy. For accuracy, however, large gaps are bad. If the inaccuracy threshold is large, we can only guarantee that queries for which the base synopsis generator is very inaccurate have their weight substantially increased during re-weighting. This degrades the accuracy guarantee of the boosting algorithm: the errors are roughly equal to the "inaccuracy" threshold ($\lambda + \mu$).

**Aggregation.** For $t \in [T]$ we will run the base generator to obtain a synopsis $\mathcal{A}_t$. The synopses will be aggregated by taking the median: given $\mathcal{A}_1, \ldots, \mathcal{A}_T$, the quantity $q(x)$ is estimated by taking the $T$

approximate values for $q(x)$ computed using each of the $\mathcal{A}_i$, and then computing their median. With this aggregation method we can show accuracy for query $q$ by arguing that a majority of the $\mathcal{A}_i$, $1 \leq i \leq T$ provide $\lambda + \mu$ accuracy (or better) for $q$. This implies that the median value of the $T$ approximations to $q(x)$ will be within $\lambda + \mu$ of the true value.

**Notation.**

1. Throughout the algorithm's operation, we keep track of several variables (explicitly or implicitly). Variables indexed by $q \in \mathcal{Q}$ hold information pertaining to query $q$ in the query set. Variables indexed by $t \in [T]$, usually computed in round $t$, will be used to construct the distribution $\mathcal{D}_{t+1}$ used for sampling in time period $t + 1$.
2. For a predicate $P$ we use $[[P]]$ to denote 1 if the predicate is true and 0 if it is false.
3. There is a final tuning parameter $\alpha$ used in the algorithm. It will be chosen (see Corollary 6.3 below) to have value

$$\alpha = \alpha(\eta) = (1/2) \ln \left( \frac{1 + 2\eta}{1 - 2\eta} \right).$$

The algorithm appears in Figure 6.1. The quantity $u_{t,q}$ in Step 2(2b) is the new, un-normalized, weight of the query. For the moment, let us set $\alpha = 1$ (just so that we can ignore any $\alpha$ factors). Letting $a_{j,q}$ be the natural logarithm of the weight change in round $j$, $1 \leq j \leq t$, the new weight is given by:

$$u_{t,q} \leftarrow \exp \left( - \sum_{j=1}^{t} a_{j,q} \right).$$

Thus, at the end of the previous step the un-normalized weight was $u_{t-1,q} = \exp(-\sum_{j=1}^{t-1} a_{j,q})$ and the update corresponds to multiplication by $e^{-a_{j,t}}$. When the sum $\sum_{j=1}^{t} a_{j,q}$ is large, the weight is small. Every time a synopsis gives a very good approximation to $q(x)$, we add 1 to this sum; if the approximation is only moderately good (between $\lambda$ and

**Boosting for Queries**$(k, \lambda, \eta, \rho, \mu, T)$
Given: database $x \in \mathcal{X}^n$, query set $\mathcal{Q}$, where each $q \in \mathcal{Q}$ is a function $q : X^n \to \mathbb{R}$ with sensitivity at most $\rho$.
Initialize $\mathcal{D}_1$ to be the uniform distribution over $\mathcal{Q}$.
For $t = 1, \ldots, T$:

1. Sample a sequence $S_t \subseteq \mathcal{Q}$ of $k$ samples chosen independently and at random from $\mathcal{D}_t$.
   Run the base synopsis generator to compute a synopsis $\mathcal{A}_t : \mathcal{Q} \to \mathbb{R}$ that is w.h.p. accurate for at least $1/2 + \eta$ of the mass of $\mathcal{D}_t$.
2. Reweight the queries. For each $q \in \mathcal{Q}$:
   (a) If $\mathcal{A}_t$ is $\lambda$-accurate, then $a_{t,q} \leftarrow 1$
   If $\mathcal{A}_t$ is $(\lambda + \mu)$-inaccurate, then $a_{t,q} \leftarrow -1$
   Otherwise, let $d_{q,t} = |q(x) - \mathcal{A}_t(q)|$ be the error of $\mathcal{A}_t$ (between $\lambda$ and $\lambda + \mu$) on $q$:
   $$a_{t,q} \leftarrow 1 - 2(d_{t,q} - \lambda)/\mu$$
   (b) $u_{t,q} \leftarrow \exp(-\alpha \cdot \sum_{j=1}^{t} a_{j,q})$, where $\alpha = (1/2)\ln((1+2\eta)/(1-2\eta))$.
3. Renormalize:
   $$Z_t \leftarrow \sum_{q \in \mathcal{Q}} u_{t,q}$$
   $$D_{t+1}[q] = u_{t,q}/Z_t$$

Output the final answer data structure $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_T)$. For $q \in \mathcal{Q}$:
$$\mathcal{A}(q) = median\{\mathcal{A}_1(q), \ldots, \mathcal{A}_T(q)\}$$

**Figure 6.1:** Boosting for queries.

$\lambda + \mu/2$), we add a positive amount, but less than 1. Conversely, when the synopsis is very bad (worse than $\lambda + \mu$ accuracy), we subtract 1; when it is barely acceptable (between $\lambda + \mu/2$ and $\lambda + \mu$), we subtract a smaller amount.

In the theorem below we see an inverse relationship between privacy loss due to sampling, captured by $\varepsilon_{\text{sample}}$, and the gap $\mu$ between the thresholds for accurate and inaccurate.

**Theorem 6.1.** Let $\mathcal{Q}$ be a query family with sensitivity at most $\rho$. For an appropriate setting of parameters, and with $T = \log|\mathcal{Q}|/\eta^2$ rounds, the algorithm of Figure 6.1 is an accurate and differentially private query-boosting algorithm:

1. When instantiated with a $(k, \lambda, \eta, \beta)$-base synopsis generator, the output of the boosting algorithm gives $(\lambda + \mu)$-accurate answers to *all* the queries in $\mathcal{Q}$ with probability at least $1 - T\beta$, where
   $$\mu \in O(((\log^{3/2}|Q|)\sqrt{k}\sqrt{\log(1/\beta)}\rho)/(\varepsilon_{\text{sample}} \cdot \eta^3)). \quad (6.2)$$

2. If the base synopsis generator is $(\varepsilon_{\text{base}}, \delta_{\text{base}})$-differentially private, then the boosting algorithm is $(\varepsilon_{\text{sample}} + T \cdot \varepsilon_{\text{base}}, \delta_{\text{sample}} + T\delta_{\text{base}})$-differentially private.

Allowing the constant $\eta$ to be swallowed up into the big-O notation, and taking $\rho = 1$ for simplicity, we get $\mu = O(((\log^{3/2} |Q|)\sqrt{k}\sqrt{\log(1/\beta)})/\varepsilon_{\text{sample}})$. Thus we see that reducing the number $k$ of input queries needed by the base sanitizer improves the quality of the output. Similarly, from the full statement of the theorem, we see that improving the generalization power of the base sanitizer, which corresponds to having a larger value of $\eta$ (a bigger "strong majority"), also improves the accuracy.

*Proof of Theorem 6.1.* We first prove accuracy, then privacy.

We introduce the notation $a^-_{t,q}$ and $a^+_{t,q}$, satisfying

1. $a^-_{t,q}, a^+_{t,q} \in \{-1, 1\}$; and
2. $a^-_{t,q} \leq a_{t,q} \leq a^+_{t,q}$.

Recall that a larger $a_{t,q}$ indicates a higher quality of the approximation of the synopsis $\mathcal{A}_t$ for $q(x)$.

1. $a^-_{t,q}$ is 1 if $\mathcal{A}_t$ is $\lambda$-accurate on $q$, and $-1$ otherwise. To check that $a^-_{t,q} \leq a_{t,q}$, note that if $a^-_{t,q} = 1$ then $\mathcal{A}_t$ is $\lambda$-accurate for $q$, and so by definition $a_{t,q} = 1$ as well. If instead we have $a^-_{t,q} = -1$ then since we always have $a_{t,q} \in [-1, 1]$, we are done.

   We will use the $a^-_{t,q}$ to lower bound a measure of the quality of the output of the base generator. By the promise of the base generator, $\mathcal{A}_t$ is $\lambda$-accurate for at least a $1/2 + \eta$ fraction of the mass of $\mathcal{D}_t$. Thus,

$$r_t \triangleq \sum_{q \in \mathcal{Q}} \mathcal{D}_t[q] \cdot a^-_{t,q} \geq (1/2 + \eta) - (1/2 - \eta) = 2\eta. \tag{6.3}$$

2. $a^+_{t,q}$ is $-1$ if $\mathcal{A}_t$ is $(\lambda + \mu)$-*inaccurate* for $q$, and 1 otherwise. To check that $a_{t,q} \leq a^+_{t,q}$, note that if $a^+_{t,q} = -1$ then $\mathcal{A}_t$ is $(\lambda + \mu)$-*inaccurate* for $q$, so by definition $a_{t,q} = -1$ as well. If instead $a^+_{t,q} = 1$ then since we always have $a_{t,q} \in [-1, 1]$, we are done.

   Thus $a^+_{t,q}$ is positive if and only if $\mathcal{A}_t$ is at least minimally adequately accurate for $q$. We will use the $a^+_{t,q}$ to prove accuracy

of the aggregation. When we sum the values $a^+_{t,q}$, we get a positive number if and only if the majority of the $\mathcal{A}_t$ are providing passable — that is, within $\lambda + \mu$ — approximations to $q(x)$. In this case the median value will be within $\lambda + \mu$.

**Lemma 6.2.** After $T$ rounds of boosting, with all but $T\beta$ probability, the answers to all but an $\exp(-\eta^2 T)$-fraction of the queries are $(\lambda+\mu)$-accurate.

*Proof.* In the last round of boosting, we have:

$$\mathcal{D}_{T+1}[q] = \frac{u_{T,q}}{Z_T}. \tag{6.4}$$

Since $a_{t,q} \leq a^+_{t,q}$ we have:

$$u^+_{T,q} \triangleq e^{-\alpha \sum_{t=1}^T a^+_{t,q}} \leq e^{-\alpha \sum_{t=1}^T a_{t,q}} = u_{T,q}. \tag{6.5}$$

(The superscript "+" reminds us that this unweighted value was computed using the terms $a^+_{t,q}$.) Note that we always have $u^+_{T,q} \geq 0$. Combining Equations (6.4) and (6.5), for all $q \in \mathcal{Q}$:

$$\mathcal{D}_{T+1}[q] \geq \frac{u^+_{T,q}}{Z_T}. \tag{6.6}$$

Recalling that $[[P]]$ denotes the boolean variable that has value 1 if and only if the predicate $P$ is true, we turn to examining the value $[[\mathcal{A}$ is $(\lambda+\mu)$-inaccurate for $q]]$. If this predicate is 1, then it must be the case that the majority of $\{\mathcal{A}_j\}_{j=1}^T$ are $(\lambda+\mu)$-inaccurate, as otherwise their median would be $(\lambda+\mu)$-accurate.

From our discussion of the significance of the sign of $\sum_{t=1}^T a^+_{t,q}$, we have:

$$\begin{aligned} \mathcal{A} \text{ is } (\lambda+\mu)\text{-inaccurate for } q &\Rightarrow \sum_{t=1}^T a^+_{t,q} \leq 0 \\ &\Leftrightarrow e^{-\alpha \sum_{t=1}^T a^+_{t,q}} \geq 1 \\ &\Leftrightarrow u^+_{T,q} \geq 1 \end{aligned}$$

Since $u^+_{T,q} \geq 0$, We conclude that:

$$[[\mathcal{A} \text{ is } (\lambda+\mu)\text{-inaccurate for } q]] \leq u^+_{T,q}$$

Using this together with Equation (6.6) yields:

$$\begin{aligned}\frac{1}{|\mathcal{Q}|}\cdot\sum_{q\in\mathcal{Q}}[[\mathcal{A}\text{ is }(\lambda+\mu)\text{-inaccurate for }q]] &\leq \frac{1}{|\mathcal{Q}|}\cdot\sum_{q\in\mathcal{Q}}u^{+}_{T,q}\\ &\leq \frac{1}{|\mathcal{Q}|}\cdot\sum_{q\in\mathcal{Q}}\mathcal{D}_{T+1}[q]\cdot Z_T\\ &= \frac{Z_T}{|\mathcal{Q}|}.\end{aligned}$$

Thus the following claim completes the proof:

**Claim 6.3.** In round $t$ of boosting, with all but $t\beta$ probability:

$$Z_t \leq \exp(-\eta^2\cdot t)\cdot|\mathcal{Q}|$$

*Proof.* By definition of a base synopsis generator, with all but $\beta$ probability, the synopsis generated is $\lambda$-accurate for at least a $(1/2+\eta)$-fraction of the mass of the distribution $\mathcal{D}_t$. Recall that $a^{-}_{t,q}\in\{-1,1\}$ is 1 if and only if $\mathcal{A}_t$ is $\lambda$-accurate on $q$, and that $a^{-}_{t,q}\leq a_{t,q}$ and recall further the quantity $r_t \triangleq \sum_{q\in\mathcal{Q}}\mathcal{D}_t[q]\cdot a^{-}_{t,q}$ defined in Equation (6.3). As discussed above, $r_t$ measures the "success" of the base synopsis generator in round $t$, where by "success" we mean the stricter notion of $\lambda$-accuracy. As summarized in Equation (6.3), if a $(1/2+\eta)$-fraction of the mass of $\mathcal{D}_t$ is computed with $\lambda$-accuracy, then $r_t\geq 2\eta$. Now observe also that for $t\in[T]$, assuming the base sanitizer did not fail in round $t$:

$$\begin{aligned}Z_t &= \sum_{q\in\mathcal{Q}}u_{t,q}\\ &= \sum_{q\in\mathcal{Q}}u_{t-1,q}\cdot e^{-\alpha\cdot a_{t,q}}\\ &= \sum_{q\in\mathcal{Q}}Z_{t-1}\cdot\mathcal{D}_t[q]\cdot e^{-\alpha\cdot a_{t,q}}\\ &\leq \sum_{q\in\mathcal{Q}}Z_{t-1}\cdot\mathcal{D}_t[q]\cdot e^{-\alpha\cdot a^{-}_{t,q}}\\ &= Z_{t-1}\cdot\sum_{q\in\mathcal{Q}}\mathcal{D}_t[q]\cdot\left(\left(\frac{1+a^{-}_{t,q}}{2}\right)\cdot e^{-\alpha}+\left(\frac{1-a^{-}_{t,q}}{2}\right)\cdot e^{\alpha}\right)\end{aligned}$$

(case analysis)

$$= \frac{Z_{t-1}}{2}\left[(e^{\alpha}+e^{-\alpha})+r_t(e^{-\alpha}-e^{\alpha})\right]$$
$$\leq \frac{Z_{t-1}}{2}\left[(e^{\alpha}+e^{-\alpha})+2\eta(e^{-\alpha}-e^{\alpha})\right] \ (r_t \geq 2\eta \text{ and } (e^{-\alpha}-e^{\alpha}) \leq 0)$$

By simple calculus we see that $(e^{\alpha}+e^{-\alpha})+2\eta(e^{-\alpha}-e^{\alpha})$ is minimized when

$$\alpha = (1/2)\ln\left(\frac{1+2\eta}{1-2\eta}\right).$$

Plugging this into the recurrence, we get

$$Z_t \leq (\sqrt{1-4\eta^2})^t|\mathcal{Q}| \leq \exp(-2\eta^2 t)|\mathcal{Q}|.$$

□

This completes the proof of Lemma 6.2. □

The lemma implies that accuracy for *all* queries simultaneously can be achieved by setting

$$T > \frac{\ln|\mathcal{Q}|}{\eta^2}.$$

**Privacy.** We will show that the entire sequence $(S_1, \mathcal{A}_1, \ldots, S_T, \mathcal{A}_T)$ can be output while preserving differential privacy. Note that this is stronger than we need — we do not actually output the sets $S_1, \ldots, S_T$. By our adaptive composition theorems, the privacy of each $\mathcal{A}_i$ will be guaranteed by the privacy guarantees of the base synopsis generator, together with the fact that $S_{i-1}$ was computed in a differentially private way. Therefore, it suffices to prove that given that $(S_1, \mathcal{A}_1, \ldots, S_i, \mathcal{A}_i)$ is differentially private, $S_{i+1}$ is as well. We can then combine the privacy parameters using our composition theorems to compute a final guarantee.

**Lemma 6.4.** Let $\varepsilon^* = \frac{4\alpha T\rho}{\mu}$. For all $i \in [T]$, once $(S_1, \mathcal{A}_1, \ldots, S_i, \mathcal{A}_i)$ is fixed, the computation of each element of $S_{i+1}$ is $(\varepsilon^*, 0)$-differentially private.

*Proof.* Fixing $\mathcal{A}_1, \ldots, \mathcal{A}_i$, for every $j \leq i$, the quantity $d_{q,j}$ has sensitivity $\rho$, since $\mathcal{A}_j(q)$ is database independent (because $\mathcal{A}_j$ is fixed), and

every $q \in \mathcal{Q}$ has sensitivity bounded by $\rho$. Therefore, for every $j \leq i$, $a_{j,q}$ is $2\rho/\mu$ sensitive by construction, and so

$$g_i(q) \stackrel{\text{def}}{=} \sum_{j=1}^{i} a_{j,q}$$

has sensitivity at most $2i\rho/\mu \leq 2T\rho/\mu$. Then $\Delta g_i \stackrel{\text{def}}{=} 2T\rho/\mu$ is an upper bound on the sensitivity of $g_i$.

To argue privacy, we will show that the selection of queries for $S_{i+1}$ is an instance of the exponential mechanism. Think of $-g_i(q)$ as the utility of a query $q$ during the selection process at round $i+1$. The exponential mechanism says that to achieve $(\varepsilon^*, 0)$-differential privacy we should choose $q$ with probability proportional to

$$\exp\left(-g_i(q)\frac{\varepsilon^*}{2\Delta g_i}\right).$$

Since $\varepsilon^*/2\Delta g_i = \alpha$ and the algorithm selects $q$ with probability proportional to $e^{-\alpha g_i(q)}$, we see that this is exactly what the algorithm does! □

We bound the privacy loss of releasing the $S_i$s by treating each selection of a query as a "mini-mechanism" that, over the course of $T$ rounds of boosting, is invoked $kT$ times. By Lemma 6.4 each mini-mechanism is $(4\alpha T\rho/\mu, 0)$-differentially private. By Theorem 3.20, for all $\beta > 0$ the composition of $kT$ mechanisms, each of which is $(\alpha 4T\rho/\mu, 0)$-differentially private, is $(\varepsilon_{\text{sample}}, \delta_{\text{sample}})$-differentially private, where

$$\varepsilon_{\text{sample}} \stackrel{\text{def}}{=} \sqrt{2kT\log(1/\delta_{\text{sample}})}(\alpha 4T\rho/\mu) + kT\left(\frac{\alpha 4T\rho}{\mu}\right)^2. \quad (6.7)$$

Our total privacy loss comes from the composition of $T$ calls to the base sanitizer and the cumulative loss from the $kT$ samples. We conclude that the boosting algorithm in its entirety is: $(\varepsilon_{\text{boost}}, \delta_{\text{boost}})$-differentially private, where

$$\begin{aligned}\varepsilon_{\text{boost}} &= T\varepsilon_{\text{base}} + \varepsilon_{\text{sample}}\\ \delta_{\text{boost}} &= T\delta_{\text{base}} + \delta_{\text{sample}}\end{aligned}$$

To get the parameters claimed in the statement of the theorem, we can take:

$$\mu \in O((T^{3/2}\sqrt{k}\sqrt{\log(1/\beta)}\alpha\rho)/\varepsilon_{\text{sample}}). \tag{6.8}$$

□

## 6.2 Base synopsis generators

Algorithm SmallDB (Section 4) is based on the insight that a small randomly selected subset of database rows provides good answers to large sets of fractional counting queries. The base synopsis generators described in the current section have an analogous insight: a small synopsis that gives good approximations to the answers to a small subset of queries also yields good approximations to most queries. Both of these are instances of *generalization bounds*. In the remainder of this section we firs prove a generalization bound and then use it to construct differentially base synopsis generators.

### 6.2.1 A generalization bound

We have a distribution $\mathcal{D}$ over a large set $\mathcal{Q}$ of queries to be approximated. The lemma below says that a sufficiently small synopsis that gives sufficiently good approximations to the answers of a *randomly selected* subset $S \subset \mathcal{Q}$ of queries, sampled according to the distribution $\mathcal{D}$ on $\mathcal{Q}$, will, with high probability over the choice of $S$, also give good approximations to the answers to *most* queries in $\mathcal{Q}$ (that is, to most of the mass of $\mathcal{Q}$, weighted by $D$). Of course, to make any sense the synopsis must include a method of providing an answer to all queries in $\mathcal{Q}$, not just the subset $S \subseteq \mathcal{Q}$ received as input. Our particular generators, described in Sections 6.2.2 and Theorem 6.6 will produce synthetic databases; to answer any query one can simply apply the query to the synthetic database, but the lemma will be stated in full generality.

Let $R(y, q)$ denote the answer given by the synopsis $y$ (when used as input for the reconstruction procedure) on query $q$. A synopsis $y$ $\lambda$-*fits* a database $x$ w.r.t a set $S$ of queries if $\max_{q\in S} |R(y, q) - q(x)| \leq \lambda$. Let $|y|$

denote the number of bits needed to represent $y$. Since our synopses will be synthetic databases, $|y| = N \log_2 |\mathcal{X}|$ for some appropriately chosen number $N$ of universe elements. The generalization bound shows that if $y$ $\lambda$-fits $x$ with respect to a large enough (larger than $|y|$) randomly chosen set $S$ of queries sampled from a distribution $\mathcal{D}$, then with high probability $y$ $\lambda$-fits $x$ for *most* of the mass of $\mathcal{D}$.

**Lemma 6.5.** Let $\mathcal{D}$ be an arbitrary distribution on a query set $\mathcal{Q} = \{q : \mathcal{X}^* \to \mathbb{R}\}$. For all $m \in \mathcal{N}$, $\gamma \in (0,1)$, $\eta \in [0, 1/2)$, let $a = 2(\log(1/\gamma) + m)/(m(1-2\eta))$. Then with probability at least $1-\gamma$ over the choice of $S \sim \mathcal{D}^{a \cdot m}$, every synopsis $y$ of size at most $m$ bits that $\lambda$-fits $x$ with respect to the query set $S$, also $\lambda$-fits $x$ with respect to at least a $(1/2 + \eta)$-fraction of $\mathcal{D}$.

Before proving the lemma we observe that $a$ is a compression factor: we are squeezing the answers to $am$ queries into an $m$-bit output, so larger $a$ corresponds to more compression. Typically, this means better generalization, and indeed we see that if $a$ is larger then, keeping $m$ and $\gamma$ fixed, we would be able to have larger $\eta$. The lemma also says that, for any given output size $m$, the number of queries needed as input to obtain an output that does well on a majority ($1/2 + \eta$ fraction) of $\mathcal{D}$ is only $O(\log(1/\gamma) + m)$. This is interesting because a smaller number of queries $k$ needed by the base generator leads, via the privacy loss $\varepsilon_{\text{sample}}$ due to sampling of $kT$ queries and its inverse relationship to the slackness $\mu$ (Equation 6.7), to improved accuracy of the output of the boosting algorithm.

*Proof of Lemma 6.5.* Fix a set of queries $S \subset \mathcal{Q}$ chosen independently according to $\mathcal{D}^{a \cdot m}$. Examine an arbitrary $m$-bit synopsis $y$. Note that $y$ is described by an $m$-bit string. Let us say $y$ is *bad* if $|R(y,q) - q(x)| > \lambda$ for at least a $(\log(1/\gamma) + m)/(a \cdot m)$ fraction of $\mathcal{D}$, meaning that $\Pr_{q \sim \mathcal{D}}[|R(y,q) - q(x)| > \lambda] \geq (\log(1/\gamma) + m)/(a \cdot m)$.

In other words, $y$ is bad if there exists a set $Q_y \subset \mathcal{Q}$ of fractional weight at least $(\log(1/\gamma) + m)/(a \cdot m)$ such that $|R(y,q) - q(x)| > \lambda$ for $q \in Q_y$. For such a $y$, what is the probability that $y$ gives $\lambda$-accurate answers for *every* $q \in S$? This is exactly the probability that none of

the queries in $S$ is in $Q_y$, or

$$(1 - (\log(1/\gamma) + m)/(a \cdot m))^{a \cdot m} \leq e^{-(\log(1/\gamma)+m)} \leq \gamma \cdot 2^{-m}$$

Taking a union bound over all $2^m$ possible choices for $y$, the probability that there exists an $m$-bit synopsis $y$ that is accurate on all the queries in $S$ but inaccurate on a set of fractional weight $(\log(1/\beta)+m)/(a \cdot m)$ is at most $\gamma$. Letting $k = am = |S|$ we see that it is sufficient to have

$$a > \frac{2(\log(1/\gamma) + m)}{m \cdot (1 - 2\eta)}. \tag{6.9}$$

□

This simple lemma is extremely powerful. It tells us that when constructing a base generator at round $t$, we only need to worry about ensuring good answers for the small set of random queries sampled from $\mathcal{D}_t$; doing well for most of $\mathcal{D}_t$ will happen automatically!

### 6.2.2 The base generator

Our first generator works by brute force. After sampling a set $S$ of $k$ queries independently according to a distribution $\mathcal{D}$, the base generator will produce noisy answers for all queries in $S$ via the Laplace mechanism. Then, making no further use of the actual database, the algorithm searches for *any* database of size $n$ for which these noisy answers are sufficiently close, and outputs this database. Privacy will be immediate because everything after the $k$ invocations of the Laplace mechanism is in post-processing. Thus the only source of privacy loss is the cumulative loss from these $k$ invocations of the Laplace mechanism, which we know how to analyze via the composition theorem. Utility will follow from the utility of the Laplace mechanism — which says that we are unlikely to have "very large" error on even one query — coupled with the fact that the true database $x$ is an $n$-element database that fits these noisy responses.[1]

[1]This argument assumes the size $n$ of the database is known. Alternatively we can include a noisy query of the form "How many rows are in the database?" and exhaustively search all databases of size close to the response to this query.

**Theorem 6.6** (Base Synopsis Generator for Arbitrary Queries)**.** For any data universe $\mathcal{X}$, database size $n$, and class $\mathcal{Q} : \{\mathcal{X}^* \to \mathbb{R}\}$ of queries of sensitivity at most $\rho$, for any $\varepsilon_{\text{base}}, \delta_{\text{base}} > 0$, there exists an $(\varepsilon_{\text{base}}, \delta_{\text{base}})$-differentially private $(k, \lambda, \eta = 1/3, \beta)$-base synopsis generator for $\mathcal{Q}$, where $k = am > 6(m + \log(2/\beta)) = 6(n \log |\mathcal{X}| + \log(2/\beta))$ and $\lambda > 2b(\log k + log(2/\beta))$, where $b = \rho\sqrt{am \log(1/\delta_{\text{base}})}/\varepsilon_{\text{base}}$.

The running time of the generator is

$$|\mathcal{X}|^n \cdot \text{poly}(n, \log(1/\beta), \log(1/\varepsilon_{\text{base}}), \log(1/\delta_{\text{base}})).$$

*Proof.* We first describe the base generator at a high level, then determine the values for $k$ and $\lambda$. The synopsis $y$ produced by the base generator will be a synthetic database of size $n$. Thus $m = |y| = n \cdot \log |\mathcal{X}|$. The generator begins by choosing a set $S$ of $k$ queries, sampled independently according to $\mathcal{D}$. It computes a noisy answer for each query $q \in S$ using the Laplace mechanism, adding to each true answer an independent draw from $\text{Lap}(b)$ for an appropriate $b$ to be determined later. Let $\{\widehat{q(x)}\}_{q \in \mathcal{Q}}$ be the collection of noisy answers. The generator enumerates over all $|\mathcal{X}|^n$ databases of size $n$, and outputs the lexicographically first database $y$ such that for every $q \in S$ we have $|q(y) - \widehat{q(x)}| \leq \lambda/2$. If no such database is found, it outputs $\perp$ instead, and we say it *fails*. Note that if $|\widehat{q(x)} - q(x)| < \lambda/2$ and $|q(y) - \widehat{q(x)}| < \lambda/2$, then $|q(y) - q(x)| < \lambda$.

There are two potential sources of failure for our particular generator. One possibility is that $y$ fails to generalize, or is *bad* as defined in the proof of Lemma 6.5. A second possibility is that one of the samples from the Laplace distribution is of excessively large magnitude, which might cause the generator to fail. We will choose our parameters so as to bound the probability of each of these events individually by at most $\beta/2$.

Substituting $\eta = 1/3$ and $m = n \log |X|$ into Equation 6.9 shows that taking $a > 6(1 + \log(2/\beta)/m)$ suffices in order for the probability of failure due to the choice of $S$ to be bounded by $\beta/2$. Thus, taking $k = am > 6(m + \log(2/\beta)) = 6(n \log |\mathcal{X}| + \log(2/\beta))$ suffices.

We have $k$ queries of sensitivity at most $\rho$. Using the Laplace mechanism with parameter $b = 2\sqrt{2k \log(1/\delta_{\text{base}})}\rho/\varepsilon_{\text{base}}$, ensures that each query incurs privacy loss at most $\varepsilon_{\text{base}}/\sqrt{2k \ln(1/\delta_{\text{base}})}$, which by

Corollary 3.21 ensures that the entire procedure will be $(\varepsilon_{\text{base}}, \delta_{\text{base}})$-differentially private.

We will choose $\lambda$ so that the probability that any draw from $\text{Lap}(b)$ has magnitude exceeding $\lambda/2$ is at most $\beta/2$. Conditioned on the event that all $k$ draws have magnitude at most $\lambda$ we know that the input database itself will $\lambda$-fit our noisy answers, so the procedure will not fail.

Recall that the concentration properties of the Laplace distribution ensure that with probability at least $1-e^t$ a draw from $\text{Lap}(b)$ will have magnitude bounded by $tb$. Setting $\lambda/2 = tb$, the probability that a given draw will have magnitude exceeding $\lambda/2$ is bounded by $e^{-t} = e^{-\lambda/2b}$. To ensure that none of the $k$ draws has magnitude exceeding $\lambda/2$ it suffices, by a union bound, to have

$$\begin{aligned} & ke^{-\lambda/2b} < \beta/2 \\ \Leftrightarrow\ & e^{\lambda/2b} > k\frac{2}{\beta} \\ \Leftrightarrow\ & \lambda/2 > b(\log k + \log(2/\beta)) \\ \Leftrightarrow\ & \lambda > 2b(\log k + \log(2/\beta)). \end{aligned}$$

□

**The Special Case of Linear Queries.** For the special case of linear queries it is possible to avoid the brute force search for a small database. The technique requires time that is polynomial in $(|\mathcal{Q}|, |\mathcal{X}|, n, \log(1/\beta))$. We will focus on the case of counting queries and sketch the construction.

As in the case of the base generator for arbitrary queries, the base generator begins by selecting a set $S$ of $k = am$ queries according to $\mathcal{D}$ and computing noisy answers using Laplace noise. The generator for linear queries then runs a *syntheticizer* on $S$ which, roughly speaking, transforms any synopsis giving good approximations to *any* set $R$ of queries into a synthetic database yielding approximations of similar quality on the set $R$. The input to the syntheticizer will be the noisy values for the queries in $S$, that is, $R = S$. (Recall that when we modify the size of the database we always think in terms of the fractional version of the counting queries: "What fraction of the database rows satisfies property $P$?")

The resulting database may be quite large, meaning it may have many rows. The base generator then subsamples only $n' = (\log k \log(1/\beta))/\alpha^2$ of the rows of the synthetic database, creating a smaller synthetic database that with probability at least $1 - \beta$ has $\alpha$-accuracy with respect to the answers given by the large synthetic database. This yields an $m = ((\log k \log(1/\beta))/\alpha^2) \log |\mathcal{X}|$-bit synopsis that, by the generalization lemma, with probability $(1 - \log(1/\beta))$ over the choice of the $k$ queries, answers well on a $(1/2 + \eta)$ fraction of $\mathcal{Q}$ (as weighted by $\mathcal{D}$).

As in the case of the base generator for arbitrary queries, we require $k = am > 6 \log(1/\beta) + 6m$. Taking $\alpha^2 = (\log \mathcal{Q})/n$ we get that

$$\begin{aligned} k &> 6 \log(1/\beta) + 6 \frac{\log k \log(1/\beta) \log |\mathcal{X}|}{\alpha^2} \\ &= 6 \log(1/\beta) + 6n \log k \log(1/\beta) \frac{\log |\mathcal{X}|}{\log |\mathcal{Q}|}. \end{aligned}$$

The syntheticizer is nontrivial. Its properties are summarized by the following theorem.

**Theorem 6.7.** Let $\mathcal{X}$ be a data universe, $\mathcal{Q}$ a set of fractional counting queries, and $A$ an $(\varepsilon, \delta)$-differentially private synopsis generator with utility $(\alpha, \beta, 0)$ and arbitrary output. Then there exists a syntheticizer $A'$ that is $(\varepsilon, \delta)$-differentially private and has utility $(3\alpha, \beta, 0)$. $A'$ outputs a (potentially large) synthetic database. Its running time is polynomial in the running time of $A$ and $(|\mathcal{X}|, |\mathcal{Q}|, 1/\alpha, \log(1/\beta))$.

In our case, $A$ is the Laplace mechanism, and the synopsis is simply the set of noisy answers. The composition theorem says that for $A$ to be $(\varepsilon_{\text{base}}, \delta_{\text{base}})$-differentially private the parameter to the Laplace mechanism should be $\rho/(\varepsilon_{\text{base}}/\sqrt{2k \log(1/\delta_{\text{base}})})$. For fractional counting queries the sensitivity is $\rho = 1/n$.

Thus, when we apply the Theorem we will have an $\alpha$ of order $(\sqrt{k \log(1/\beta)}/\varepsilon_{\text{base}})\rho$. Here, $\rho$ is the sensitivity. For counting queries it is 1, but we will shift to fractional counting queries, so $\rho = 1/n$.

*Proof Sketch for Theorem 6.7.* Run $A$ to get (differentially private) (fractional) counts on all the queries in $R$. We will then use linear programming to find a low-weight fractional database that approximates

these fractional counts, as explained below. Finally, we transform this fractional database into a standard synthetic database by rounding the fractional counts.

The output of $A$ yields a fractional count for each query $q \in \mathcal{Q}$. The input database $x$ is never accessed again and so $A'$ is $(\varepsilon, \delta)$-differentially private. Let $v$ be the resulting vector of counts, i.e., $v_q$ is the fractional count that $A$'s output gives on query $q$. With probability $1 - \beta$, all of the entries in $v$ are $\alpha$-accurate.

A "fractional" database $z$ that approximates these counts is obtained as follows. Recall the histogram representation of a database, where for each element in the universe $\mathcal{X}$ the histogram contains the number of instances of this element in the database. Now, for every $i \in \mathcal{X}$, we introduce a variable $a_i \geq 0$ that will "count" the (fractional) number of occurrences of $i$ in the fractional database $z$. We will impose the constraint

$$\sum_{i \in \mathcal{X}} a_i = 1.$$

We represent the count of query $q$ in $z$ as the sum of the count of items $i$ that satisfy $q$:

$$\sum_{i \in \mathcal{X} \text{ s.t. } q(i)=1} a_i$$

We want all of these counts to be within a an additive $\alpha$ accuracy of the respective counts in $v_q$. Writing this as a linear inequality we get:

$$(v_q - \alpha) \sum_{i \in \mathcal{X}} a_i \leq \sum_{i \in \mathcal{X} \text{ s.t. } q(i)=1} a_i \leq (v_q + \alpha) \sum_{i \in \mathcal{X}} a_i.$$

When the counts are all $\alpha$-accurate with respect to the counts in $v_c$, it is also the case that (with probability $1 - \beta$) they are all $2\alpha$-accurate with respect to the true counts on the original database $x$.

We write a linear program with two such constraints for each query (a total of $2|\mathcal{Q}|$ constraints). $A'$ tries to find a fractional solution to this linear program. To see that such a solution exists, observe that the database $x$ itself is $\alpha$-close to the vector of counts $v$, and so there *exists* a solution to the linear program (in fact even an integer solution), and hence $A'$ will find *some* fractional solution.

We conclude that $A'$ can generate a fractional database with $(2\alpha, \beta, 0)$-utility, but we really want a synthetic (integer) database. To transform the fractional database into an integer one, we round down each $a_i$, for $i \in \mathcal{X}$, to the closest multiple of $\alpha/|\mathcal{X}|$, this changes each fractional count by at most a $\alpha/|\mathcal{X}|$ additive factor, and so the rounded counts have $(3\alpha, \beta, 0)$ utility. Now we can treat the rounded fractional database (which has total weight 1), as an integer synthetic database of (polynomial) size at most $|\mathcal{X}|/\alpha$. □

Recall that in our application of Theorem 6.7 we defined $A$ to be the mechanism that adds Laplace noise with parameter $\rho/(\varepsilon_{\text{base}}/\sqrt{2k\log(1/\delta_{\text{base}})})$. We have $k$ draws, so by taking

$$\alpha' = \rho\sqrt{2k\log(1/\delta_{\text{base}})}(\log k + \log(1/\beta))$$

we have that $A$ is $(\alpha', \beta, 0)$-accurate. For the base generator we chose error $\alpha^2 = (\log|\mathcal{Q}|)/n$. If the output of the syntheticizer is too large, we subsample

$$n' = \frac{\log|\mathcal{Q}|\log(1/\beta)}{\alpha^2} = \frac{\log k \log(1/\beta)}{\alpha^2}$$

rows. With probability $1-\beta$ the resulting database maintains $O(\rho\sqrt{(\log|\mathcal{Q}|)/n} + (\sqrt{2k\log(1/\delta_{\text{base}})}/\varepsilon_{\text{base}})(\log k + \log(1/\beta))$-accuracy on all of the concepts simultaneously.

Finally, the base generator can fail if the choice of queries $S \in \mathcal{D}^k$ does not lead to good generalization. With the parameters we have chosen this occurs with probability at most $\beta$, leading to a total failure probability of the entire generator of $3\beta$.

**Theorem 6.8** (Base Generator for Fractional Linear Queries)**.** For any data universe $\mathcal{X}$, database size $n$, and class $\mathcal{Q} : \{\mathcal{X}^n \to \mathbb{R}\}$ of fractional linear queries (with sensitivity at most $1/n$), for any $\varepsilon_{\text{base}}, \delta_{\text{base}} > 0$, there exists an $(\varepsilon_{\text{base}}, \delta_{\text{base}})$-differentially private $(k, \lambda, 1/3, 3\beta)$-base synopsis generator for $\mathcal{Q}$, where

$$k = O\left(\frac{n\log(|\mathcal{X}|)\log(1/\beta)}{\log|\mathcal{Q}|}\right)$$

$$\lambda = O\left(\frac{\log(1/\beta)}{\sqrt{n}}\left(\sqrt{\log|\mathcal{Q}|} + \sqrt{\frac{log|\mathcal{X}|}{log|\mathcal{Q}|}}\cdot\frac{1}{\varepsilon_{\text{base}}}\right)\right).$$

The running time of the base generator is $\text{poly}(|\mathcal{X}|, n, \log(1/\beta), \log(1/\varepsilon_{\text{base}}))$.

The sampling bound used here is the same as that used in the construction of the SmallDB mechanism, but with different parameters. Here we are using these bounds for a base generator in a complicated boosting algorithm with a very small query set; there we are using them for a single-shot generation of a synthetic database with an enormous query set.

### 6.2.3 Assembling the ingredients

The total error comes from the choice of $\mu$ (see Equation 6.2) and $\lambda$, the accuracy parameter for the based generator.

Let us recall Theorem 6.1:

**Theorem 6.9** (Theorem 6.1)**.** Let $\mathcal{Q}$ be a query family with sensitivity at most $\rho$. For an appropriate setting of parameters, and with $T = \log|\mathcal{Q}|/\eta^2$ rounds, the algorithm of Figure 6.1 is an accurate and differentially private query-boosting algorithm:

1. When instantiated with a $(k, \lambda, \eta, \beta)$-base synopsis generator, the output of the boosting algorithm gives $(\lambda + \mu)$-accurate answers to *all* the queries in $\mathcal{Q}$ with probability at least $1 - T\beta$, where
$$\mu \in O(((\log^{3/2}|Q|)\sqrt{k}\sqrt{\log(1/\beta)}\rho)/(\varepsilon_{\text{sample}} \cdot \eta^3)). \quad (6.10)$$
2. If the base synopsis generator is $(\varepsilon_{\text{base}}, \delta_{\text{base}})$-differentially private, then the boosting algorithm is $((\varepsilon_{\text{sample}} + T \cdot \varepsilon_{\text{base}}), T(\beta + \delta_{\text{base}}))$-differentially private.

By Equation 6.7,
$$\varepsilon_{\text{sample}} \stackrel{\text{def}}{=} \sqrt{2kT\log(1/\beta)}(\alpha 4T\rho/\mu) + kT\left(\frac{\alpha 4T\rho}{\mu}\right)^2,$$
where $\alpha = (1/2)(\ln(1+2\eta)(1-2\eta)) \in O(1)$. We always have $T = (\log|\mathcal{Q}|)/\eta^2$, so substituting in this value into the above equation we see that the bound
$$\mu \in O(((\log^{3/2}|Q|)\sqrt{k}\sqrt{\log(1/\beta)}\rho)/(\varepsilon_{\text{sample}} \cdot \eta^3))$$
in the statement of the theorem is acceptable.

For the case of arbitrary queries, with $\eta$ a constant, we have

$$\lambda \in O\left(\frac{\rho}{\varepsilon_{\text{base}}}(\sqrt{n \log |\mathcal{X}| \log(1/\delta_{\text{base}})}(\log(n \log |\mathcal{X}|) + \log(2/\beta)))\right).$$

Now, $\varepsilon_{\text{boost}} = T\varepsilon_{\text{base}} + \varepsilon_{\text{sample}}$. Set these two terms equal, so $T\varepsilon_{\text{base}} = \varepsilon_{\text{boost}}/2 = \varepsilon_{\text{sample}}$, whence we can replace the $1/\varepsilon_{\text{base}}$ term with $2T/\varepsilon_{\text{boost}} = (\log |\mathcal{Q}|/\eta^2)/2\varepsilon_{\text{boost}}$. Now our terms for $\lambda$ and $\mu$ have similar denominators, since $\eta$ is constant. We may therefore conclude that the total error is bounded by:

$$\lambda + \mu \in \tilde{O}\left(\frac{\sqrt{n \log |\mathcal{X}|}\rho \log^{3/2} |\mathcal{Q}|(\log(1/\beta))^{3/2}}{\varepsilon_{\text{boost}}}\right).$$

With similar reasoning, for the case of fractional counting queries we get

$$\lambda + \mu \in \tilde{O}\left(\frac{\sqrt{\log |\mathcal{X}|} \log |\mathcal{Q}| \log(1/\beta)^{3/2}}{\varepsilon_{\text{boost}}\sqrt{n}}\right).$$

To convert to a bound for ordinary, non-fractional, counting queries we multiply by $n$ to obtain

$$\lambda + \mu \in \tilde{O}\left(\frac{\sqrt{n \log |\mathcal{X}|} \log |\mathcal{Q}| \log(1/\beta)^{3/2}}{\varepsilon_{\text{boost}}}\right).$$

## 6.3 Bibliographical notes

The boosting algorithm (Figure 6.1) is a variant of AdaBoost algorithm of Schapire and Singer [78]. See Schapire [77] for an excellent survey of boosting, and the textbook "Boosting" by Freund and Schapire [79] for a thorough treatment. The private boosting algorithm covered in this section is due to Dwork et al. [32], which also contains the base generator for linear queries. This base generator, in turn, relies on the syntheticizer of Dwork et al. [28]. In particular, Theorem 6.7 comes from [28]. Dwork, Rothblum, and Vadhan also addressed differentially private boosting in the usual sense.