Practical Privacy: The SuLQ Framework

Avrim Blum* Cynthia Dwork† Frank McSherry‡ Kobbi Nissim§

ABSTRACT

We consider a statistical database in which a trusted administrator introduces noise to the query responses with the goal of maintaining privacy of individual database entries. In such a database, a query consists of a pair (S, f) where S is a set of rows in the database and f is a function mapping database rows to $\{0, 1\}$. The true answer is $\sum_{i \in S} f(d_i)$, and a noisy version is released as the response to the query. Results of Dinur, Dwork, and Nissim show that a strong form of privacy can be maintained using a surprisingly small amount of noise – much less than the sampling error – provided the total number of queries is sublinear in the number of database rows. We call this query and (slightly) noisy reply the *SuLQ* (Sub-Linear Queries) primitive. The assumption of sublinearity becomes reasonable as databases grow increasingly large.

We extend this work in two ways. First, we modify the privacy analysis to real-valued functions f and arbitrary row types, as a consequence greatly improving the bounds on noise required for privacy. Second, we examine the computational power of the SuLQ primitive. We show that it is very powerful indeed, in that slightly noisy versions of the following computations can be carried out with very few invocations of the primitive: principal component analysis, k means clustering, the Perceptron Algorithm, the ID3 algorithm, and (apparently!) all algorithms that operate in the in the statistical query learning model [11].

*Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891. avrim@cs.cmu.edu.

†Microsoft Research, SVC, 1065 La Avenida, Mountain View CA 94043. dwork@microsoft.com.

‡Microsoft Research, SVC, 1065 La Avenida, Mountain View CA 94043. mcsherry@microsoft.com

§Department of Computer Science, Ben-Gurion University, P.O.B. 653 Be'er Sheva 84105, ISRAEL. Work done while at Microsoft Research, SVC. kobbi@cs.bgu.ac.il.

PODS 2005 June 13-15, 2005, Baltimore, Maryland.

1. INTRODUCTION

Suppose we have collected a large amount of very sensitive personal data. Suppose further that we are completely trustworthy, and we wish to release accurate statistical information about our population while provably maintaining the privacy of the individuals. We might even wish to allow other, possibly less trustworthy, parties ask and receive answers to statistical queries, again while provably protecting individuals' privacy. This is the problem we address.

For concreteness, let us (temporarily) define a statistical database as follows; our first definition will not preserve privacy. The database consists of some number n of rows. We are completely agnostic about the type of rows – they may be tuples of attributes, strings, or even pictures. A *query* is a function mapping rows to real number in the range $[0, 1]$. The *exact answer* to a query f is $\sum_{r=1}^{n} f(DB_r)$, where DB_r denotes row r of the database.

As the following example shows, this is completely non-private. Consider an employee database, where each row includes an employee name and salary. We now pose two queries: the first evaluates to one iff the employee draws salary at least X and the second evaluates to one iff the employee draws salary at least X and is not named Y. Clearly by subtracting the two sums we can determine if employee Y draws salary at least X. While this is itself a breach of privacy, for concreteness a few more queries could perform a binary search to determine the precise salary of employee Y. Note that in this example the functions aggregate very large sets, and still do not achieve privacy.

A natural approach, and one that has been explored by others in the 1980's (see also [3] for a survey), is to add random noise to the answer. This *output perturbation* technique, proposed, eg, by Leiss [14], seems to have been abandoned, possibly because the noise may be effectively cancelled by repeatedly posing the same question[1]. In a related vein Denning [5] suggested choosing a random subset of the database and applying the query to the subset; we discuss this variant presently.

Nonetheless, our approach does involve adding random noise.

[1]Although one could record queries and always give the same response when a query is repeated, there can be semantically equivalent but syntactically different queries, and it may be hard to decide whether two queries are semantically equivalent. Depending on the query language, the problem may even be undecidable.

A rigorous treatment of noise was initiated by Dinur and Nissim [6], who obtained (high) lower bounds for the amount of noise needed to maintain privacy against computationally unbounded, as well as computationally bounded adversaries. A careful analysis shows that the approach can be made to work by *bounding the number of queries* for the lifetime of the database [6, 7]. Extending these analyses, now for real-valued functions on arbitrary rows, we show, very roughly, that to maintain a given amount ϵ of information leakage under T queries with probability at least $1-\delta$, it suffices to add noise that is normally distributed with mean zero and variance $2T\log(1/\delta)/\epsilon^2$. This means that the amount of noise added is almost always $O(\sqrt{T\log(1/\delta)}/\epsilon)$. Interestingly, this distribution is *independent of the number n of rows*. Thus, as n gets large the statistics obtained become increasingly accurate. Compare this with the random-subset approach mentioned earlier. Consider a predicate that holds for qn rows of the database. If we choose a random subset of, say, cn rows, then the number of sampled rows satisfying the predicate will be of order $qcn \pm \Omega(\sqrt{cnq(1-q)})$. In contrast, using our approach, with overwhelming probability the response to the query will be $qn \pm O((T\log(1/\delta))/\epsilon)$.

A similar reasoning shows that when the noise is independent of n, we get privacy "for free". Consider a property that occurs in a p fraction of the population. If we view the database as a sample of the underlying population, then the number of rows in the database for which the property holds will be, with overwhelming probability, $pn \pm O(\sqrt{pn})$. However, due to sampling error we do expect deviation from the mean the order of $\Omega(\sqrt{pn})$. Since the noise added for privacy is independent of n, the amount of privacy-protecting noise is less than the sampling error. Dwork and Nissim [7] observed this for constant p, provided the total number of queries is just slightly sublinear in n (their work applied to a more restricted class of functions and used a slightly different noise distribution). Accordingly, they coined the term Sub-linear Queries database, or SuLQ database. Sublinearity becomes reasonable as databases grow large.

The results described show that SuLQ databases provide utility, as they allow for learning statistics of the input. In this paper, after extending previous analyses to handle real-valued functions of arbitrary row types, we ask what kinds of more sophisticated analyses of the data might be conducted in a privacy-preserving way through the SuLQ interface. We obtain algorithms for a rich collection of supervised and unsupervised functionalities, including approximations to principal component analysis, k-means clustering, perceptron, and the ID3 classification algorithm, while preserving privacy in a precise and rigorous sense.

Our analysis of these algorithms is straightforward: as the algorithms interact with the data only through the SuLQ primitive, they and the results that they compute provably maintain privacy. Whereas in prior work privacy had to be proved "from scratch", algorithms that execute using exclusively the SuLQ primitive enjoy provable and immediate privacy. The SuLQ primitive presents itself as the basis of a calculus of noisy computation, in which *any* algorithm can operate without the risk of intentional or accidental disclosure of private information about participants.

1.1 Additional Related Work

The problem of ensuring privacy in statistical databases has been studied extensively since the 1970's with mixed results. We focus here on some of the recent results, and refer the reader to [3] for an excellent survey of results on perturbation and other techniques for statistical disclosure control.

In 2000 Lindell and Pinkas [13] and Agrawal and Srikant [2] presented two approaches to privacy preserving datamining. Lindell and Pinkas constructed an efficient secure function evaluation protocol for the ID3 algorithm. The parties in their protocol collect private individual data, and want to share it to compute a classification tree. The protocol enables the parties to compute this classification tree, without leaking any information about the inputs they hold other than what can be learned from the output. Agrawal and Srikant demonstrated how to learn the probability distribution underlying some collection of data, in the presence of perturbation noise (introduced for maintaining privacy).

The research on secure function evaluation protocols almost completely ignores the question of which functionalities preserve privacy. For example, the the work of Lindell and Pinkas guarantees that no information beyond what is implied by the outcome of the ID3 algorithm is revealed, but it does not guarantee that this outcome itself preserves privacy, nor that it is superior in this respect to any other classification algorithm of similar quality. To our best knowledge, the only attempts to define such 'functional privacy' in the context of secure function evaluation was by Feigenbaum, Ishai, Malkin, Nissim, Strauss, and Wright [9] in defining private approximations, and similarly Halevi, Kushilevitz, Krauthgamer, and Nissim [10] in defining *almost* private approximations. Both definitions are strongly tied to the generic definition of secure function evaluation – they consider f to be functionally private with respect to g if f does not reveal any information beyond whatever g reveals; they do not say when is the information revealed by g 'private'.

The work of Agrawal and Srikant [2] rekindled the interest in input-perturbation techniques (here, a noisy version of the data is computed and released for arbitrary use and study by others). Subsequent work demonstrated that the privacy definition introduced in [2] is far too weak [1] and suggested alternative privacy definitions [4, 6, 7, 8] and perturbation techniques (see [7] for a short survey of the evolution of privacy definitions since [2]).

1.2 Paper Outline

After defining the notion of privacy (and privacy compromise) and the power of the adversary (Section 2), we review the definition of the SuLQ primitive and state our main theorem connecting information leakage, probability of privacy violation (i.e., of leaking more information than some specified threshold), and the distribution on privacy-preserving noise to be added to the exact answer to each query. We also discuss some properties of SuLQ privacy. Section 4 contains algorithms for noisy versions of a variety of supervised and unsupervised learning problems in which the data are only accessed via invocations of the SuLQ primitive. We prove our main theorem in Section 5.

2. DEFINITIONS

We model a database as an n-tuple $(d_1, d_2, \ldots, d_n)$ of elements drawn from an arbitrary domain D. The domain could be points in $\mathbb{R}^k$, text strings, images, or any other imaginable set of objects. In previous work, the elements d_i were assumed random and independent, so that revealing one to the adversary would not give information about another. We advance this approach by using *a priori* beliefs about the elements d_i, which we assume are independent. Beliefs are a way of reasoning about how an individual perceives an object that need not make any probabilistic assumptions about the data, but follow similar rules as probability, notably Bayes' law. Rather than assume the rows are drawn from independent probability distributions, we assume that the adversary holds beliefs about the rows that are similarly independent. If the reader is uncomfortable with beliefs (see [16], section 2.2 for a good introduction), they can simply assume that the data *is* drawn from a distribution without missing much, aside from the generality.

The intent of the independence assumption is to characterize what information is under the control of a given individual. Specifically, if there is information about a row that can be learned from other rows, this information is not truly under the control of that row. Even if the row in question were to sequester itself away in a high mountaintop cave, information about the row that can be gained from the analysis of other rows is still available to an adversary. It is for this reason that we focus our attention on those inferences that can be made about rows without the help of others.

For any predicate $f : D \to \{0, 1\}$ we let $p_0^{i,f}$ be the *a priori* belief that $f(d_i) = 1$ and $p_T^{i,f}$ be the *a posteriori* belief that $f(d_i) = 1$, given the answers to T queries, *as well as* all the values in all rows other than i: $d_{i'}$ for all $i' \neq i$. Note that because we assume independence among the rows of the database, $p_0^{i,f}$ is also the *a priori* belief that $f(d_i) = 1$ given all the values in all rows other than row i. In other words, seeing the other rows, before interacting with the database, cannot affect beliefs about d_i.

We define the monotonically-increasing 1-1 mapping $\text{conf} : (0, 1) \to \mathbb{R}$ as follows: $\text{conf}(p) = \log(p/(1-p))$. While a small additive change in $\text{conf}(p)$ implies a small additive change in p, the converse does not hold.

Our definition of privacy is based on bounding the additive increase from $\text{conf}(p_0^{i,f})$ to $\text{conf}(p_T^{i,f})$.

DEFINITION 1 ((ϵ, δ, T)-PRIVACY). *A database access mechanism $\mathcal{A}$ is (ϵ, δ, T)-private if for every set of independent a priori beliefs, for every data element index i, for every predicate $f : D \to \{0, 1\}$, and for every (possibly randomized) adversary making at most T queries,*

$$\Pr\left[\ \text{conf}(p_T^{i,f}) - \text{conf}(p_0^{i,f}) \ > \ \epsilon\ \right] \quad \leq \quad \delta\ .$$

The probability is taken over the randomness of the adversary as well as the database access mechanism.

To reiterate, our definition of privacy attempts to bound the information that the adversary can learn about a row that it could not learn in the absence of that row.

3. GENERAL SULQ DATABASES

The database access mechanism we consider is an extension of the SuLQ DB of [7] to continuous, real distributions. Here we use $N(0, R)$ to refer to a random number distributed according to a zero mean normal with variance $R = R(\epsilon, \delta, T)$.

> SuLQ Database Algorithm $\mathcal{A}(R)$
> Input: a query $(g : D \to [0, 1], S \subseteq [n])$.
>
> 1. Return $\sum_{i \in S} g(d_i) + N(0, R)$.

The main technical theorem of this paper is an extension of the previous privacy result of [7] from Boolean functions on the domain $\{0, 1\}^k$ to bounded functions on arbitrary domains. The theorem holds even if the domain of g is $[n] \times D$, but we do not exploit that fact here.

THEOREM 1. *For ϵ, δ, T with $\epsilon \leq 2\log(1/\delta)$, the SuLQ algorithm $\mathcal{A}(R)$ is (ϵ, δ, T)-private for $R \geq 2T\log(1/\delta)/\epsilon^2$.*

The result can be extended to $\epsilon > 2\log(1/\delta)$, but the requirement of R changes to $2T/\epsilon$. Clearly there is a monotonicity in privacy: any algorithm that is (ϵ, δ, T)-private is also (ϵ', δ, T)-private for $\epsilon' > \epsilon$. The only question is how quickly R is allowed to decay as ϵ increases.

The proof analyzes the *a posteriori* belief $p_j^{i,f}$ that $f(d_i) = 1$ given the answers to the first j queries $(a_1, \ldots, a_j)$ *and* the entire database except for the ith row. As the initial beliefs are assumed independent, this definition of $p_0^{i,f}$ is equivalent to our first definition of $p_0^{i,f}$ as the *a priori* beliefs; seeing other rows should not refine the beliefs about a given row. Our proof refines the martingale approach of [6] using the moment generating function, and is conducted in Section 5.

3.1 Some Remarks on Theorem 1

3.1.1 Noise is Independent of n.

The distribution on noise, $N(0, R)$ is independent of n, the number of rows in the database. As n increases, the accuracy of simple statistics increases. This is particularly exciting as several algorithms, for example k-means, PCA, and association rules, are agnostic to the size of the database; they tally up all rows satisfying particular properties and operate no differently if the database is 10x as large. The only change is that as the sums are performed over larger and larger sets, the negative impact of the added noise is marginalized to the point of inconsequence. This gives us some insight into the types of algorithms that are easily and gainfully adapted to our framework: those that use aggregate functions of the data as their basis, and are insensitive to the particulars of single elements.

3.1.2 Per-Row Analysis.

At the same time, our analysis provides a "per row" result, bounding the increase in confidence as a function of the number of queries the row participates in. If a query is executed over a subset of the rows, perhaps based on a

public attribute such as gender, only the participants in the query will risk confidence increase. It is important to emphasize that the privacy guarantees we provide are not dependent on the size of the database; individual privacy is not maintained by an abundance of peers, but rather on an individual level. Our methods can be applied to arbitrarily large or small databases, ensuring equivalent privacy in each but giving enhanced utility for larger databases.

3.1.3 Per-User Bounds on T

If different users A and B each make T queries to the database, then the theorem says that neither can violate privacy. However, even if they make their queries independently, without colluding, if after interacting with the database they were to analyze their combined query and response transcripts, the theorem offers no privacy guarantees. For this reason we think in terms of "turning off" the database after T queries *in the lifetime of the database*.

3.1.4 What is a Reasonable Change in Confidence?

This is subjective, so we discuss the effect of an (additive) change in confidence of 1. Because of the logarithm, this corresponds to a change in $p/(1-p)$ of a (multiplicative) factor of 2. When p_0 is $1/2$, this means $p_f = 2/3$. When $p_0 = .9$, however, $p_f \approx .945$; although a smaller net change in probability, in this case the chance of guessing incorrectly has been roughly *halved*.

Another way to examine how confidence changes with probability is to consider the derivative of $\text{conf}(p)$ with respect to p. We get $\text{conf}(p)' = 1/\ln 2 \cdot p \cdot (1-p)$. Hence, for $p \approx 1/2$ a change of Δp in p results in a comparable change in $\text{conf}(p)$, approximately $4\Delta p/\ln 2$. For $p \approx 0$ (and similarly for $p \approx 1$) we get that a change of Δp results in a much larger change in $\text{conf}(p)$, approximately $\Delta p/p$. That is, effecting even a small change in p can require an enormous change in $\text{conf}(p)$.

3.1.5 Immunity to Timing Attacks.

Recall that each query is a function and that functions must be evaluated to produce an answer. Cleverly constructed functions can reveal much in the amount of time it takes to execute. Imagine a function that typically returns zero, unless the input row happens to be a particular individual with a particular property, in which case it spends an enormous amount of time computing a very accurate estimate of the low order bits of π. In this case, the amount of time taken to execute reveals information about the participants: if the computation takes a while, the individual has the property, otherwise not.

Timing attacks are well-known in cryptography [12], and it is in general a hard problem to address them. Simply adding random time delays does *not* solve the problem in the usual cryptographic setting. However, it does here!

For a function f, let $t_f(row)$ denote the time it takes to compute $f(row)$. Just as we assume our functions have range $[0,1]$, we fix a maximum permitted time $t_{\max}$ for any row computation – if a computation takes longer, we terminate it and output 0. We then add a normally distributed delay with mean $t_{\max}R^{1/2}\log(T/\delta)$ and variance $t_{\max}R$. (This is small compared to the total amount of time that might be taken to evaluate the query.) Much as our analysis concludes that the summed output of functions clouded by noise preserves privacy, so too will the total execution time of the query, once clouded by an equivalent noise. (Strictly speaking we should also increase T to $2T$, since each query is essentially giving the adversary two values for each row: the $f(row)$ and $t_f(row)$.)

3.1.6 Do Beliefs Need to be True?

A commonly voiced concern is that the naive adversary, who arrives at the data set with many misconceptions, might learn an enormous amount about the participants in the database simply by being so wrong to begin with. Imagine an adversary who initially believes that nearly all people have blue hair, but after a single query immediately learns that this simply can not be the case. This adversary has arguably made a remarkable amount of progress in understanding the database, despite only issuing a single query. His belief about each individual has shifted from being assured that they each have blue hair, to being substantially less certain, what each individual might view as a substantial change in confidence and therefore a breach of privacy.

In essence, this adversary is engaging in *non-independent* reasoning about rows. By learning something about other rows, that not all of them have blue hair, he infers something new about an unseen row. We explicitly consider non-independent reasoning as a non-violation of privacy; information that can be learned about a row from sources other than the row itself is not information that the row could hope to keep private. Even if the row choses to absent itself and its hair from the database, this information would still be available to the adversary.

Philosophically, we distinguish between *facts of life*, such as the correlation between smoking and heart disease, and compromises of privacy. Facts of life are precisely what we *want* to learn from the database. Consider smoker X. The fact that smoking causes heart disease can be learned independent of whether X is or is not a member of the database: the adversary can run its own study to learn these things. Thus if learning this correlation changes the adversary's perception of whether X has or will develop heart disease, the change in perception would occur even if X were not present. We do not concern ourselves with privacy "violations" of this kind.

4. COMPUTATION WITH SULQ

The basic SuLQ operation – query and noisy reply – can be viewed as a noisy computational primitive that may be used to compute more advanced functions of the database than simple statistical queries. In this section we describe five examples of the power of the primitive. We reiterate that by basing these algorithms on the SuLQ access primitive we are immediately assured privacy, with no additional reasoning required.

In all of the examples below the rows of the database are drawn from $[0,1]^d$, although it should be apparent how to generalize many of the techniques to other domains. For example, any function that we apply to data could first transform the data into $[0,1]^d$ using a domain specific function.

For a query $f : D \to [0,1]$, we let $SuLQ(f)$ be the query invocation of $\mathcal{A}(R)$ with the function f and the set $S = \{1, \ldots, n\}$. For notational simplicity, we also consider SuLQ queries using functions that return multi-dimensional answers. Clearly, we could run a query for each output dimension independently, and, so long as we charge the dimensionality of the result against our allotted queries, we are simply shortening notation. We additionally use $N(0,R)^d$ to refer to a d-dimensional vector whose entries are each independent $N(0,R)$ random variables.

4.1 Singular Value Decomposition and PCA

Many powerful data mining and data filtering tasks make use of the singular value decomposition of an incidence matrix associated with the data. Given a $n \times d$ matrix A whose rows are the rows of the database, Latent Semantic Indexing, Principal Component Analysis, and many flavors of spectral clustering operate by projecting data elements (eg, rows) onto the space spanned by the top k right singular vectors of A, these being the top k eigenvectors of the matrix $A^T A$. Given the matrix $A^T A$, its eigenvectors can be easily computed using standard algorithms from numerical analysis.

Notice that as d_i is the ith row of A, we may write $A^T A$ as the sum over outer products $d_i^T d_i \in \mathbb{R}^{d \times d}$,

$$A^T A \quad = \quad \sum_i d_i^T d_i \; . \tag{1}$$

Casting the quantity of interest as a simple sum over the rows suggests the following natural SuLQ implementation:

SuLQ SVD(k):

1. (d^2 queries) Approximate $A^T A = \sum_i d_i^T d_i$ computing
$$C \quad = \quad SuLQ(f(d_i) := d_i^T d_i)$$
2. Compute and return the top k eigenvectors of C.

While C is not exactly $A^T A$, and therefore our computed eigenvectors are not exactly correct, eigenvectors are notoriously robust in the presence of independent, zero-mean noise. In fact, the normal error $N(0,R)^{d \times d}$ that we add is about the most benign form of error. Moreover, the magnitude of the noise that we add is roughly $O(\sqrt{R}/n)$, which can be terribly small as T is only d^2.

We remark that, using the techniques of [7] for vertically partitioned databases, this computation can be carried out even if each column of the database is stored in a separate, independent, SuLQ database.

4.1.1 Principle Component Analysis

PCA [15] is a related technique that uses the space spanned by the top k right singular vectors of the matrix A, with the mean of the rows, denoted μ, subtracted from each. These are the top k eigenvectors of the covariance matrix of A, equal to $\sum_i d_i^T d_i - \sum_i \mu^T \mu$. We can compute an accurate approximation $\overline{\mu}$ to μ with only d additional queries and arrive at a faithful approximation to the covariance matrix.

4.1.2 Use of Subspace Projections

While computing the space spanned by the top k eigenvectors of A may give us some information about the structure of the data in A, ultimately we would like to use it to filter actual data. Clearly we can not directly extract rows and project them onto this space, but we can make the space available for future functions. Given C, it is easy to modify any function f on the rows of the database to first project the rows onto the appropriate subspace before continuing with whatever functionality f originally intended.

Alternately, projection onto eigenvectors has been used in recommendation setting, noting that an incomplete or noisy preference vector d_i can be "cleaned" to a more accurate one via projection. While it is not possible for a row to query the database to learn what it should prefer, it is trivial for the database to return the eigenvectors to the participants, or any querier, and have them perform the projection themselves. Each receives the benefits of projecting their data onto an advantageous subspace, but the only information that is "leaked" comes from the SuLQ primitive, which is (ϵ, δ, T)-private.

4.2 k-Means Clustering

Given a collection of points $\{d_i\} \subset [0,1]^d$, it is natural to try to cluster the points so that each cluster contains points that are mutually proximate. One approach to solving this problem is the k-means algorithm, which maintains a set of k points ("means") in $[0,1]^d$, and forms clusters by associating each sample with the closest mean. Given a clustering of the points, the points minimizing the radii of each cluster are exactly the means of each cluster, suggesting the following iterative update rule:

***k*-Means**$(\mu_1, \ldots, \mu_k)$:

1. Partition the samples $\{d_i\}$ into k sets $S_1, \ldots, S_k$, associating each d_i with the nearest μ_j.
2. For $1 \le j \le k$, set $\mu'_j = \sum_{i \in S_j} d_i / |S_j|$, the mean of the samples associated with μ_j.

This update rule is typically iterated until some convergence criterion has been reached, or a fixed number of iterations have been applied.

While the first step seems unlikely to be implementable privately – computing the nearest mean of any one sample would breach privacy – the update rule's interface is feasible: we supply k points in $[0,1]^d$ and receive k new points in $[0,1]^d$. Each of the output points is a simple average over a subset of the rows defined by a simple predicate. Using some care in the definition of the query function f, we are able to approximate both the sum and count of these points, from which we can estimate the mean.

SuLQ *k*-Means$(\mu_i, \ldots, \mu_k)$:

1. (k queries): Approximate the number of points in each of the S_j, computing for $1 \le j \le k$
$$\overline{s}_j \quad = \quad SuLQ\left(f(d_i) := \left\{ \begin{array}{ll} 1 & \text{if } d_i \in S_j \\ 0 & \text{otherwise} \end{array} \right\} \right)$$

2. (kd queries): Approximate the sum of points in each of the S_j, computing for $1 \leq j \leq k$,

$$\overline{m}_j \quad = \quad SuLQ\left(f(d_i) := \left\{ \begin{array}{ll} d_i & \text{if } d_i \in S_j \\ 0 & \text{otherwise} \end{array} \right\}\right)$$

3. Update each mean, setting $\overline{\mu}'_i = \overline{m}_j/\overline{s}_j$ for $1 \leq j \leq k$.

As long as the number of points in each cluster greatly exceeds $R^{1/2}$, we expect $\overline{s}_j$ to be a good estimate of $s_j = |S_j|$, and the computed $\overline{\mu}'_j$ to accurately estimate the μ'_j of the non-private approach. Formally,

LEMMA 2. *For each $1 \leq j \leq k$, if $|S_j| \gg R^{1/2}$ then with high probability*

$$\|\overline{\mu}'_j - \mu'_j\| \quad \textit{is} \quad O\left((\|\mu'_j\| + d^{1/2})R^{1/2}/\,|S_j|\right)$$

PROOF. Notice that the SuLQ algorithm averages the exact same set of rows as the exact algorithm, with two sources of error: the inaccuracy of the $\overline{s}_j$ and the error in the summation $\sum_{i \in S_j} d_i/\overline{s}_j$. Let $s_j = |S_j|$ and let n_j denote the difference between $\sum_{i \in S_j} d_i$ and $\overline{s}_j\overline{\mu}'_j$. We must bound

$$\begin{aligned} \|\overline{\mu}'_j - \mu'_j\| &= \|(\sum_{i \in S_j} d_i + n_j)/\overline{s}_j - \sum_{i \in S_j} d_i/s_j\| \\ &= \|(1/\overline{s}_j - 1/s_j)\sum_{i \in S_j} d_i + n_j/\overline{s}_j\| \\ &\leq |(s_j - \overline{s}_j)/\overline{s}_j| \cdot \|\mu'_j\| + \|n_j/\overline{s}_j\| \end{aligned}$$

From our assumption that $|S_j| \gg R^{1/2}$, we get that with high probability $|(s_j - \overline{s}_j)/\overline{s}_j|$ is $O(R^{1/2}/|S_j|)$ and $\|n_j/\overline{s}_j\|$ is $O((dR)^{1/2}/|S_j|)$. The lemma follows. □

4.3 The Perceptron Algorithm

Given a collection of rows d_i and labels $\ell_i \in \{-1, +1\}$, a *linear threshold* is a vector w such that for all i, $\langle d_i, w\rangle \cdot \ell_i > 0$. That is, the sign of the label ℓ_i agrees with that of the inner product $\langle d_i, w\rangle$, and the hyperplane orthogonal to w thus separates positively labeled instances from negatively labeled ones.

For simplicity, we assume that all labels are $+1$, permitted by multiplying each row d_i by its label. The property $\langle d_i, w\rangle \cdot \ell_i > 0$ is clearly equivalent to the property $\langle d_i\ell_i, w\rangle \cdot 1 > 0$

The perceptron algorithm is useful for finding a separator when one is known (or assumed) to exist. It operates by repeatedly incorporating the misclassified samples into its estimate of w:

Perceptron(w):

1. As long as there exists a j such that $\langle d_j, w\rangle < 0$,
   (a) Set $w = w + d_j$.

We will be unable to privately learn of the misclassification of a specific row, much less select it out for incorporation into w. However, the proof of convergence for the perceptron algorithm relies only on the repeated incorporation of misclassified points, and does not require that the points actually exist in the data set.Instead, will synthesize an aggregate misclassified point and incorporate it, as in the following algorithm:

SuLQ Perceptron($w_0, \overline{s}_0$):

1. For $j = 1, 2, \ldots$, repeating so long as $\overline{s}_j \gg R^{1/2}$
   (a) (1 query) Count the misclassified rows,

$$\overline{s}_j = SuLQ\left(f(d_i) := \left\{ \begin{array}{ll} 1 & \text{if } \langle d_i, w_j\rangle \leq 0 \\ 0 & \text{otherwise} \end{array} \right\}\right)$$

   (b) (d queries) Synthesize a misclassified vector,

$$\overline{u}_j = SuLQ\left(f(d_i) := \left\{ \begin{array}{ll} d_i & \text{if } \langle d_i, w_j\rangle \leq 0 \\ 0 & \text{otherwise} \end{array} \right\}\right)$$

   (c) Set $\overline{v}_j = \overline{u}_j/\overline{s_j}$ and $w_{j+1} = w_j + \overline{v}_j$.

This process incorporates into w_j a noisy average of all points that are currently misclassified. If there are not sufficiently many, the algorithm stops, as we no longer expect the aggregates to reflect substance rather than noise. This termination occurs only once the bulk of the points are correctly classified.

THEOREM 3. *If there exists a unit vector w' and scalar δ such that for all i, $\langle w', d_i\rangle \geq \delta$ and for all j, $\delta \gg (dR)^{1/2}/\overline{s}_j$ then with high probability the algorithm terminates in at most $32 \max_i \|d_i\|^2/\delta$ rounds.*

PROOF. The proof we use is not new, aside from the technical issue of analyzing the error returned by the SuLQ primitive. The proof is by contradiction: we will show that in each round j the inner product $\langle w', w_j\rangle$ increases by more than $\|w_j\|$. However, as $\langle w', w_j\rangle \leq \|w'\|\|w_j\| = \|w_j\|$, this growth can not continue forever, otherwise $\langle w', w_j\rangle$ would overtake $\|w_j\|$. Therefore, there is a bound (which we compute) on the number of iterations that are applied.

We let S_j be the set of vectors misclassified by w_j in the jth round, and let $v_j = \sum_{i \in S_j} d_i/\overline{s}_j$ be the actual sum of misclassified vectors. We use $n_j = v_j - \overline{v}_j$ for the error introduced by the SuLQ query, divided by $\overline{s}_j$

During any iteration j, w_j will be incremented by a collection of vectors $\sum_{i \in S_j} l_i d_i/\overline{s}_j$ that are misclassified by w_j, plus a vector of random noise n_j.

$$\begin{aligned} \langle w', w_{j+1}\rangle &= \langle w', w_j + \sum_{i \in S_j} d_i/\overline{s}_j + n_j\rangle \\ &= \langle w', w_j\rangle + \langle w', \sum_{i \in S_j} d_i/\overline{s}_j\rangle + \langle w', n_j\rangle \end{aligned}$$

The distribution on n_j ensures that with high probability $\|\langle w', n_j\rangle\| \ll \delta/4$. Recalling that $\langle w', d_i\rangle \geq \delta$ for all i, we

are left with

$$\begin{aligned}\langle w', w_{j+1}\rangle &\geq \langle w', w_j\rangle + \langle w', \sum_{i\in S_j} d_i/\overline{s}_j\rangle - \delta/4\\ &\geq \langle w', w_j\rangle + \delta|S_j|/\overline{s}_j - \delta/4\\ &\geq \langle w', w_j\rangle + \delta/4.\end{aligned}$$

From this, collecting the contribution of each round we establish that with high probability

$$\langle w', w_j\rangle \geq j\delta/4 \quad (2)$$

On the other hand, the length of w is evolving slowly.

$$\|w_{j+1}\|^2 = \langle (w_j + v_j + n_j), (w_j + v_j + n_j)\rangle \quad (3)$$

Expanding this inner product, we see that it is equal to

$$\langle w_j, w_j\rangle + \langle v_j, v_j\rangle + \langle n_j, n_j\rangle + 2\langle (w_j + v_j), n_j\rangle + 2\langle w_j, v_j\rangle \quad (4)$$

As each vector v_j that we incorporate into w_j has negative projection onto it, we have that $\langle w_j, v_j\rangle \leq 0$, and therefore

$$\|w_{j+1}\|^2 \leq \|w_j\|^2 + \|v_j\|^2 + \|n_j\|^2 + 2\langle (w_j + v_j), n_j\rangle \quad (5)$$

With the exception of the $2\langle (w_j + v_j), n_j\rangle$ term, all terms in Eq. (5) are never negative. Bounding $\|v_j\|^2 + \|n_j\|^2$ by $(\max_i \|d_i\|^2 + \|n_j\|^2)$, we get that in a round r where $\sum_{j\leq r}\langle (w_j + v_j), n_j\rangle$ is negative

$$\|w_r\|^2 < \sum_{j\leq r}(\max_i \|d_i\|^2 + \|n_j\|^2) \quad (6)$$

We have assumed that for all i, $\langle w', d_i\rangle \geq \delta$, implying that $\|d_i\| \geq \delta$. On the other hand, our assumption on $|S_j|$ ensures that with high probability $\|n_j\| \ll \delta/4 \leq \|d_i\|$. For any round r for which $\sum_j \langle (w_j + v_j), n_j\rangle$ is negative, combining Eq. (2) and Eq. (6) we see that

$$\delta r/4 \leq \langle w', w\rangle \leq \|w\| < (2r \max_i \|d_i\|)^{1/2}$$

constraining $r < 32 \max_i \|d_i\|/\delta^2$, and hence bounding the number of iterations.

Note that if $\|w_j\|$ shrinks the bounds on running time are improved. On the other hand, if $\|w_j\|$ does not shrink we expect $\sum_j \langle (w_j + v_j), n_j\rangle$ to be negative again in round r' not much after r steps because the $\langle (w_j + v_j), n_j\rangle$ terms are each independent, zero mean normals. □

4.4 ID3 Classifiers

Let $\mathcal{A} = A_1, \ldots, A_d$ be a collection of categorical attributes and let $\mathcal{T} = \{T_1, \ldots, T_n\} \subseteq A_1 \times \cdots \times A_d$ be a collection of transactions over these attributes. Each transaction T_i is assigned a label $\ell_i \in L$ that we wish to predict given only the transaction attributes. The ID3 algorithm, introduced by Quinlan [17], is a heuristic for constructing a decision tree classifier, that is, a rooted tree where each internal node is assigned an attribute A in $A_1, \ldots, A_d$, and its out-degree corresponds to the number of possible values A may assume. Leaves are assigned a value in L. The prediction made by the decision tree on a transaction T is the leaf value reached by starting at the root, following the path that agrees with the values assigned to the corresponding attributes in T.

The ID3 tree is constructed starting from the root in a recursive manner. The algorithm chooses a "best attribute" A to be put at the root, that "best classifies" the transaction set $\mathcal{T}$. The transaction set is partitioned by A, and the algorithm is then applied recursively (without the attribute A). Recursion stops either (i) when the classification of a partition is consistent (all transaction in it have the same value for the attribute which is assigned to the corresponding leaf), or (ii) when the attribute set becomes empty (then, the leaf value is determined according to a majority vote). For simplicity of presentation, we assume that each attribute (including L) may take values in $[t]$. We use $T[A]$ for the value of attribute A in transaction T, the subset of $\mathcal{T}$ for which A takes the value j is denoted $\mathcal{T}[A = j]$.

ID3$(\mathcal{A}, \mathcal{T})$:

1. If $\mathcal{A} = \emptyset$: Return a leaf whose value is the majority vote on the labels of transactions in $\mathcal{T}$.
2. If $T[L] = \ell$ for all $T \in \mathcal{T}$: Return a leaf whose value is ℓ.
3. Determine the attribute A that "best classifies" $\mathcal{T}$ (see below).
4. For $j \in [t]$, recursively apply the ID3 algorithm on inputs $((\mathcal{A} \setminus A), \mathcal{T}[A = j])$.
5. Return a tree with a root node labeled A and edges labeled $1, \ldots, t$ going to the trees obtained in the corresponding recursive calls to the ID3 algorithm.

To complete the description we need to specify how to determine the attribute A. The entropy of the label attribute is defined as

$$H_L(\mathcal{T}) = -\sum_{k=1}^{t}(|\mathcal{T}[L = k]|/|\mathcal{T}|)\log(|\mathcal{T}[L = k]|/|\mathcal{T}|).$$

Given attribute A with possible values $a_1, \ldots, a_t$ we have

$$H_{L|A}(\mathcal{T}) = \sum_{j=1}^{t}(|\mathcal{T}[A = j]|/|\mathcal{T}|) \cdot H_L(\mathcal{T}[A = j]).$$

The information gain of attribute A is defined to be

$$H_L(\mathcal{T}) - H_{L|A}(\mathcal{T}).$$

The attribute A that exhibits the highest information gain is chosen in step 3 of the ID3 algorithm. Observe that to figure out the "best attribute" one need only maximize

$$\begin{aligned}V_A &= |\mathcal{T}| \cdot H_{L|A}(\mathcal{T})\\ &= \sum_{j=1}^{t}\sum_{k=1}^{t}|\mathcal{T}[A = j \wedge L = k]| \cdot \log\frac{|\mathcal{T}[A = j \wedge L = k]|}{|\mathcal{T}[A = j]|}.\end{aligned}$$

Using the SuLQ framework, we will be limited in the accuracy of computing $H_{L|A}(\mathcal{T})$. Hence, we will settle for an approximation to the ID3 algorithm, that picks an attribute A whose gain is not more than Δ away from the "best attribute" for some suitably chosen constant Δ (this approach is also followed also in [13], but for other reasons). Another

limitation is that we will not be able to meaningfully recurse with the ID3 algorithm with small transaction sets. In the following, the transaction set $\mathcal{T}$ used in the recursive call to the ID3 algorithm is expressed as a conjunction of terms of the the form $(A = j)$. At the top level of the recursion, the conjunction contains no terms (and hence evaluates to TRUE), corresponding to $\mathcal{T}$ containing all transactions.

SuLQ ID3$(\mathcal{A}, \mathcal{T})$:

1. Let $N_{\mathcal{T}} = SuLQ(f(d_i) := 1 \text{ if } \mathcal{T}(d_i))$, and for $j \in [t]$ let $N_j = SuLQ(f(d_i) := (\mathcal{T} \wedge (L = j))\,(d_i))$.
2. If $\mathcal{A}$ contains only A_1, return a leaf labeled j that maximizes N_j.
3. If $N_{\mathcal{T}} < \gamma\epsilon$ (where $\epsilon = (R\log(1/\delta))^{1/2}$ and $\gamma = O(t^2/\Delta)$), return a leaf labeled j that maximizes N_j.
4. For every attribute A:
   (a) Let $N_j^A = SuLQ(f(d_i) := (\mathcal{T} \wedge (A = j))$.
   (b) Let $N_{j,k}^A = SuLQ(f(d_i) := (\mathcal{T} \wedge (A = j) \wedge (L = k))$.
5. Choose the attribute $\bar{A}$ that maximizes
$$\bar{V}_A = \sum_{j=1}^{t}\sum_{k=1}^{t} N_{j,k}^A \cdot \log \frac{N_{j,k}^A}{N_j^A}$$
where terms for which $N_{j,k}^A$ or N_j^A are smaller than $N_{\mathcal{T}}/\gamma$ are skipped.
6. For $i \in [t]$, recursively apply the ID3 algorithm on inputs $(\mathcal{A} \setminus \bar{A}), \mathcal{T}_i \wedge (\bar{A} = i)$.
7. Return a tree with a root node labeled $\bar{A}$ and edges labeled 1 to t going to the trees obtained in the corresponding recursive calls to the ID3 algorithm.

LEMMA 4. *The gain of $\bar{A}$ differs from the maximum gain by Δ with probability $1 - O(dt^2\delta)$.*

PROOF. Note that with probability $1 - O(dt^2\delta)$ all estimates from the SuLQ primitive are within error ϵ. Hence, we assume this is the case.

There are two contributions to $\bar{V}_A - V_A$. One comes from skipped terms in the sum, these each contribute at most a $N_{j,k}^A + \epsilon = O(|\mathcal{T}|\Delta/t^2)$, and hence form a total of $O(|\mathcal{T}|\Delta)$.

The second contribution to the difference comes from the noise, of magnitude ϵ, added in N_j^A and $N_{j,k}^A$ (we use the notation τ_j for $|\mathcal{T}[A = j]|$ and $\tau_{j,k}$ for $|\mathcal{T}[A = j \wedge L = k]|$):

$$\begin{aligned} N_{j,k}^A \log \frac{N_{j,k}^A}{N_j^A} &= \\ &= (\tau_{j,k} \pm O(\epsilon)) \cdot \log \frac{\tau_{j,k} \pm O(\epsilon)}{\tau_j \pm O(\epsilon)} \\ &= (\tau_{j,k} \pm O(\epsilon)) \cdot \left(\log \frac{\tau_{j,k}}{\tau_j} + \log \frac{1 \pm O(\epsilon)/\tau_{j,k}}{1 \pm O(\epsilon)/\tau_j} \right) \\ &= (\tau_{j,k} \pm O(\epsilon)) \cdot \left(\log \frac{\tau_{j,k}}{\tau_j} \pm O(\epsilon)(1/\tau_{j,k} + 1/\tau_j) \right) \end{aligned}$$

Note that the ratio within the log is bounded by constants (as we skip the other terms in Step 5), and that the terms $O(\epsilon)/\tau$ are at most constants, hence we get:

$$N_{j,k}^A \log \frac{N_{j,k}^A}{N_j^A} = \tau_{j,k} \log \frac{\tau_{j,k}}{\tau_j} \pm O(\epsilon)\ .$$

The total contribution here is hence $O(t^2\epsilon)$ which again evaluates to $O(|\mathcal{T}|\Delta)$, as needed. □

4.5 The Statistical Queries Learning Model

The Statistical Query model, proposed by Kearns in [11], is a framework for examining statistical algorithms executed on samples drawn independently from an underlying distribution. In this framework, an algorithm repeatedly specifies a predicate p and an accuracy τ, and is returned the expected fraction of samples satisfying p to within additive error τ. Conceptually, the framework models drawing a sufficient number of samples so that the observed count of samples satisfying p is a good estimate of the actual expectation.

The statistical query model is most commonly used in the computational learning theory community, where the goal is typically to learn a "concept", a predicate on the data, to within a certain degree of accuracy. Formally, an algorithm δ-learns a concept c if it produces a predicate such that the probability of misclassification under the latent distribution is at most $1 - \delta$.

We will now see that any concept that is learnable in the statistical query model is privately learnable using the equivalent algorithm on a SuLQ database. The emulation of the Statistical Query primitive is rather straightforward: we must execute a sufficient number of queries so that we are assured that the accuracy is within the alloted accuracy parameter τ. The efficiency of the learning algorithm, measured by number and accuracy of queries, will determine the size of the database required to privately learn the concept.

SuLQ STAT(p, τ):

1. Initialize $tally = 0$.
2. Repeat $t = \lceil R/\tau n^2 \rceil$ times
   (a) Set $tally = tally + SuLQ(f(d_i) := p(d_i))$
3. Return $tally/tn$.

THEOREM 5. *For any algorithm that δ-learns a class F using at most q statistical queries of accuracy $\{\tau_1, \ldots, \tau_j\}$, the algorithm can δ-learn F on a SuLQ database of n elements, provided that*

$$n^2 \quad \geq \quad \frac{R\,\log(q/\delta)}{T - q} \times \sum_{j \leq q} 1/\tau_j$$

PROOF. The repetition in Step 2 reduces the variance, so that each emulated execution of $STAT(p, \tau)$ results in an answer that looks like $\sum_i p(d_i)/n + N(0, \tau')$ for some $\tau' \leq \tau$. If we augment each of the accuracies for the q queries by a factor of $1/\log(q/\delta)$, we ensure that the probability that any exceeds their associated τ_i is at most δ.

With this understood, we now need to determine how large a database we require to ensure privacy, or rather to ensure that the SuLQ primitive does not prematurely terminate the algorithm. An execution of a learning algorithm determines a number of SuLQ queries that must be performed, captured precisely by the set of accuracies $\{\tau_1, \ldots \tau_j\}$ required by the STAT queries of the algorithm.

$$T \geq \sum_{j \leq q} \lceil R \log(q/\delta)/\tau_j n^2 \rceil \quad (7)$$

$$\geq q + R \log(q/\delta)/n^2 \times \sum_{j \leq q} 1/\tau_j \quad (8)$$

From this we determine that for fixed R, T all is well so long as n satisfies

$$n^2 \geq \frac{R \log(q/\delta)}{T-q} \times \sum_{j \leq q} 1/\tau_j. \quad (9)$$

□

As a brief word, if we view $R = 2T \log(1/\delta)/\epsilon^2$, imagine that $T > 2q$, and conflate the two interpretations of δ, the above bound becomes

$$n \geq 2 \log(T/\delta)/\epsilon \times \left(\sum_{j \leq q} 1/\tau_j \right)^{1/2}$$

which is eminently reasonable: the τ terms characterize the query "cost" of the algorithm, and the number of samples required is only larger by a $\log(T/\delta)/\epsilon$ factor.

5. PROOF OF PRIVACY THEOREM

The privacy we offer is a guarantee that the confidence in any predicate applied to a row of the database will not increase substantially over the course of T arbitrary queries. Our approach is to first view the change in confidence as a sum of independent random variables, and then use a moment generating function argument to bound the probability that it greatly exceeds its mean. We also include a discussion of a modified scheme that gives deterministic guarantees on privacy, at the expense of increased variance.

In our analysis, we use a_j to represent the random value that is the result of the jth query. This value is equal to $f_j(d_i) + n_j$, where f_j is the query function in round j, d_i is the value of the ith row, and n_j is the normally distributed noise that the SuLQ primitive adds. We also use the vector $\mathbf{a}_j$ to represent the random variable $(a_1, \ldots, a_j)$.

5.1 The Evolution of Confidence as a Sum

Our first step is to convert the confidence at any point in time into a sum, starting from the initial confidence and with terms contributed by the responses to each of the T queries. We convert the *a posteriori* beliefs into joint beliefs, relying on the fact that the appropriate scaling is equivalent for the numerator and denominator.

$$\frac{b(f(d_i) = 1 | \mathbf{a}_j)}{b(f(d_i) = 0 | \mathbf{a}_j)} = \frac{b(f(d_i) = 1 \wedge \mathbf{a}_j)}{b(f(d_i) = 0 \wedge \mathbf{a}_j)} \quad (10)$$

Decomposing the numerator into the integral over the subset $D_1 \subseteq D$ of elements x satisfying f, to get

$$\frac{\int_{D_1} b(\mathbf{a}_j \wedge d_i = x) dx}{\int_{D_0} b(\mathbf{a}_j \wedge d_i = x) dx} \quad (11)$$

The law of conditional probability lets us extract $b(d_i = x)$

$$\frac{\int_{D_1} b(\mathbf{a}_j | d_i = x)\, b(d_i = x) dx}{\int_{D_0} b(\mathbf{a}_j | d_i = x)\, b(d_i = x) dx} \quad (12)$$

As each a_j is independent of $a_1, \ldots, a_{j-1}$ when conditioned on d_i, we may extract $b(a_j | d_i = x)$ from the first term.

$$\frac{\int_{D_1} b(a_j | d_i = x)\, b(\mathbf{a}_{j-1} | d_i = x)\, b(d_i = x) dx}{\int_{D_0} b(a_j | d_i = x)\, b(\mathbf{a}_{j-1} | d_i = x)\, b(d_i = x) dx} \quad (13)$$

As the nature of the perturbation is clear, $b(a_j | d_i = x)$ can be transformed into the probability $p(a_j | d_i = x)$,

$$\frac{\int_{D_1} p(a_j | d_i = x)\, b(\mathbf{a}_{j-1} | d_i = x)\, b(d_i = x) dx}{\int_{D_0} p(a_j | d_i = x)\, b(\mathbf{a}_{j-1} | d_i = x)\, b(d_i = x) dx} \quad (14)$$

At this point we observe that for any functions y_x, z_x if we divide $\int_x y_x z_x dx$ by $\int_x z_x dx$, it simply becomes a convex combination of the y_x terms.

$$\int_x y_x z_x dx = \int_x \left(\frac{z_x}{\int_x z_x dx} \right) y_x dx \int_x z_x dx$$

The coefficients $z_x / \int_x z_x dx$ clearly integrate to one, and, critically, are independent of the y_x terms. Applying this observation to the above equation, choosing $y_x = p(a_j | d_i = x)$, and letting w_x be the convex coefficients, we get

$$\frac{\int_{D_1} w_x p(a_j | d_i = x) dx \int_{D_1} b(\mathbf{a}_{j-1} | d_i = x) b(d_i = x) dx}{\int_{D_0} w_x p(a_j | d_i = x) dx \int_{D_0} b(\mathbf{a}_{j-1} | d_i = x) b(d_i = x) dx} \quad (15)$$

Recalling the equivalence of equation (12) to the initial ratio of beliefs, we rewrite the second ratio as the ratio of beliefs given evidence $\mathbf{a}_{j-1}$

$$\frac{\int_{D_1} w_x p(a_j | d_i = x) dx}{\int_{D_0} w_x p(a_j | d_i = x) dx} \times \frac{b(f(d_i) = 1 | \mathbf{a}_{j-1})}{b(f(d_i) = 0 | \mathbf{a}_{j-1})} \quad (16)$$

If we repeat this process, we can write our ratio of beliefs as a product over the changes in each of the T steps,

$$\prod_j \left(\frac{\int_{D_1} w_x p(a_j | d_i = x) dx}{\int_{D_0} w_x p(a_j | d_i = x) dx} \right) \times \frac{b(f(d_i) = 1)}{b(f(d_i) = 0)} \quad (17)$$

As we are actually considering the logarithms of the ratio of beliefs, we see that the change in confidence from our initial confidence $\log(b(f(d_i) = 1)/b(f(d_i) = 0))$ equals

$$\sum_j \log \left(\frac{\int_{D_1} w_x p(a_j | d_i = x) dx}{\int_{D_0} w_x p(a_j | d_i = x) dx} \right) \quad (18)$$

Each term in this sum is independent of the previous ones, once we condition on the previous answers. The only dependence otherwise is that the random events $\mathbf{a}_{j-1}$ will determine the coefficients w_x for the jth term. We will ignore any structure in the w_x othen than the fact that they integrate to one.

5.2 Jensen's Inequality and Moment Bounds

In this subsection we derive concentration bounds on the change in confidence after T steps, using a standard technique from large deviation theory: bounding the moment generating function $E[\exp(sX)]$ for a random variable X.

Before proceeding, we introduce another classical tool in the analysis of random variables, Jensen's inequality, which relates the average of a convex function applied at several points to the function applied at the average of the input points.

THEOREM 6 (JENSEN). *For any function f, probability measure $p(x)$ (ie: $\int_x p(x)dx = 1$), and convex fenction ψ,*

$$\int_x p(x)\psi(f(x))dx \quad \geq \quad \psi\left(\int_x p(x)f(x)dx\right)$$

Conceptually, it is always greater to average the output of a convex function ψ at several points than to first average the points and then apply ψ. This result can easily be seen by drawing a convex curve and drawing the line between any two points, which will lie entirely above the curve. We will be specifically interested in the functions $\exp(y)$ and y^s for $s \geq 1$, on the positive reals, which are both convex.

Recall that the probability density function that we have chosen is a normal distribution with variance R, so that

$$p(a_j|d_i = x) \quad \propto \quad \exp(-(a_j - f_j(x))^2/2R) \tag{19}$$

As such, let's look at the random variable

$$X_j \quad \equiv \quad \log\left(\frac{\int_{D_1} w_x \exp\left(-(a_j - f_j(x))^2/2R\right) dx}{\int_{D_0} w_x \exp\left(-(a_j - f_j(x))^2/2R\right) dx}\right) \tag{20}$$

and its moment generating function $E[\exp(sX)]$, equal to

$$E\left[\left(\frac{\int_{D_1} w_x \exp\left(-(a_j - f_j(x))^2/2R\right) dx}{\int_{D_0} w_x \exp\left(-(a_j - f_j(x))^2/2R\right) dx}\right)^s\right] \tag{21}$$

Applying Jensen's inequality to the denominator, noting that $\exp(y)$ is convex,

$$\leq \; E\left[\left(\frac{\int_{D_1} w_x \exp\left(-(a_j - f_j(x))^2/2R\right) dx}{\exp\left(-\int_{D_0} w_x (a_j - f_j(x))^2 dx/2R\right)}\right)^s\right] \tag{22}$$

If we expand out the squares, cancel the a_j^2 terms, and let $c_1 = \int_{D_0} w_x f_j(x)dx$ and $c_2 = \int_{D_0} w_x f_j(x)^2 dx$, we arrive at

$$= \; E\left[\left(\int_{D_1} w_x \frac{\exp\left(2a_j f_j(x) - f_j(x)^2)/2R\right)}{\exp\left((2a_j c_1 - c_2)/2R\right)} dx\right)^s\right] \tag{23}$$

We now restrict our attention to the case of $s \geq 1$, noting that $f(y) = y^s$ is convex-up on the positive real line for such values of s. We then apply Jensen's inequality again, extracting the integral over x from the powering by s.

$$\leq \; E\left[\int_{D_1} w_x \left(\frac{\exp\left(2a_j f_j(x) - f_j(x)^2)/2R\right)}{\exp\left((2a_j c_1 - c_2)/2R\right)}\right)^s dx\right] \tag{24}$$

Reordering the integral and expectation:

$$= \; \int_{D_1} w_x E\left[\left(\frac{\exp\left(2a_j f_j(x) - f_j(x)^2)/2R\right)}{\exp\left((2a_j c_1 - c_2)/2R\right)}\right)^s\right] dx \tag{25}$$

This expectation is simply a function of x, and there is an x' that maximizes it. Inserting that maximizer allows us to dispose of the integral over x,

$$\leq \; E\left[\left(\frac{\exp\left(2a_j f_j(x') - f_j(x')^2)/2R\right)}{\exp\left((2a_j c_1 - c_2)/2R\right)}\right)^s\right] \tag{26}$$

Recalling that $a_j = n_j + f_j(d_i)$, we can simplify the above by extracting all terms that do not depend on n_j. Letting $\mu = (2f_j(d_i)(f_j(x') - c_1) - (f_j(x')^2 - c_2))/2R$ and $\alpha = f_j(x') - c_1$,

$$E[\exp(sX_j)] \quad \leq \quad E\left[\exp\left(n_j \alpha s/R\right)\right] \exp(s\mu) \tag{27}$$

Notice that $\alpha = f_j(x') - c_1$ is simply a scalar of magnitude at most one. As such, the random variable in the exp is simply a normal random variable with variance at most s^2/R, from which we bound

$$E[\exp(sX_j)] \quad \leq \quad \exp(s^2/2R) \exp(s\mu) \tag{28}$$

Finally, notice that we can rewrite $-\mu$ as

$$\left((f_j(d_i) - f_j(x'))^2 + \int_{D_0} w_x (f_j(d_i) - f_j(x))^2 dx\right)/2R \tag{29}$$

which lives in $[-1/2R, +1/2R]$, bounding $\mu \leq 1/2R$.

5.3 Applying Markov's Inequality

We use the standard concentration trick of applying Markov's inequality to the moment generating function $E[\exp(sX)]$ for an intelligently chosen s. Letting $X \equiv \sum_j X_j$ be the total change in confidence, the monotonicity of exp gives

$$Pr[X > x + T/2R] \; \leq \; E[\exp(sX)]/\exp(s(x + T/2R)) \tag{30}$$

As the X_j are independent (once conditioned on $\mathbf{a}_{j-1}$), we expand the expected product into a product of expectations.

$$\leq \; \prod_j E[\exp(sX_j)]/\exp(s(x + T/2R)) \tag{31}$$

Applying the bound of equation (28) and simplifying,

$$Pr[X > x + T/2R] \quad \leq \quad \exp(s^2 T/2R - sx) \tag{32}$$

We now take $s = xR/T$, constraining $R \geq T/x$ from the constraint that $s \geq 1$.

$$Pr[X > x + T/2R] \quad \leq \quad \exp(-x^2 R/2T) \tag{33}$$

Taking $R = 2T\log(1/\delta)/\epsilon^2$, which is at least T/ϵ for $\epsilon \leq 2\log(1/\delta)$,

$$Pr[\Delta\text{conf} > \epsilon + T/2R] \quad \leq \quad \delta$$

For $\epsilon \leq 1$ we have that $T/2R = \epsilon^2/4\log(1/\delta)$ is way less than ϵ but not equal to zero, and so not what we claimed. However, the slightest tweak to the constant in front of R corrects this, without changing the spirit of the bound.

5.4 Deterministic Bounds on Confidence

The normal distribution leads to privacy violations in the rare, but possible case that $\log(p(a_j \pm 1)/p(a_j))$ is very large. For example, when a_j is large, we have that this ratio is

$$\log\left(\frac{\exp(-(a_j - 1)^2/2R)}{\exp(-a_j^2/2R)}\right) \; = (2a_j - 1)/2R \tag{34}$$

It seems reasonable to imagine distributions for which the ratio of adjacent probabilities is bounded, and *never* results in substantial privacy loss.

The density function $q(x) \propto \exp(-|x-\mu|/R)$ is one such distribution, as for all a_j, the ratio $q(a_j)/q(a_j \pm 1)$ is bounded by $\exp(1/R)$. Using a fairly crude bound, after T steps the additive increase can be at most T/R, independent of any randomization.

THEOREM 7. *The SuLQ primitive using the density function $q(x) \propto \exp(-|x|/R)$ is $(\epsilon, 0, T)$-private for $R \geq T/\epsilon$.*

Remark: While the requirement above that $R \geq T/\epsilon$ looks better than the result proved for normally distributed noise, this is misleading. The variance of the density function $q(x) \propto \exp(-|x|/R)$ is not R, but rather $2R^2$, and so the error we must incorporate to achieve the deterministic bounds is of the order of R rather than $R^{1/2}$.

One could imagine a scheme that stitches together the two approaches, using a density function like $\exp(-x^2/R)$ near the origin, and $\exp(-|x|/R)$ as we get farther out. This would have the deterministic guarantees of the latter distribution but with less variance. Understanding the tradeoffs and appropriate combination is interesting future research.

6. ACKNOWLEDGEMENTS

Several people have helped greatly to improve the exposition of this work and enhance our understanding of the related mathematics. We would particularly like to thank John MacCormick for his helpful discussions regarding the nature and validity of Bayesian reasoning.

7. REFERENCES

[1] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2001.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, 2000.

[3] N. R. Adam and J. C. Wortmann, Security-Control Methods for Statistical Databases: A Comparative Study, *ACM Computing Surveys 21*(4), pp. 515–556, 1989.

[4] S. Chawla, C. Dwork, F. McSherry, A. Smith and H. Wee, Toward Privacy in Public Databases, TCC 2005.

[5] D.E. Denning. Secure statistical database with random sample queries. ACM Trans. Database Syst. 5, 3(Sept.), 291-315, 1980.

[6] I. Dinur and K. Nissim, Revealing information while preserving privacy, *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 202-210, 2003.

[7] C. Dwork and N. Nissim, Privacy-Preserving Datamining on Vertically Partitioned Databases, *Proceedings of CRYPTO 2004*, pp. 528–544, 2004.

[8] A. Evfimievsky, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 211–222, 2003.

[9] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In *Proc. 28th International Colloquium on Automata, Languages and Programming*, pages 927–938. Springer-Verlag, 2001.

[10] S. Halevi, E. Kushilevitz, R. Krauthgamer, and K. Nissim. Private approximations of np-hard functions. In *Proc. 33th Annual ACM Symposium on the Theory of Computing*, pages 550–559, 2001.

[11] M. Kearns, Efficient Noise-Tolerant Learning from Statistical Queries, *JACM 45*(6), pp. 983 – 1006, 1998. See also *Proc. 25th ACM STOC*, pp. 392–401, 1993

[12] P. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems, CRYPTO 1996, Lecture Notes in Computer Science, Volume 1109, Jan 1996, Page 104.

[13] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002. An earlier version appeared in *Proc. Crypto 2000*.

[14] E. Leiss. Randomizing a practical method for protecting statistical databases against compromise. In Proceedings of the 8th Conference on Very Large Databases, pp. 189–196.

[15] M. J. O'Connel, Search Program for Significant Variables, *Comp. Phys. Comm. 8*, 1974.

[16] D. MacKay Information Theory, Inference, and Learning Algorithms *Cambridge University Press*, 2003.

[17] J. R. Quinlan, Induction of Decision Trees, Machine learning 1(1), 1986, pp. 81-106.