Practical Privacy: The SuLQ Framework

Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim

[1] Carnegie Mellon University
[2] Microsoft Research
[3] Microsoft Research
[4] Ben Gurion University

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 response is $\sum_{r \in S} f(DB_r)$, a noisy version of which is released. 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. Second, we examine the computational power of the SuLQ primitive. We show that it is very powerful indeed, in that slightly noisy versons of the following computations can be carried out with very few invocations of the primitive: principal component analysis, k means clustering, the ID3 algorithm, the perceptron algorithm, and (apparently!) all algorithms in the statistical queries learning model.

1 Introduction

We consider the problem of applying algorithms to a collection of private individual data, with the goal of producing utility while preserving privacy. We work in the Sub-Linear Queries output-perturbation framework, introduced by Dwork and Nissim [6]. In this framework, 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 response is $\sum_{r \in S} f(DB_r)$, a noisy version of which is released. It was shown in [6] that a strong form of privacy can be maintained using a surprisingly small amount of noise, provided the total number of queries is sublinear in the number database rows; hence the term Sub-Lineare Queries, or SuLQ, framework. The sub-linearity assumption becomes reasonable as databases grow increasingly larger.

It was already demonstrated in [6] that SuLQ databases provide utility as they allow for learning statistics of the input, sometimes even in the case of independently-operated vertically-partitioned databases. In this paper we greatly extend these results, slightly modifying the noise distribution and strengthening the analysis in [6] to support real-valued functions f, and then obtaining algorithms to compute a rich collection of supervised and unsupervised functionalities, including approximations to k-means clustering, principal component analysis, and the ID3 classification algorithm, while preserving privacy in a very precise and rigorous sense.

Our approach is quite straight forward – we run the SuLQ primitive with the individual private information playing the role of database entries. This gives rise to a calculus of noisy computation, in which we formulate our algorithms. We note that whereas in prior work privacy had to be proved 'from scratch', in the SuLQ framework privacy follows by virtue of working within the framework.

1.1 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 [11] 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 individual data, in the presence of perturbation noise (introduced for maintaining privacy).

Interestingly, the research of secure function evaluation protocols almost completely ignored the question of which functionalities preserve privacy. E.g. 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 case 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 et al. [8] in defining private approximations, and similarly Halevi et al. [9] 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; it does not say when is the information revealed by g 'private'.

The work of Agrawal and Srikant [2] rekindled the interest in perturbation techniques. Subsequent work demonstrated that the privacy definition introduced in [2] is far too weak [1] and suggested alternative privacy definitions [4–7] and perturbation techniques (see [6] for a short survey of the evolution of privacy definitions since [2]). Dinur and Nissim [5] researched the limitation of perturbation techniques in one-dimensional statistical databases, and showed that a fairly large perturbation noise is needed for preserving privacy with respect to polynomial time adversaries. Jointly with Dwork they showed, however, that if the total number of queries to a database is sublinear in the number of its entries n, then a very strong notion of privacy may be maintained with a surprisingly small amount of noise – a random quantity whose standard deviation is of order $o(\sqrt{n})$. This result was further extended to multi-attribute databases in [6].

This low amount of noise is significant for the following reason. If we think of each database entry a sample from some underlying probability distribution and we wish to gather statistics on properties P that occur with possibly small but still constant probability in the population, then the sampling error in our population of size n will be of order $\Omega(\sqrt{n})$. Thus, the noise that is added for the sake of protecting privacy is significantly smaller than the sampling error. In other words, providing privacy need not interfere with accuracy, so long as the number of statistical queries is not too large.

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.

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 $i' \neq i$. We define the monotonically-increasing 1-1 mapping $\mathrm{conf} : (0, 1) \to \mathbb{R}$ as follows: $\mathrm{conf}(p) = \log(p/(1-p))$. Note that while a small additive change in $\mathrm{conf}(p)$ implies a small additive change in p, the converse does not hold. Our definition of privacy is based on bounding the additive increase from $\mathrm{conf}(p_0^{i,f})$ to $\mathrm{conf}(p_T^{i,f})$.

Definition 1 ((ϵ, δ, T)-Privacy). *A database access mechanism is (ϵ, δ, T)-private if for every set of independent a priori beliefs $b : D \times [n] \to \mathbb{R}$, for every data element index i, for every predicate $f : D \to \{0, 1\}$, and for every adversary $\mathcal{A}$ making at most T queries,*

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

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

3 General SuLQ Databases

The database access mechanism we consider is an extension of the SuLQ DB of [6] 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 $(g : D \to [0,1])$.

1. Return $\sum_i g(d_i) + N(0,R)$.

The main technical theorem of this paper is an extension of the previous privacy result of [6] 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 all ϵ, δ, T, the SuLQ algorithm $\mathcal{A}(R)$ is (ϵ, δ, T)-private for $R > 8T \log^2(T/\delta)/\epsilon^2$.*

The proof analyzes the *a posteriori* belief $p_\ell^{i,f}$ that $f(d_i) = 1$ given the answers to the first ℓ queries $(\tilde{a}_1, \ldots, \tilde{a}_\ell)$ *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. Following [5], we study the random walk on the real line defined by $\mathrm{conf}(p_\ell^{i,f})$, and argue that with high probability T steps of the random walk do not suffice to reach distance ϵ. The full proof is conducted in Section 5.

4 Computation with the SuLQ Primitive

The basic SuLQ operation – query and noisy reply – can be viewed as a noisy computational primitive which 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.

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 notational simplicity, we consider SuLQ queries 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 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 Optimal Projections

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 $A^T A$, the eigenvectors can be easily computed using standard algorithms from numerical analysis.

Notice that as d_i is simply the ith row of A, we may write the matrix $A^T A$ as

$$A^T A = \sum_i d_i^T d_i \ , \tag{1}$$

where $d_i^T d_i \in \mathbb{R}^{d \times d}$ is the *outer product* of d_i. This suggests the following rather simple SuLQ implementation:

1. (d^2 queries) Approximate $A^T A = \sum_i d_i^T d_i$ by 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. See [**?**] for concrete perturbation bounds.

We remark that, using the techniques of [6] 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.

Principle Component Analysis PCA [12] 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, or equivalently, of $\sum_i (d_i - \mu)^T (d_i - \mu))$. We can compute an accurate approximation $\overline{\mu}$ to μ with only d additional queries, and then apply the approach above using $C = SuLQ(f(d_i) := (d_i - \overline{\mu})^T (d_i - \overline{\mu}))$. The query complexity of this function is still $O(d^2)$.

4.2 *k* Means

Given a collection of points $\{d_i\} \subset \mathbb{R}^d$, 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 "centers", points in $\mathbb{R}^d$, and forms clusters by associating each sample with the closest cluster center. Given a clustering of the points, the cluster centers minimizing the radii are exactly the means of each cluster, suggesting the following iterative update rule:

Given cluster centers $\mu_1, \ldots \mu_k$:
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 $1 \leq j \leq k$, 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 $\mathbb{R}^d$ and receive k new points in $\mathbb{R}^d$. In fact, we are able to emulate iterative k-means in SuLQ, using some care in our definition of f:

Given cluster centers $\mu_i \ldots, \mu_k$:

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

$$\overline{s}_j \quad = \quad SuLQ(f(d_i) := 1 \text{ if } j = \arg\min_j \|\mu_j - d_i\| \text{ and 0 otherwise})$$

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

$$\overline{\mu}'_j \quad = \quad SuLQ(f(d_i) := d_i \text{ if } j = \arg\min_j \|\mu_j - d_i\| \text{ and 0 otherwise})/\overline{s}_j$$

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

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

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

$$\|(\sum_{i \in S_j} d_i + n_j)/\overline{s}_j - \sum_{i \in S_j} d_i/s_j\| \quad = \quad \|(1/\overline{s}_j - 1/s_j)\sum_{i \in S_j} d_i + n_j/\overline{s}_j\| \tag{2}$$

$$\leq \quad |(s_j - \overline{s}_j)/\overline{s}_j|\|\mu'_j\| + \|n_j/\overline{s}_j\| \tag{3}$$

From our assumption that $|S_j| \gg R^{1/2}$, 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 $O((dR)^{1/2}/|S_j|)$.

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, $\langle d_i, w\rangle \cdot l_i > 0$. That is, the sign of ℓ_i agrees with that of the inner product $\langle d_i, w\rangle$, and the hyperplane orthogonal to w therefore separates positively labeled instances from negatively labeled ones.

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:

1. Initialize w randomly.
2. As long as there exists a j such that $\langle d_j, w\rangle \cdot l_j < 0$,
   (a) Set $w = w + d_j$.
3. Return w.

We will be unable to 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, not their membership in the set. We will synthesize an aggregate misclassified point and incorporate it, as in the following algorithm:

1. Initialize $w_0 = 0^d$ and $\overline{s}_0 = n$.
2. For $j = 0, 1, 2, \ldots$, repeating so long as $\overline{s}_j \gg R^{1/2}$
   (a) (1 query) Count the misclassified rows, computing

$$\overline{s}_j \quad = \quad SuLQ(f(d_i) := 1 \text{ if } \langle d_i, w_j \rangle \cdot l_i \leq 0 \text{ and 0 otherwise.})$$

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

$$\overline{v}_j \quad = \quad SuLQ(f(d_i) := l_i d_i \text{ if } \langle d_i, w_j \rangle \cdot l_i \leq 0 \text{ and 0 otherwise})/\overline{s}_j.$$

   (c) Set $w_{j+1} = w_j + \overline{v}_j$.
3. Return the final value of w.

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.

We now sketch the modified proof of convergence of the perceptron algorithm using the SuLQ primitive. We let S_j be the set of vectors misclassified by w_j in the jth round, and let $v_j = \sum_{i \in S_j} l_i d_i / \overline{s}_j$ 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 $\overline{s}_j$

Theorem 2. *If there exists a unit vector w' and scalar δ such that for all i, $l_i \langle w', d_i \rangle \geq \delta$ and for all j, $\delta \gg (dR)^{1/2}/s_j$ 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. Details appear in the Appendix (Section A).

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 $l_i \in L$ that we wish to predict given only the transaction attributes. The ID3 algorithm, introduced by Quinaln [13], 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 A_1 which is assigned to the corresponding leaf), or (ii) when the attribute set becomes empty (and 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 T, the subset of $\mathcal{T}$ for which A takes the value j is denoted $\mathcal{T}[A=j]$.

Given attributes $\mathcal{A} = A_1, \ldots, A_d$ and a labeled transaction set $\mathcal{T}$:

1. If $\mathcal{A} = \emptyset$: Return a leaf whose value is the majority vote on the labels of transactions in $\mathcal{T}$.
2. If $T[L] = l$ 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. Recursively apply the ID3 algorithm on inputs $((\mathcal{A} \setminus A), \mathcal{T}[A=j])$ for $j \in [t]$.
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 $H_L(\mathcal{T}) = -\sum_{k=1}^{t}(|\mathcal{T}[L=k]|/|\mathcal{T}|)\log(|\mathcal{T}[L=k]|/|\mathcal{T}|)$. 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 as $H_L(\mathcal{T}) - H_{L|A}(\mathcal{T})$. The attribute A that exhibits the highest gain is chosen in step 3 of the ID3 algorithm. Observe that one need only maximize $V_A = |\mathcal{T}| \cdot H_{L|A}(\mathcal{T}) = \sum_{j=1}^{t}\sum_{k=1}^{t}|\mathcal{T}[A=j \wedge L=k]| \cdot \log \frac{|\mathcal{T}[A=j \wedge L=k]|}{|\mathcal{T}[A=j]|}$ to figure out the "best attribute".

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 followed also in [11] for other reasons). Another limitation is that we will not be able to meaningfully recurse with the ID3 algorithm with small transaction sets.

Given attributes $\mathcal{A} = A_1, \ldots, A_d$ and a transaction set $\mathcal{T}$ expressed as a conjunction of terms $(A=j)$:

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. recursively apply the ID3 algorithm on inputs $(\mathcal{A} \setminus \bar{A}), \mathcal{T}_i \wedge (\bar{A}=i)$ for $i \in [t]$.
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 2. *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 $N^A_{j,k} + \epsilon = O(|\mathcal{T}|\Delta/t^2)$, 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^A_j and $N^A_{j,k}$ (we use the notation τ_j for $|\mathcal{T}[A = j]|$ and $\tau_{j,k}$ for $|\mathcal{T}[A = j \wedge L = k]|$):

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

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

$$N^A_{j,k} \log \frac{N^A_{j,k}}{N^A_j} = \tau_{j,k} \log \frac{\tau_{j,k}}{\tau_j} \pm O(\epsilon) \ .$$

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

4.5 Capturing the Statistical Queries Learning Model

The Statistical Query model, proposed by Kearns in [10], 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 f and an accuracy τ, and is returned the expected fraction of samples satisfying f to within additive error τ. Conceptually, the framework models drawing a sufficient number of samples so that the observed count of samples satisfying f 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 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.

Given as input a predicate p and accuracy τ:
1. Initialize $tally = 0$.
2. Repeat $t \geq R/\tau n^2$ times
   (a) Set $tally = tally + SuLQ(f(d_i) := p(d_i))$
3. Return $tally/tn$.

Theorem 3. *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*

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

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 properly 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.

$$T \;\geq\; \sum_{j \leq q} \lceil R\,\log(q/\delta)/\tau_j n^2 \rceil \;\geq\; q + R\,\log(q/\delta)/n^2 \times \sum_{j \leq q} 1/\tau_j \tag{4}$$

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

$$n^2 \;\geq\; \frac{R\,\log(q/\delta)}{T-q} \times \sum_{j \leq q} 1/\tau_j. \tag{5}$$

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. To bound the change that occurs, we will view the confidence as a submartingale, and use Azuma's inequality to bound its magnitude.

5.1 The Evolution of Confidence as a Sum

We start by working to convert the confidence at any point in time into a sum, with terms contributed at each step. We first convert the *a posteriori* beliefs into joint beliefs, relying on the fact that the appropriate scaling is equivalent for the numerator and denominator.

$$\frac{b(f(d_i)=1|a_1,a_2,\ldots,a_j)}{b(f(d_i)=0|a_1,a_2,\ldots,a_j)} \;=\; \frac{b(f(d_i)=1 \wedge a_1 \wedge a_2 \ldots \wedge a_j)}{b(f(d_i)=0 \wedge a_1 \wedge a_2 \ldots \wedge a_j)} \;=\; \frac{\mathbf{Numer}_j}{\mathbf{Denom}_j}$$

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

$$\mathbf{Numer}_j \;=\; \int_{D_1} b(a_1 \wedge a_2 \ldots \wedge a_j \wedge d_i = x)dx \;=\; \int_{D_1} b(a_1 \wedge a_2 \ldots \wedge a_j | d_i = x)\; b(d_i = x)dx \tag{6}$$

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.

$$\mathbf{Numer}_j = \int_{D_1} b(a_j|d_i=x)\; b(a_1 \wedge a_2 \ldots \wedge a_{j-1}|d_i=x)\; b(d_i=x)dx \tag{7}$$

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

$$\mathbf{Numer}_j = \int_{D_1} p(a_j|d_i = x)\ b(a_1 \wedge a_2 \ldots \wedge a_{j-1}|d_i = x)\ b(d_i = x)dx \tag{8}$$

At this point observe that letting $w_x = b_x / \int_x b_x dx$, one can write $\int_x a_x b_x dx = \int_x w_x a_x dx \int_x b_x dx$. Moreover, the w_x integrate to one and are independent of the a_x terms. Applying this observation to the above equation, choosing $a_x = p(a_j|d_i = x)$, we get

$$\begin{aligned}\mathbf{Numer}_j &= \int_{D_1} w_x p(a_j|d_i = x)dx \int_{D_1} b(a_1 \wedge a_2 \ldots \wedge a_{j-1}|d_i = x)b(d_i = x)dx &(9)\\ &= \left(\int_{D_1} w_x p(a_j|d_i = x)dx\right) \times \mathbf{Numer}_{j-1} &(10)\end{aligned}$$

Applying the same arguments to $\mathbf{Denom}_j$, we arrive at the equation describing the evolution of confidence from one step to the next.

$$\log\left(\frac{\mathbf{Numer}_j}{\mathbf{Denom}_j}\right) = \log\left(\frac{\int_{D_1} w_x p(a_j|d_i = x)dx}{\int_{D_0} w_x p(a_j|d_i = x)dx}\right) + \log\left(\frac{\mathbf{Numer}_{j-1}}{\mathbf{Denom}_{j-1}}\right) \tag{11}$$

The prior beliefs are encoded in the w_x, whose rich structure we will ignore completely.

5.2 Azuma's Inequality

The evolution of confidence takes the form of a sum whose terms are random quantities. This is analogous to a martingale, and we use Azuma's inequality to bound the variation of the sum.

Lemma 3 (Azuma). *Let $s_1, \ldots s_T$ be i.i.d. random variables such that $E[s_i] \leq \alpha$ and $|s_i| \leq \beta$.*

$$Pr[|\sum_i s_i| > \lambda(\alpha + \beta)T^{1/2} + T\alpha] \leq 2e^{-\lambda^2/2}$$

We apply Azuma's Inequality to the sum suggested in Eq. (11), with each s_j term associated with the confidence increase from seeing a_j. Some work will be conducted in Lemmas 4 and 6 in Appendix B to determine the relevant values of α and β, but once determined we can prove

Theorem 4. *$\forall\, \delta$, with probability at least $1 - \delta$ choosing R, the variance on the noise, to satisfy*

$$R > 8\log(2/\gamma)\log(T/\delta)T/\epsilon^2 + (2\lambda T^{1/2} + T)/\epsilon$$

ensures that for each $(target, predicate)$ pair, after T queries the probability that the confidence has increased by more than ϵ is at most γ.

Proof. We apply Azuma's inequality to the change in confidence, which by Lemmas 4 and 6 has bounds on expected and absolute increases, respectively,

$$\alpha \ = \ 1/2R \ \text{ and } \ \beta \ = \ (2\log(T/\delta)/R)^{1/2} + 1/2R$$

yielding

$$Pr[|\Delta(conf)| > \lambda(1/2R + (2\log(T/\delta)/R)^{1/2} + 1/2R)T^{1/2} + T/2R] \leq 2e^{-\lambda^2/2}\ .$$

Rearranging and collecting terms gives

$$Pr[|\Delta(conf)| > \lambda(2\log(T/\delta)T/R)^{1/2} + \lambda T^{1/2}/R + T/2R] \leq 2e^{-\lambda^2/2} .$$

The second and third terms are smaller than the first, and will generally be of little consequence as R grows large. When $R \geq 4$,

$$Pr[|\Delta(conf)| > \lambda(4\log(T/\delta)T/R)^{1/2}] \leq 2e^{-\lambda^2/2}$$

Finally, choosing $\lambda = (2\log(2/\gamma))^{1/2}$ gives the bound we desire.

5.3 Deterministic Bounds on Confidence

The bounds on the growth in confidence presented thusfar rely on two types of events that happen with high probability. First, the absolute increase in a single step should be small, which occurs whenever $|a_j|$ is not overly large. Second, the martingale should not deviate wildly from its expectation, which also happens with a significant probability.

We can remove these two sources of randomness, and the concerns about improbable but possible privacy breaches, through a slightly altered sanitization and analysis. First, we can discard the martingale argument and simply use the bound of Lemma 6 on the absolute increase in confidence. This will weaken our result, but leaves no doubt as to the aggregate change in confidence. Second, we can remove our use of the normal distribution, which can leak privacy if large values of a_j emerge. Instead, the density function $q(x) \propto e^{-|x-\mu|/R}$ serves as an excellent distribution, as no matter the size of the sample a_j, the ratio $q(a_j)/q(a_j \pm 1)$ is bounded by $e^{-1/R}$. Combining these two techniques, we get the following result:

Theorem 5. *The modified SuLQ primitive that incorporates noise drawn from the density function $q(x) = e^{-|x|/R}/R$ is $(\epsilon, 0, T)$-private for $R > T/\epsilon$.*

Proof. It is not hard to see that for two densities $p(x)$ and $q(x)$ with means μ_p and μ_q, we can bound the absolute increase in confidence by

$$\log(p(x)/q(x)) \;\leq\; -|x-\mu_p|/R + |x-\mu_q|/R \;\leq\; |\mu_p - \mu_q|/R \,. \tag{12}$$

As this bound is independent of x, we have a deterministic guarantee. We multiply this by T to yield an deterministic upper bound on the aggregate increase in confidence of $T/R = \epsilon$, for $\|\mu_p - \mu_q\| \leq 1$.

Remark: While the requirement above that $R > T/\epsilon$ looks better than the result proved for normally distributed noise, this is misleading. The variance of the density function $q(x) = e^{-|x|/R}/R$ 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 $R^{1/2}$.

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, to appear, TCC 2005.
5. 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.
6. C. Dwork and N. Nissim, Privacy-Preserving Datamining on Vertically Partitioned Databases, *Proceedings of CRYPTO 2004*
7. 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.
8. 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.
9. 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.
10. 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
11. Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002. An earlier version appeared in *Proc. Crypto 2000*.
12. M. J. O'Connel, Search Program for Significant Variables, *Comp. Phys. Comm. 8*, 1974.
13. J. R. Quinlan, Induction of Decision Trees, Machine learning 1(1), 1986, pp. 81-106.

A Proof of Perceptron Implementation

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

$$\langle w', w_{j+1} \rangle \;=\; \langle w', w_j + \sum_{i \in S_j} (l_i d_i)/\overline{s}_j + n_j \rangle \;=\; \langle w', w_j \rangle + \langle w', \sum_{i \in S_j} (l_i d_i)/\overline{s}_j \rangle + \langle w', n_j \rangle$$

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

$$\begin{aligned} \langle w', w_{j+1} \rangle \;&\geq\; \langle w', w_j \rangle + \langle w', \sum_{i \in S_j} l_i d_i/\overline{s}_j \rangle - \delta/4 && (13) \\ &\geq\; \langle w', w_j \rangle + \delta |S_j|/\overline{s}_j - \delta/4 && (14) \\ &\geq\; \langle w', w_j \rangle + \delta/4. && (15) \end{aligned}$$

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

$$\langle w', w_j \rangle \;\geq\; j\delta/4 \qquad (16)$$

On the other hand, the length of w is evolving slowly. As each vector v_j that we incorporate has negative projection onto it, we have that $\langle w_j, v_j \rangle \leq 0$, and therefore

$$\begin{aligned} \|w_{j+1}\|^2 = \|w_j + v_j + n_j\|^2 \;&=\; \langle (w_j + v_j + n_j), (w_j + v_j + n_j) \rangle \\ &=\; \langle w_j, w_j \rangle + \langle v_j, v_j \rangle + \langle n_j, n_j \rangle + 2\langle (w_j + v_j), n_j \rangle + 2\langle w_j, v_j \rangle \\ &\leq\; \|w_j\|^2 + \|v_j\|^2 + \|n_j\|^2 + 2\langle (w_j + v_j), n_j \rangle \end{aligned}$$

With the exception of the $2\langle (w_j + v_j), n_j \rangle$ term, it is clear that $\|w_j\|^2$ is evolving linearly with j. The $2\langle (w_j + v_j), n_j \rangle$ terms are each independent, zero mean normals. We expect their sum to be negative at many points in the future. In a round r when their sum is non-positive, we have that

$$\|w_r\|^2 \;\leq\; \sum_{j \leq r} (\max_i \|d_i\|^2 + \|n_j\|^2) \qquad (17)$$

We have assumed that for all i, $\langle w', l_i d_i \rangle \geq \delta$, 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. (16) and Eq. (17) we see that

$$\delta r/4 \;\leq\; \langle w', w \rangle \;\leq\; \|w\| \;\leq\; (2r \max_i \|d_i\|)^{1/2} \tag{18}$$

constraining $r \leq 32 \max_i \|d_i\|/\delta^2$, bounding the number of iterations. Assuming that $\|w_j\|$ does not shrink, we expect $\sum_j \langle (w_j + v_j), n_j \rangle$ to be negative again not much after r steps. If $\|w_j\|$ does shrink, excellent.

B Supporting Lemmas: Expected Increase and Absolute Range

Before proceeding to bound the expected increase, we will need to observe a particular instance of Jensen's inequality, specifically applied to the log function.

Theorem 6 (Jensen). *For an arbitrary function f, for a probability measure $p(x)$ (ie: $\int_x p(x)dx = 1$)*

$$\int_x p(x) \log(f(x)) dx \;\leq\; \log\left(\int_x p(x) f(x) dx\right)$$

We will also use the above inequality with each side negated and the inequality reversed.

We now bound the expected amount of increase in the confidence value from one additional observed a_j. Here we bound the expected increase by a convex combination of relative entropies, which we then evaluate in Lemma 5.

While we state the following lemma generally, it may help to think of the density function $p(y)$ as a normal distribution centered around the "true" value of $g(d_i)$ and each of the $p_x(y)$ density function as corresponding to normal distributions centered around $g(x)$.

Lemma 4 (Expected Increase). *For any density function $p(y)$ of finite entropy, set of density functions $\{p_x(y)\}$, and w_x with $\int_x w_x dx = 1$, we bound the expected increase in confidence in a single step as*

$$\int_y p(y) \left[\log\left(\frac{\int_{D_1} w_x p_x(y) dx}{\int_{D_0} w_x p_x(y) dx}\right)\right] dy \;\leq\; \int_{D_0} w_x \int_y p(y) \log\left(p(y)/p_x(y) dy\right) dx \,.$$

Proof. We start by dividing the numerator and denominator by $p(y)$. While $p(y)$ can be quite small, these divisions amount to adding and subtracting $\int_y p(y) \log(p(y))$, the entropy of the distribution, which is finite.

$$\int_y p(y) \left[\log\left(\frac{\int_{D_1} w_x p_x(y) dx}{\int_{D_0} w_x p_x(y) dx}\right)\right] dy = \int_y p(y) \left[\log\left(\frac{\int_{D_1} w_x p_x(y)/p(y) dx}{\int_{D_0} w_x p_x(y)/p(y) dx}\right)\right] dy \tag{19}$$

Separating the numerator and denominator, and then the integrands

$$= \int_y p(y) \log\left(\int_{D_1} w_x p_x(y)/p(y) dx\right) dy - \int_y p(y) \log\left(\int_{D_0} w_x p_x(y)/p(y) dx\right) dx \tag{20}$$

Applying Jensen's Inequality to log both ways, then switching the order of integration,

$$\leq \log\left(\int_y p(y)\int_{D_1} w_x p_x(y)/p(y)dxdy\right) - \int_y p(y)\int_{D_0} w_x \log\left(p_x(y)/p(y)\right)dxdy \tag{21}$$

$$= \log\left(\int_{D_1} w_x \int_y p(y)p_x(y)/p(y)dydx\right) - \int_{D_0} w_x \int_y p(y)\log\left(p_x(y)/p(y)\right)dydx \tag{22}$$

The $p(y)$ terms cancel in the first term, leaving $\int_y p_x(y)dy = 1$. The log of the first term is thus zero, leaving us with only the second term. Passing the negation through the log, we get

$$\int_y p(y)\left[\log\left(\frac{\int_{D_1} w_x p_x(y)dx}{\int_{D_0} w_x p_x(y)dx}\right)\right]dy \leq \int_{D_0} w_x \int_y p(y)\log\left(p(y)/p_x(y)\right)dydx \tag{23}$$

We now prove that the right hand quantity in the above lemma is small when the distributions are Gaussians with close means and high common variance.

Lemma 5. *Let $p(x)$ and $q(x)$ be normal densities with means μ_p and μ_q and common variance R.*

$$\int_x p(x)\log\left(p(x)/q(x)\right)dx = (\mu_p - \mu_q)^2/2R$$

Proof. Expanding the LHS using the definition of the Gaussian density function,

$$\int_x p(x)\log\left(p(x)/q(x)\right)dx = \int_x p(x)\log\left(\frac{(2\pi R)^{-1/2}e^{-(x-\mu_p)^2/2R}}{(2\pi R)^{-1/2}e^{-(x-\mu_q)^2/2R}}\right)dx \tag{24}$$

$$= \int_x p(x)\log\left(\frac{e^{(2x\mu_p-\mu_p^2)/2R}}{e^{(2x\mu_q-\mu_q^2)/2R}}\right)dx \tag{25}$$

Taking the log and rearranging the resulting algebra, we get

$$\int_x p(x)\log\left(p(x)/q(x)\right)dx = \int_x p(x)(2x(\mu_p-\mu_q) - (\mu_p^2-\mu_q^2))dx/2R \tag{26}$$

$$= \left(2(\mu_p-\mu_q)\int_x xp(x)dx - (\mu_p^2-\mu_q^2)\int_x p(x)dx\right)/2R \tag{27}$$

Note that p is just a distribution, with $\int_x p(x)dx = 1$ and $\int_x xp(x)dx = \mu_p$. Substituting accordingly yields

$$\int_x p(x)\log\left(p(x)/q(x)\right)dx \;=\; \left(2\mu_p^2 - 2\mu_q\mu_p - \mu_p^2 + \mu_q^2\right)/2R \;=\; (\mu_p-\mu_q)^2/2R \tag{28}$$

Lemma 6 (Absolute Increase). *For any two gaussians $p(x), q(x)$ with variance R satisfying $|\mu_p - \mu_q| \leq 1$,*

$$Pr[\max_j\left(\log\left(p(x)/q(x)\right)\right) \;>\; (2\log(T/\delta)/R)^{1/2} + 1/2R] < \delta\;. \tag{29}$$

Proof. Following calculations from the previous lemma,

$$\log\left(p(x)/q(x)\right) \;=\; 2x(\mu_p-\mu_q)/2R - (\mu_p^2-\mu_q^2)/2R \;\leq\; x/R + 1/2R \tag{30}$$

Using the definition of the normal density function we can see that the probability that x is large is quite small.

$$Pr[\log\left(p(x)/q(x)\right) \;>\; (2\log(1/\delta)/R)^{1/2} + 1/2R] < \delta \tag{31}$$

Generalizing to T trials using a union bound,

$$Pr[\max_j\left(\log\left(p(x)/q(x)\right)\right) \;>\; (2\log(T/\delta)/R)^{1/2} + 1/2R] < \delta \tag{32}$$