K-Randomization

Maxim Zhilyaev David Zeber

January 7, 2016

1 Differential Privacy

The typical setting for differential privacy is as follows. We consider a **database** as a collection of records. Each record is an element of some space $\mathcal{D}$, and a database $\mathbf{x}$ is a vector of n records: $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{D}^n$.

We wish to release information retrieved from the database by means of a **query**, a function A mapping the database into another space: $A : \mathcal{D}^n \to \mathcal{S}$. The result of applying a query to a database is termed a **transcript**. The query usually applies some aggregation to the database records, and so the output space $\mathcal{S}$ is generally of lower dimensionality than the original database. If the query is randomized, i.e., $A(\mathbf{x}) = A(\mathbf{x}; \xi)$ for a random element ξ, then the transcript will be a random element of $\mathcal{S}$.

The notion of differential privacy for a database query is that the resulting transcript does not change substantially when a record in the database is modified, i.e., transcripts are not sensitive to particular individual records in the database. Hence, releasing query transcripts publicly will not jeopardize privacy, since information regarding individual records cannot be gained by analyzing query transcripts.

Differential privacy for a randomized query A is formulated by comparing the transcripts generated by applying A to two very similar databases $\mathbf{x}, \mathbf{x}' \in \mathcal{D}^n$. We say the databases **differ in one row** if $\sum_{i=1}^{n} I(x_i \neq x_i') = 1$.

Definition. A randomized query A is ϵ-**differentially private** if, for any two databases $\mathbf{x}, \mathbf{x}' \in \mathcal{D}^n$ differing in one row,

$$\mathsf{P}[A(\mathbf{x}) \in S] \leq \exp(\epsilon) \cdot \mathsf{P}[A(\mathbf{x}') \in S] \tag{1.1}$$

for all $S \subset \mathcal{S}$ (measurable).

In other words, the transcripts from the two databases databases differing in one row are close in distribution. An alternative notion of differing in one row that is sometimes used is that $\mathbf{x}'$ includes an additional record that is not in $\mathbf{x}$: $\mathbf{x} \in \mathcal{D}^n$, $\mathbf{x}' \in \mathcal{D}^{n+1}$, and $x_i = x_i'$ for $i = 1, \ldots, n$.

If $\mathcal{S}$ is finite, which is common in cases where the transcript involves integer counts, then the

distribution of the transcript $A(\mathbf{x})$ can be represented using its pmf $\mathsf{P}[A(\mathbf{x}) = s]$ for $s \in \mathcal{S}$. In this case, the differential privacy condition can also be expressed in terms of the pmf.

Proposition 1.1. *If $\mathcal{S}$ is finite, then A is ϵ-**differentially private** if and only if*

$$\mathsf{P}[A(\mathbf{x}) = s] \leq \exp(\epsilon) \cdot \mathsf{P}[A(\mathbf{x}') = s] \tag{1.2}$$

for all $s \in \mathcal{S}$, where $\mathbf{x}, \mathbf{x}'$ differ in one row.

Proof. ($\Leftarrow$) Given $S \subset \mathcal{S}$, we can write $\mathsf{P}[A(\mathbf{x}) \in S] = \sum_{s \in S} \mathsf{P}[A(\mathbf{x}) = s]$. If $\mathsf{P}[A(\mathbf{x}') \in S] = 0$, then $\mathsf{P}[A(\mathbf{x}') = s] = 0$ for each $s \in S$. From (1.2) we have that $\mathsf{P}[A(\mathbf{x}) = s] = 0$ as well, and so $P[A(\mathbf{x}) \in S] = 0$, verifying (1.1). Otherwise, if $\mathsf{P}[A(\mathbf{x}') \in S] > 0$,

$$\frac{\mathsf{P}[A(\mathbf{x}) \in S]}{\mathsf{P}[A(\mathbf{x}') \in S]} = \frac{\sum_{s \in S} \mathsf{P}[A(\mathbf{x}) = s]}{\sum_{s \in S} \mathsf{P}[A(\mathbf{x}') = s]} \leq \max_{s \in S} \frac{\mathsf{P}[A(\mathbf{x}) = s]}{\mathsf{P}[A(\mathbf{x}') = s]} \leq \exp(\epsilon),$$

using Lemma (need ref).
($\Rightarrow$) Take $S = \{s\}$ in (1.1). □

2 Bit vector reporting

Our goal is to establish differential privacy properties for user data reported in the form of vectors of bits. To protect user privacy, each user record is randomized prior to leaving the client and anonymized on reaching the server. We now describe the randomization procedure, and place ourselves in the setting of Section 1 by representing it as a query applied to a database.

2.1 Bit randomization

For our purposes, a **bit** is an integer $b \in \{0, 1\}$, and a **bit vector** is a vector $x \in \{0, 1\}^L$. Bits and bit vectors are randomized in the following way.

Definition. The **bit randomization** procedure R with **lie probability** $0 < q < 1/2$ flips a bit b with probability q, and leaves it as-is with probability $p := 1 - q$:

$$R(b) = \begin{cases} b & \text{with prob } p \\ 1 - b & \text{with prob } q \end{cases}.$$

This can be expressed concisely as

$$R(b) = R(b; \xi) = b \cdot \xi + (1 - b) \cdot (1 - \xi) \quad \text{where } \xi \sim Ber(p).$$

We extend the procedure to **bit vector randomization** by applying the randomization independently to each bit in the vector. Given a bit vector $x = (b_1, \ldots, b_L)$, define

$$R(x) = R(x; \xi) = \big(R(b_1; \xi_1), \ldots, R(b_L; \xi_L)\big) \quad \text{where } \xi = (\xi_1, \ldots, \xi_L) \overset{\text{iid}}{\sim} Ber(p).$$

Remark. Note that R reports the original bit value with probability $p = 1 - q > q$, and lies with probability q. This is equivalent to the randomized response procedure where the value is reported as-is with probability $1 - f$, and with probability f the reported value is the outcome of the toss of a fair coin. In our case, $q = f/2$.

Remark. If $q = 1/2$, then $R(0) \stackrel{d}{=} R(1)$, and the reported value is "completely" randomly generated, i.e., independently of the original value.

We now consider the distribution of the randomized bit vectors. It can be expressed in terms of the Hamming distance between the original and randomized vectors:

$$\delta(x, x') = \sum_{j=1}^{L} I(x_j \neq x'_j) = \sum_{j=1}^{L} |x_j - x'_j|.$$

For a single bit, the randomization has lied when the outcome is different from the original value:

$$\mathsf{P}[R(b) = s] = p^{I(b=s)} \cdot q^{I(b\neq s)} = (1-q)^{1-\delta(b,s)} \cdot q^{\delta(b,s)}.$$

For a bit vector x, this becomes

$$\mathsf{P}[R(x) = s] = p^{\sum I(x_j=s_j)} \cdot q^{\sum I(x_j\neq s_j)} = (1-q)^{L-\delta(x,s)} \cdot q^{\delta(x,s)}.$$

Note that this probability is maximized when $\delta(x, s) = 0$ (the randomized vector s is identical to the original vector x), and minimized when $\delta(x, s) = L$. In the latter case, we say that s is the **opposite** of x. In other words, the most likely outcome of randomizing a bit vector is obtaining an identical vector.

2.2 Reporting for bit records

We now place ourselves in the setting of Section 1 for bit-vector user records, as required in the sequel.

Set $\mathcal{D} = \{0, 1\}^L$. We use the term **collection** (of records) interchangeably with "database". We consider a randomized query A that randomizes each record in the collection independently, and aggregates the results by reporting occurrence counts for every possible randomization outcome. We adopt this aggregation step as a model for anonymization. After anonymization, any link to the original collection or ordering is lost, and the information contained in the results is embodied solely by the reported values.

In the following, we rely on the fact that $\mathcal{D}$ is finite, and we assume a specific enumeration $\mathcal{D} = (d_1, \ldots, d_{2^L})$. The ordering is unimportant at this point, although it will be convenient to assume that $d_1 = (1, \ldots, 1)$ and $d_{2^L} = (0, \ldots, 0)$.

Definition. The randomized query $A : \mathcal{D}^n \to \mathcal{S} = \{0, \ldots, n\}^{2^L}$ maps collections of bit vectors to occurrence counts as follows.

(a) Extend the bit randomization R to collections $\mathbf{x}$ by applying it independently to each vector:

$$R(\mathbf{x}) = R(\mathbf{x}; \boldsymbol{\xi}) = \big(R(x_1; \xi_1), \ldots, R(x_n; \xi_n)\big) \quad \text{where } \xi_i = (\xi_{i1}, \ldots, \xi_{iL}) \text{ and } \xi_{ij} \overset{\text{iid}}{\sim} Ber(p).$$

We call $R(\mathbf{x})$ the **synthetic** collection obtained from the **original** collection $\mathbf{x}$.

(b) Define the function Φ that counts occurrences of the elements of $\mathcal{D} = (d_1, \ldots, d_{2^L})$ in a collection $\mathbf{y}$:

$$\Phi(\mathbf{y}) := \left(\sum_{i=1}^{n} I(y_i = d_1), \ldots, \sum_{i=1}^{n} I(y_i = d_{2^L}). \right)$$

Note that, if $\Phi(\mathbf{y}) = (s_1, \ldots, s_{2^L})$, then $s_1 + \cdots + s_{2^L} = n$.

Finally,

$$A = \Phi \circ R.$$

3 Old stuff

Furthermore, if A randomizes each record in the database independently, i.e., $A(\boldsymbol{x}) = A(\boldsymbol{x}, \boldsymbol{X}) := \big(A_0(x_1, X_1), \ldots, A_0(x_n, X_n)\big)$ where X_i are independent, then $\boldsymbol{S} = \boldsymbol{S}_0^n$ and $s = (s_1, \ldots, s_n)$ with $s_i \in \boldsymbol{S}_0$. In this case $P[A(\boldsymbol{x}) = s] = P[A_0(x_1) = s_1, \ldots, A_0(x_n) = s_n] = \prod P[A_0(x_i) = s_i]$. If $\boldsymbol{x}$ and $\boldsymbol{x'}$ differ in one row (wlog $x_1 \neq x_1'$ and $x_i = x_i'$ for $i = 2, \ldots, n$), then

$$\frac{P[A(\boldsymbol{x}) = s]}{P[A(\boldsymbol{x'}) = s]} = \frac{P[A_0(x_1) = s_1]}{P[A_0(x_1') = s_1]}.$$

Therefore, in this case, the query A will satisfy differential privacy if

$$P[A_0(x) = s] \leq \epsilon \cdot P[A_0(x') = s]$$

for all $x, x' \in D$ and $s \in \boldsymbol{S}_0$. This is the formulation used in the RAPPOR paper that applies to differences between individual records rather than collections differing on a single element.