A Framework of Metrics for Differential Privacy from Local Sensitivity

Abstract: The meaning of differential privacy (DP) is tightly bound with the notion of distance on databases, typically defined as the number of changed rows. Considering the semantics of data, this metric may be not the most suitable one, particularly when a distance comes out as larger than the data owner desired (which would undermine privacy). In this paper, we give a mechanism to specify continuous metrics that depend on the locations and amounts of changes in a much more nuanced manner. Our metrics turn the set of databases into a Banach space. In order to construct DP information release mechanisms based on our metrics, we introduce derivative sensitivity, an analogue to local sensitivity for continuous functions. We use this notion in an analysis that determines the amount of noise to be added to the result of a database query in order to obtain a certain level of differential privacy, and demonstrate that derivative sensitivity allows us to employ powerful mechanisms from calculus to perform the analysis for a variety of queries. We have implemented the analyzer and evaluated its efficiency and precision.

Keywords: differential privacy, sensitivity

DOI 10.2478/popets-2020-0023
Received 2019-08-31; revised 2019-12-15; accepted 2019-12-16.

1 Introduction

Differential privacy [8] (DP) is one of the most prominent ways to quantitatively define privacy losses from releasing derived information about data collections, and to rigorously argue about the accumulation of these losses in information processing systems. Differentially private information release mechanisms invariably employ the addition of noise somewhere during the processing, hence reducing the utility of the result. For specific information processing tasks, or families of tasks, there exist carefully designed methods to achieve DP with only a little loss in utility [16]; for some methods, this loss asymptotically approaches zero as the size of the processed dataset grows [24]. But a general method for making an information release mechanism differentially private is to add noise of appropriate magnitude to the output of this mechanism. This magnitude depends on the sensitivity of the mechanism — the amount of change of its output when its input is changed by a unit amount. This paper is devoted to the study of computing (or safely approximating) the sensitivity of information release mechanisms given by their source code.

The definition of the ratio of changes of the outputs and inputs of the information release mechanism requires metrics on both of them. A common approach is requiring the outputs to be numeric. In fact, the output of the mechanism is a single real number, and the distance between outputs is their difference. This approach can be generalized to mechanisms that output histograms, i.e. tuples of numbers.

The definition of the metric on inputs is a richer question, and really reflects how the data owner wants to quantify its privacy. DP can be defined with respect to any metric on the set of possible inputs [5]. For database tables and databases, a common metric has been the number of different rows [23]. But the data owner may consider different changes in one row of some database table as different, particularly if it involves some numeric attributes. A small change in such attribute in a single row may be considered as “small”, and the DP mechanism should strive to hide such change. A large change in the same attribute in a single row may be allowed to be more visible in the output. A change in geographic coordinates may be defined differently from a change in the length or the quantity of something. Finally, when several attributes change in a single row, then there are many reasonable ways of combining their changes (including summing them up, or taking their maximum) in defining how much the row has changed. A variety of definitions for the combination of changes is also possible if several rows change in a table, or several tables change in a database. We note that for inputs to the information release mechanism, the overestimation of their distance leads to the underestimation of the
amount of noise added to achieve a certain level of DP. Hence it is crucial to handle a wide range of metrics, from which the data owner may select the one that he considers to best reflect his privacy expectations.

With new definitions of DP come the questions of achieving it. DP w.r.t. different metrics has been considered before [5], but only with global sensitivity of the query being used to scale the noise necessary for achieving it. In this paper, we generalize the notion of local sensitivity [24] to real-valued metrics and at the same time simplify it, tying it with the fundamental notions of functional analysis and expanding the kinds of input data to which it may be applied. We give a general framework for defining different metrics on databases, and computing the sensitivities of queries with respect to these input metrics. It has the following scope:

- Being based on local sensitivity, we require the actual content of the database in order to compute the distribution of added noise for achieving DP.
- Our metrics for rows, tables, and databases are combinations and compositions of $\ell_p$-metrics with $1 \leq p \leq \infty$. Certain rows or columns may be left out from the computation of the metric, effectively declaring them as public. The columns that remain should be numeric (or encoded as numeric). The number of table rows defines dimensionality of the metric space, and is considered public.
- Computation of sensitivity of a function is based on its derivative, so the framework primarily covers continuous functions. Non-continuous constructions (like filtering) are approximated by continuous functions, which itself introduces some noise.
- As queries we support a significant subset of SQL with projections, filters, joins and also certain types of subqueries. The final output is numeric, i.e. an aggregation or a couple of aggregations. We do not support DISTINCT queries, and GROUP BY is limited to public and discrete attributes.

We start from presenting the metric-defining framework (Sec. 4.1). We follow it up with the definition of sensitivity that makes use of the continuity of the metrics; this sensitivity is applicable for mappings from the defined metric spaces to real numbers (Sec. 4.2). We show that this sensitivity can be used to measure the amount of noise to be added to the outputs of these maps in order to make them differentially private. We describe how to compute this sensitivity and its smooth upper bounds from the description of mappings. We show how these results can be applied to SQL queries (Sec. 5).

The multitude of different metrics presents a novel problem. The computations of derivative sensitivity (our name for the analogue of local sensitivity, defined in this paper) for various functions out of metric spaces can be done only for certain metrics, depending on the particular function. At the same time, the data owner has defined a particular metric for the databases, and wants the queries against his database to be differentially private with respect to this metric. In Sec. 5.3 we show how to approximate and upper-bound sensitivities of the same function with respect to different metrics, and thereby provide a solution to this problem.

We evaluate a concrete implementation of our theory in Sec. 7, demonstrating its benefits on particular examples. Our analyzer takes as inputs the description of the database and the metric on it, the query to be performed, as well as the actual contents of the database (as required by local sensitivity based approaches), and returns the value of smooth derivative sensitivity of this query at this database. The returned value can be used to scale the added noise in order to obtain a certain level of DP. The analyzer is integrated with the database management system in order to perform the computations specific to the contents of the database.

2 Related Work

Differential privacy was introduced by Dwork [8]. PINQ [23] is a worked-out example of using it for providing privacy-preserving replies to database queries. In an implementation, our analyzer of SQL queries would occupy the same place as the PINQ wrapper which analyses LINQ queries and maintains the privacy budget.

It has been recognized that any metric on the set of possible inputs gives us a definition of DP with respect to this metric [5, 13], but there has been no investigation of a systematic construction of such metrics that are also usable in constructing DP mechanisms. A particular application of DP w.r.t. a particular metric has been the privacy-preserving processing of location data [6]. The personalized differential privacy [12] can also be seen as an instance of using an arbitrary metric, albeit with a more complex set of distances. In Blowfish [17, 20], the metric rises from the distance on a graph where vertices are the possible database instances.

We use norms to state the privacy requirements on input data. Normed vector spaces have appeared in the DP literature in the context of $K$-normed mechanisms [3, 15], which extend the one-dimensional and
generalize the many-dimensional Laplace mechanism. These mechanisms are rather different from our techniques and they do not explore the use of completeness of norms and differentiability of information release mechanism to find their sensitivity.

Nissim et al. [24] introduce local sensitivity and its smooth upper bounds, and use them to give differentially private approximations for certain statistical functions. The local sensitivity of a function is similar to its derivative. This has been noticed [19], but we are not aware of this similarity being thoroughly exploited, except perhaps for devising DP machine learning methods [28]. In this paper, this similarity will play a central role.

A couple of different static approaches for determining sensitivities of SQL queries or their upper bounds have been proposed. Palamidessi and Stronati [25] apply abstract interpretation to an SQL query, following assumption extending sensitivities of SQL queries or their upper bounds may be computed in practice. The local sensitivity of a function is similar to its smooth upper bounds, and use them to give differentially private approximations for certain statistical functions. The local sensitivity of a function is similar to its derivative. This has been noticed [19], but we are not aware of this similarity being thoroughly exploited, except perhaps for devising DP machine learning methods [28]. In this paper, this similarity will play a central role.

A couple of different static approaches for determining sensitivities of SQL queries or their upper bounds have been proposed. Palamidessi and Stronati [25] apply abstract interpretation to an SQL query, following assumption extending sensitivities of SQL queries or their upper bounds may be computed in practice. The local sensitivity of a function is similar to its smooth upper bounds, and use them to give differentially private approximations for certain statistical functions. The local sensitivity of a function is similar to its derivative. This has been noticed [19], but we are not aware of this similarity being thoroughly exploited, except perhaps for devising DP machine learning methods [28]. In this paper, this similarity will play a central role.

The computability of the precise sensitivity of the queries is studied by Arapinis et al. [2]. They identify a subclass of queries (Conjunctive queries with restricted WHERE-clauses) for which the sensitivity can be precisely determined. However, they also show that the problem is uncomputable in general. In addition, they show how functional dependencies and cardinality constraints may be used to upper-bound sensitivities of join-queries.

Cardinality constraints are also used by Johnson et al. [18] in their abstract interpretation based approach. For a database, they consider the maximum frequency of a value of an attribute in one of the tables, maximized over all databases at most at some distance to the original database, thereby building on the notion of (smooth) local sensitivity.

3 Preliminaries

3.1 Sensitivity and Differential Privacy

Let $X$ be the set of possible databases. We assume that there is a metric $d_X(x, x')$ for $x, x' \in X$, quantifying the difference between two databases. For example, we could define $d_X(x, x') = n$ for two datatables whose tables differ in exactly $n$ rows in total.

For a set $Y$, let $D(Y)$ denote the set of all probability distributions over $Y$ (seen as mappings from $Y$ to $[0, 1]$). For $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$. Let $[1, \infty]$ denote the set $\{x \in \mathbb{R} | x \geq 1\} \cup \{\infty\}.$

Suppose that someone wants to make a query to the database. If the data is (partially) private, the query output may leak some sensitive information about the data. Noise can be added to the output to reduce privacy leakage. One possible definition of privacy is that, from the output one should not be able to learn whether an individual row is present in the table or not. More generally, we may consider two neighboring databases $x, x' \in X$, i.e. such that $d_X(x, x') = 1$.

Definition 1 (differential privacy, [9]). Let $X$ be a metric space and $f : X \to D(Y)$. The mapping $f$ is $(\epsilon, \delta)$-differentially private if for all (measurable) $Y' \subseteq Y$, and for all $x, x'$, where $d_X(x, x') = 1$, the following inequality holds:

$$Pr[f(x) \in Y'] \leq e^\epsilon Pr[f(x') \in Y'] + \delta.$$  

The mapping $f$ is $\epsilon$-differentially private if it is $(\epsilon, 0)$-differentially private.

The noise magnitude depends on the difference between the outputs of $f$. The more different the outputs are, the more noise we need to add to make them indistinguishable from each other. This is quantified by the global sensitivity of $f$.

Definition 2 (global sensitivity). For $f : X \to Y$, the global sensitivity of $f$ is $GS_f = \max_{x, x' \in X} \frac{d_X(f(x), f(x'))}{d_X(x, x')}$. Sensitivity is the main tool in arguing the differential privacy of various information release mechanisms. For mechanisms that add noise to the query output, this value serves as a parameter for the noise distribution. The noise is proportional to $GS_f$. One suitable noise distribution is $\text{Lap}(\frac{GS_f}{\epsilon})$, where $\text{Lap}(\lambda)(z) \propto e^{-|z|/\lambda}$. The sampled noise is sufficient to make the output of $f$ differentially private, regardless of the input of $f$.

Differential privacy itself can also be seen as an instance of sensitivity. Define the following distance $d_{\text{dp}}$ over $D(Y)$:

$$d_{\text{dp}}(x; x') = \inf\{\epsilon \in \mathbb{R}^+ | \forall y \in Y : |\ln(f(y)/f'(y))| \leq \epsilon\}.$$  

Then a mechanism $M$ from $X$ to $Y$ is $\epsilon$-DP iff it is $\epsilon$-sensitive with respect to the distances $d_X$ on $X$ and $d_{\text{dp}}$ on $D(Y)$. 
3.2 Local and Smooth Sensitivity

Our work extends the results of [24], which makes use of instance-based additive noise. Since noise is always added to the output of a query that is applied to a particular state of the database, and some state may require less noise than the other, the noise magnitude may depend on the data to which the function is applied.

Definition 3 (local sensitivity). For \( f : X \rightarrow Y \), an integer-valued metric \( d_X : X \times X \rightarrow \mathbb{N} \), and \( x \in X \), the local sensitivity of \( f \) at \( x \) is \( LS_f(x) = \max_{x' \in X : d_X(x,x') = 1} d_Y(f(x), f(x')) \).

The use of local sensitivity may allow the use of less noise, particularly when the global sensitivity of \( f \) is unbounded. However, \( LS_f(x) \) may not be directly used to determine the magnitude of the noise, because this magnitude may itself leak something about \( x \). To solve this problem, Nissim et al. [24] use a smooth upper bound on \( LS_f(x) \). It turns out that such an upper bound is sufficient to achieve differential privacy for \( f \); potentially with less noise than determined by \( GS_f \).

Definition 4 (smooth bound). For \( \beta > 0 \), a function \( S : X \rightarrow \mathbb{R}_+ \) is a \( \beta \)-smooth upper bound on \( f \) if it satisfies the following requirements:
- \( \forall x \in X : S(x) \geq f(x) \);
- \( \forall x, x' \in X : S(x) \leq e^{\beta d_X(x,x')} S(x') \).

Nissim et al. [24] showed how to add noise based on the smooth bound on \( LS_f \). The statement that we present in Theorem 1 is based on combination of Lemma 2.5 and Example 2 of [24].

Definition 5 (generalized Cauchy distribution). For a parameter \( \gamma \in \mathbb{R}_+, \gamma > 1 \), the generalized Cauchy distribution \( \text{GenCauchy}(\gamma) \in D(\mathbb{R}) \) is given by the proportionality

\[ \text{GenCauchy}(\gamma)(x) \propto 1/(1 + |x|^\gamma) \, . \]

Theorem 1 (local sensitivity noise [24]). Let \( \eta \) be a fresh random variable sampled according to \( \text{GenCauchy}(\gamma) \). Let \( \alpha = \frac{1}{\gamma} \) and \( \beta = \frac{\alpha}{\gamma} \). For a function \( f : X \rightarrow \mathbb{R} \), let \( S : X \rightarrow \mathbb{R} \) be a \( \beta \)-smooth upper bound on the local sensitivity of \( f \). Then the information release mechanism \( f(x) + \frac{S(x)}{\alpha} \cdot \eta \) is \( \epsilon \)-differentially private.

3.3 Norms and Banach Spaces

The local sensitivity in Def. 3 is defined for integer-valued metrics on inputs. Let us try to generalize it to real-valued metrics. We may choose a small value \( \delta > 0 \) and round \( d_X \) up to the nearest higher multiple of \( \delta \):

\[ d_X^\delta(x, x') := \delta \left\lceil \frac{d_X(x, x')}{\delta} \right\rceil \]

It is easy to see that \( d_X^\delta \) is still a metric. Then we can define local sensitivity as

\[ LS_f^\delta(x) = \frac{1}{\delta} \max_{x' \in X : d_X^\delta(x, x') = \delta} d_Y(f(x), f(x')) \, . \]

This is very similar to rescaled Def. 3. Finally, we define

\[ LS_f(x) = \lim_{\delta \rightarrow 0} LS_f^\delta(x) \]

\[ = \lim_{\delta \rightarrow 0} \frac{1}{\delta} \max_{x' \in X : 0 < d_X(x, x') \leq \delta} d_Y(f(x), f(x')) \]

This looks quite similar to the definition of derivative (except for the use of max). If \( X = Y = \mathbb{R} \) and \( d_X \) and \( d_Y \) are the absolute-value metrics then this will be equal to the absolute value of the derivative \( f' \) at \( x \) if \( f'(x) \) exists. In this case we get \( LS_f(x) = |f'(x)| \).

Because it is based on derivative, we call it derivative sensitivity and write \( DS[f](x) = |f'(x)| \).

We would like to extend derivative sensitivity to metrics other than absolute value and to functions with more than one variable. One such extension of derivative is the Fréchet derivative in Banach spaces.

First, we recall some basics of Banach space theory. Throughout this paper, we denote vectors by \( \mathbf{x} \), and norms by \( \|\cdot\|_X \), where \( N \) specifies the particular norm. Banach spaces do not allow arbitrary metrics and instead require norms, but many useful metrics can also be viewed as norms.

Definition 6 (norm and seminorm). A seminorm is a function \( \|\cdot\| : V \rightarrow \mathbb{R} \) from a vector space \( V \), satisfying the following axioms for all \( \mathbf{x}, \mathbf{y} \in V \):
- \( \|\mathbf{x}\| \geq 0 \);
- \( \|\alpha \mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\| \) (implying that \( \|0\| = 0 \));
- \( \|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \) (triangle inequality).

Additionally, if \( \|\mathbf{x}\| = 0 \) holds only if \( \mathbf{x} = 0 \), then \( \|\cdot\| \) is a norm.

Some of the most useful norms in practice are \( \ell_1 \) (i.e., sum), \( \ell_2 \) (geographical distance), and \( \ell_\infty \) (maximum). These are instances of \( \ell_p \)-norms.

Definition 7 (\( \ell_p \)-norm). Let \( X_i \subseteq \mathbb{R}, p \in [1, \infty] \). The \( \ell_p \)-norm of \( \mathbf{x} \in X_1 \times \cdots \times X_n \), denoted \( \|\mathbf{x}\|_p \), is defined
The notion of the derivative of a function can be also defined as follows:

\[ \|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}. \]

For \( p = \infty \), \( \ell_\infty \) is defined as

\[ \|x\|_\infty = \lim_{p \to \infty} \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \max_{i=1}^{n} |x_i|. \]

**Definition 8** (Banach space). A Banach space is a vector space with a norm that is complete (i.e., each converging sequence has a limit).

Banach spaces combine vector spaces with distances, which are necessary for defining differential privacy. The completeness property allows us to define derivatives. Using the norm of a Banach space, we may generalize the notion of continuous function from real numbers to Banach spaces.

**Definition 9** (Continuous function in Banach space). Let \( V \) and \( W \) be Banach spaces, and \( U \subset V \) an open subset of \( V \). A function \( f : U \to W \) is called continuous if

\[ \forall \epsilon > 0 : \exists \delta > 0 : \|x - x'\|_V \leq \delta \Rightarrow \|f(x) - f(x')\|_W \leq \epsilon. \]

The notion of the derivative of a function can be also extended to Banach spaces.

**Definition 10** (Fréchet derivative). Let \( V \) and \( W \) be Banach spaces, and \( U \subset V \) an open subset of \( V \). A function \( f : U \to W \) is called Fréchet differentiable at \( x \in U \) if there exists a bounded linear operator \( df_x : V \to W \) such that

\[ \lim_{h \to 0} \left( \frac{\|f(x+h) - f(x) - df_x(h)\|_W}{\|h\|_V} \right) = 0. \]

Such operator \( df_x \) is called Fréchet derivative of \( f \) at the point \( x \).

The mean value theorem can be generalized to Banach spaces (to a certain extent).

**Theorem 2** (Mean value theorem ([4], Chapter XII)). Let \( V \) and \( W \) be Banach spaces, and \( U \subset V \) an open subset of \( V \). Let \( f : U \to W \), and let \( x, x' \in U \). Assume that \( f \) is defined and is continuous at each point \( (1-t)x + tx' \) for \( 0 \leq t \leq 1 \), and differentiable for \( 0 < t < 1 \). Then there exists \( t^* \in (0,1) \) such that

\[ \|f(x) - f(x')\|_W \leq \|df_x\|_{V \to W} \|x - x'\|_V \]

for \( z = (1-t^*)x + t^*x' \), where \( \|\|_{V \to W} \) denotes the norm of operator that maps from \( V \) to \( W \).

### 4 Metrics and Derivative Sensitivity

#### 4.1 Banach Spaces of Databases

Let us have a database with a number of tables. The schema for each table is fixed. We see that database as a point in some Banach space (thus the distance between databases is the norm of their difference), where each cell in each table corresponds to a dimension of that space. As dimensions of Banach spaces are fixed (indeed, they are vector spaces \( \mathbb{R}^n \) with some extra structure), the number of rows in each table is also fixed, and each row can be seen to have a public identity (similarly to [17]). In this paper, we consider the following composite norms and seminorms as possible norms for databases:

**Definition 11** (composite seminorm). Let \( \|\|_N \) be a seminorm of the vector space \( \mathbb{R}^n \). It is a composite seminorm if one of the following holds for all \( x = (x_1, \ldots, x_n) \in \mathbb{R}^n \):

- There exists \( i \in [n] \), such that \( \|x\|_N = |x_i| \). Such seminorm uses the variable \( x_i \).
- There exists a composite seminorm \( \|\|_M \) and \( a \in \mathbb{R}^+ \), such that \( \|x\|_N = a \cdot \|x\|_M \). The seminorm \( \|\|_N \) uses the same variables as \( \|\|_M \).
- There exist composite seminorms \( \|\|_{M_1}, \ldots, \|\|_{M_k} \) and \( p \in [1, \infty] \), s.t. \( \|x\|_N = \left( \sum_{k=1}^{k} \|x\|_{M_k} \right)^p \). The seminorm \( \|\|_N \) uses the union of the variables used by all \( \|\|_{M_k} \).

Let \( \text{vars}(N) \) be the set of variables used by \( \|\|_N \).

We normally define the norms for rows of each table using the constructions allowed in Def. 11. We then state that the norm of the table is some \( \ell_p \)-norm of the vector of the norms of its rows (last item of Def. 11) and the norm of the database is some \( \ell_p \)-norm of the vector of the norms of its tables. Note that the notion of seminorm is used only for building blocks of composite \( \ell_p \)-norms, and a seminorm constructed as in Def. 11 becomes a norm if \( \text{vars}(N) = \{x_1, \ldots, x_n\} \).

#### 4.2 From Derivative Sensitivity to DP

Having defined a database as a point in some Banach space, we propose constructions that allow us to achieve differential privacy for functions defined over Banach spaces.
Definition 12. Let \( X \) be (an open convex subset of) a Banach space. Let \( f : X \to \mathbb{R} \). Let \( f \) be Fréchet differentiable at each point of \( X \). The derivative sensitivity of \( f \) is the following mapping from \( X \) to \( \mathbb{R}_+ \):

\[
\text{DS}[f](x) = \|df_x\|
\]

where \( df_x \) is the Fréchet derivative of \( f \) at \( x \) and \( \|df_x\| \) is the operator norm of \( df_x \).

Similarly to the local sensitivity of [24], we will need to find smooth upper bounds on derivative sensitivity to compute the noise. We extend the definition of smoothness (Def. 4) to the case where \( X \) is any Banach space:

Definition 13. Let \( p : X \to \mathbb{R} \) and \( \beta \in \mathbb{R} \). The mapping \( p \) is \( \beta \)-smooth, if \( p(x) \leq e^{b} \|x' - x\| \cdot p(x') \) for all \( x, x' \in X \).

The next theorem shows how to compute the magnitude of noise to be added to \( f(x) \) in order to obtain a differentially private information release mechanism. A smooth upper bound on \( \text{DS}[f] \) plays the central role here. We consider the same noise distributions as in [24].

Theorem 3. Let \( \gamma, b, \beta \in \mathbb{R}_+ \), \( \gamma > 1 \). Let \( \epsilon = (\gamma + 1)(b + \beta) \). Let \( \eta \) be a random variable distributed according to GenCauchy(\( \gamma \)). Let \( c \) be a \( \beta \)-smooth upper bound on \( \text{DS}[f] \) for a function \( f : X \to \mathbb{R} \). Then \( g(x) = f(x) + \frac{c(x)}{\beta} \cdot \eta \) is \( \epsilon \)-differentially private.

Theorem 3 can be proven similarly to Theorem 1, which has been done in [24]. The main difference is that, since we are using derivative sensitivity instead of local sensitivity, we apply the mean value theorem to find an upper bound on \( \|f(x) - f(y)\| \) for two neighbouring databases \( x \) and \( y \). The detailed proof can be found in App. C.2.

Cauchy noise has heavy tails. In its stead, we may use Laplace noise to achieve DP, but this requires a more complex proof and only gives us \((\epsilon, \delta)\)-DP. The strength of our result depends on a technical condition on the \( \beta \)-smooth upper bound on \( \text{DS}[f] \), for which the key notion is the following:

Definition 14. A path in a Banach space \( X \) is a continuous function \( h : [0, 1] \to X \). The path \( h \) is shortest, if for all \( x_1, x_2, x_3 \in [0, 1] \), \( x_1 \leq x_2 \leq x_3 \), the equality \( d_X(h(x_1), h(x_3)) = d_X(h(x_1), h(x_2)) + d_X(h(x_2), h(x_3)) \) holds.

Theorem 4. Let \( b, \beta, \epsilon \in \mathbb{R}_+ \), \( b > 0 \), \( b + \beta \leq \epsilon \). Define \( k = 1 + (\epsilon - b)/\beta \). Let \( \delta = e^{-k} \). Let \( \eta \) be a random variable distributed according to Lap(1). Let \( c \) be a \( \beta \)-smooth upper bound on \( \text{DS}[f] \) for a function \( f : X \to \mathbb{R} \), where \( X \) is Banach space and \( d_X \) is the distance corresponding to the norm of \( X \). Define \( g(x) := f(x) + \frac{c(x)}{\beta} \cdot \eta \). Then \( g \) is \((\epsilon, 2\epsilon\delta)\)-differentially private. If, additionally for any two points \( x_1, x_2 \in X \) there exists a shortest path between them, such that \( c \) is monotonic along that path, then the factor \( \sim 2 \) may be removed.

This theorem is proved in App. C.3.

4.3 Computing Derivative Sensitivity and Smooth Upper Bounds

Given a description of a function \( f : \mathbb{R}^n \to \mathbb{R} \), how do we determine (a \( \beta \)-smooth upper bound) on its derivative sensitivity? This question is ill-posed, because we have not stated the norm on \( \mathbb{R}^n \); Def. 11 gives us many possibilities to define it. If two norms are related, then the derivative sensitivity with respect to one of them should tell us something about the other, too.

A composite seminorm in \( \mathbb{R}^n \) can be seen as the semantics of a formal expression over the variables \( x_1, \ldots, x_n \), where the term constructors are \(|\ldots|_p\) of any arity. We write \( N \leq M \) for two seminorms \( N \) and \( M \), if \( ||x||_N \leq ||x||_M \) for all \( x \in \mathbb{R}^n \). The following results about \( \leq \) are proved in App. C.4.

Lemma 1. Let \( N \) be a composite norm over \( X = (x_1, \ldots, x_n) \). Let composite seminorms \( N' \) and \( V_1, \ldots, V_m \) be such, that \( N = N'(V_1, \ldots, V_m) \), and for all \( i \in [n] \) let \( W_i \) be a seminorm such that \( V_i \leq W_i \). Then, \( N'(V_1, \ldots, V_m) \leq N'(W_1, \ldots, W_m) \).

Lemma 2. Let \( x = (x_1, \ldots, x_n) \in \mathbb{R}^n \). Let \( N \) be a composite norm, defined over variables \( x \). There exist \( 0 \leq \alpha_i, \beta_i \in \mathbb{R} \) for \( i \in [n] \), such that \( ||\alpha_1 x_1, \ldots, \alpha_n x_n||_p \leq ||x_1, \ldots, x_n||_N \leq ||\beta_1 x_1, \ldots, \beta_n x_n||_q \), where:
- \( p \) is the largest \( \ell_p \)-norm constructor in \( N \);
- \( q \) is the smallest \( \ell_p \)-norm constructor in \( N \).

The following three lemmas give us the basic combiners for statements about derivative sensitivity, reducing the task of finding the derivative sensitivity of a particular function with respect to a particular norm on its domain, to a series of tasks from basic calculus. We prove these lemmas in App. C.5.

Lemma 3. Let \( f : \mathbb{R}^n \to \mathbb{R} \), and let \( \mathbb{R}^n \) be equipped with the norm \( \ell_p \). Then \( ||df_x|| \) is the \( \ell_q \)-norm of the gradient.
Lemma 5. (a) Let \( \nabla f(x) \), where \( q = \frac{p}{p-1} \) (if \( p = 1 \) then \( q = \infty \) and vice versa).

The \( \ell_p \)-norm is the dual norm of the \( \ell_q \)-norm; we denote \( q \) by dual(\( p \)).

Lemma 4. Let \( f : \mathbb{R}^n \to \mathbb{R} \) have the derivative sensitivity \( g \) with respect to the norm \( N \). Let \( a \cdot N \) denote the scaling of the output of the norm \( N \) by \( a \in \mathbb{R}_+ \). Then \( f \) has the derivative sensitivity \( g/a \) with respect to \( a \cdot N \).

Lemma 5. (a) Let \( (V_1, \| \cdot \|_{V_1}) \) and \( (V_2, \| \cdot \|_{V_2}) \) be Banach spaces. Let \( V = V_1 \times V_2 \). Let for all \( (v_1, v_2) \in V \),

\[
\|(v_1, v_2)\|_V = \|(v_1, v_2)\|_{V_1} = \|(v_2)\|_{V_2}
\]

Then \( (V, \| \cdot \|_V) \) is a Banach space.

(b) Suppose furthermore that a function \( f : V \to \mathbb{R} \) is differentiable at each point of \( V \). Fix a point \( v = (v_1, v_2) \in V \). Let \( g : V_1 \to \mathbb{R} \) be such that \( g(x_1) = f(x_1, v_2) \) and \( h : V_2 \to \mathbb{R} \) be such that \( h(x_2) = f(v_1, x_2) \).

Let \( c_1 = DS[y](v_1) \) and \( c_2 = DS[b](v_2) \). Then \( DS[f](v) = \|(c_1, c_2)\|_{\text{dual}(p)} \).

These lemmas reduce the computation of the derivative sensitivity of a function to the computation of certain partial derivatives. We note that while the computation of the local sensitivity of a mapping may have a high computational complexity [24], the continuity of our functions ensures that the computation of the derivatives falls into the polynomial complexity class under some mild conditions [21].

Example. Consider computing differentially privately the time that a ship takes to reach the port. This time can be expressed as

\[
f(x, y, v) = \frac{\|x, y\|_2}{v}, \quad f : \mathbb{R}^3 \to \mathbb{R}
\]

where \( (x, y) \) are the coordinates of the ship (with the port at \((0, 0)\)) and \( v \) is the speed of the ship. When thinking about sensitivity of the information, and expressing it in terms of the change to the variables \( x, y, v \), we consider the distance between geographic locations to be the Euclidean distance, and the distance between speeds to be their arithmetic difference. We are interested in hiding both the location and the speed, and would like to combine the distances by adding them up, i.e., a change in both the location and speed is considered a greater change than only one of these two changes. As the change in location and change in speed are measured in different units, and may have different significance, we add scalings to these two norms. Hence we want to compute the sensitivity of \( f \) with respect to the norm \( \|a \cdot x, y\|_2, b \cdot v\|_1 \). Here \( a, b \in \mathbb{R}_+ \) are scalings.

Let \( f^a(x, y) = f^{ab}(y) = f^{bv}(v) = f(x, y, v) \), where the \( f \)-s with superscripts are considered functions with less arguments; the superscripts are interpreted as parameters instead.

The partial derivatives of \( f \) (as well as \( f \)-s with superscripts) are

\[
\frac{\partial f}{\partial x} = \frac{\partial f^v}{\partial x} = \frac{df^{y,v}}{dx} = \frac{x}{v \sqrt{x^2 + y^2}}
\]

\[
\frac{\partial f}{\partial y} = \frac{\partial f^v}{\partial y} = \frac{df^{x,v}}{dy} = \frac{y}{v \sqrt{x^2 + y^2}}
\]

\[
\frac{\partial f}{\partial v} = \frac{\partial f^{y,v}}{dv} = -\frac{\sqrt{x^2 + y^2}}{v^2}
\]

By Lemma 3, the absolute values of these derivatives are the derivative sensitivities of \( f^{y,v} \), \( f^{x,v} \) and \( f^{y,v} \), respectively. Also by Lemma 3, the derivative sensitivity of \( f^v \) with respect to the \( \ell_2 \)-norm \( \|x, y\|_2 \) is

\[
DS[f^{y,v}](x, y, v) = DS[f^{x,v}](x, y, v) = DS[f^{y,v}](x, y, v) = \frac{\sqrt{2}}{|v|}.
\]

By Lemma 4, we have to divide (3) and (2) by respectively \( a \) and \( b \), in order to obtain the derivative sensitivities of \( f^a \) and \( f^{bv} \) with respect to the scaled norms \( a \cdot \|x, y\|_2 \) and \( b \cdot \|v\|_1 \). Finally, we use Lemma 5 to obtain the derivative sensitivity of \( f \):

\[
DS[f](x, y, v) = \|(DS[f^a](x, y), DS[f^{bv}](y))\|_2 = \max\{\sqrt{2}/|a||v|, \sqrt{x^2 + y^2}/(bv^2)\}.
\]

In this example, we have used the \( \ell_1 \) norm to compare speeds. This is perhaps not the most useful measure, because it states that the difference between speeds “1” and “2” is the same as between the speeds “101” and “102”, although the first change would affect the arrival time greatly, and the second, not so much. A more reasonable approach is to apply the norm not to the speed \( v \) itself, but to something derived from it. One reasonable choice seems to be \( 1/v \). In this way, a multiplicative change in speed would correspond to a multiplicative change in the arrival time. An even better way to define the norm on speeds is to apply the \( \ell_1 \) norm to \( \ln v \). We explore this variant below.

To achieve differential privacy, we need to find a smooth upper bound on the derivative sensitivity. We have combining lemmas for deriving the smoothness properties of functions from Banach spaces, and their derivative sensitivities. The following lemma, proved in App. C.6, gives an alternative definition of \( \beta \)-smoothness. It is easier to use, and implies Def. 13.
Lemma 6. Let $X$ be a Banach space. If $DS[f]$ exists then $f : X \rightarrow \mathbb{R}$ is $\beta$-smooth if $\frac{DS[f](x)}{|f(x)|} \leq \beta$ for all $x \in X$.

As a particular instance of Lemma 6, a differentiable function $f : \mathbb{R} \rightarrow \mathbb{R}$ is $\beta$-smooth if $|f'(x)| \leq \beta$.

The following three lemmas describe the different possibilities for computing $\beta$-smooth upper bounds for composite functions. They are proved in App. C.7.

Lemma 7. Let $f(x) : \mathbb{R} \rightarrow \mathbb{R}$ be $\beta_f$-smooth, and let $g(x) : \mathbb{R} \rightarrow \mathbb{R}$ be $\beta_g$-smooth.
1. If $f(x), g(x) > 0$, then $f(x) + g(x)$ is $\max(\beta_f, \beta_g)$-smooth;
2. $f(x) \cdot g(x)$ is $\beta_f + \beta_g$-smooth;
3. $f(x) / g(x)$ is $\beta_f + \beta_g$-smooth.

Lemma 8. Let $X_i$ for $i \in \{1, \ldots, n\}$ be Banach spaces, and let $X = \prod_{i=1}^{n} X_i$. Let $f_i : X_i \rightarrow \mathbb{R}$ be $\beta_i$-smooth. Then, $f(x_1, \ldots, x_n) = \|f_1(x_1), \ldots, f_n(x_n)\|_p$ is $(\|\beta_i\|_1)_n$-smooth as well as max$_{1 \leq i \leq n}(\beta_i)$-smooth, where the norm of $X$ is the $\ell_{\text{dual}(p)}$-combination of the norms of all $X_i$.

In general, the smoothness is worse if the variables of different $f_i$ are not disjoint, as shown in the next lemma.

Lemma 9. Let $X_i$ for $i \in \{1, \ldots, n\}$ be Banach spaces, $X = \prod_{i=1}^{n} X_i$. Let $f_i : X \rightarrow \mathbb{R}$ be $\beta_i$-smooth for $X_j$. Let $x = (x_1, \ldots, x_n)$. Then, $f(x) = \|f_1(x), \ldots, f_n(x)\|_p$ is $(\text{max}_i \beta_i)^n$-smooth.

Example. Extending the previous example, let us find a smooth upper bound for the derivative sensitivity of the function expressing the time it takes for some ship (out of $n$) to reach the port at the coordinates $(0,0)$. This $3n$-variable function is

$$f(x_1, y_1, v_1, \ldots, x_n, y_n, v_n) = \frac{n}{\text{max}_i \|x_i, y_i\|_2 / v_i}.$$ 

Note that $v_i$ is in the power $-1$ and we do not know how to find the smooth derivative sensitivity of the function $v_i^{-1} (v_i)$ (we only know how to do it for power functions with exponent $\geq 1$). Let us define $w_i = \zeta \ln v_i$.

The coefficient $\zeta$ is used to control the distance by which the whole input vector changes if $\ln v_i$ is changed by 1. Similarly, we add a coefficient to the geographical coordinates: $s_i = \alpha x_i, t_i = \alpha y_i$. The ratio between $\zeta$ and $\alpha$ shows how much the privacy of the speed is valued compared to privacy of the geographical coordinates, relative to each other. The smaller $\zeta / \alpha$ is, the more we value the privacy of the speed.

We consider $(s_1, t_1, v_1, \ldots, s_n, t_n, w_n)$ as an element of the Banach space $(\mathbb{R}^{3n}, \|\cdot\|)$ where

$$\|(s_1, t_1, v_1, \ldots, s_n, t_n, w_n)\| = \|\left(\|\cdot\|_2, \|\cdot\|_2\right)\|_p$$

Then $v_i = e^{w_i / \zeta}, x_i = s_i / w_i, y_i = t_i / w_i$ and

$$g(s_1, t_1, v_1, \ldots, s_n, t_n, w_n) = \frac{1}{\zeta} \min_{\alpha=1}^n \|s_i, t_i\|^2_p e^{w_i / \zeta}$$

Now the derivative sensitivity of $g_{w,i}(w_i) = e^{w_i / \zeta}$ is

$$DS[g_{w,i}](w_i) = \frac{1}{\zeta} e^{w_i / \zeta}$$

which is $\frac{1}{\zeta}$-smooth. The function $g_{w,i}$ itself is also $\frac{1}{\zeta}$-smooth. The derivative sensitivity of $g_{s,t,i}(s_i, t_i) = \|s_i, t_i\|^2_p$ in $(\mathbb{R}^2, \ell_2)$ is

$$DS[g_{s,t,i}](s_i, t_i) = 1$$

which is $\beta$-smooth for all $\beta$. The function $g_{s,t,i}$ is $\frac{1}{\zeta}$-smooth if $\frac{1}{\|s_i, t_i\|^2_p} \leq \frac{1}{\zeta}$, i.e. if $\|s_i, t_i\|^2_p \geq \zeta$. A $\frac{1}{\zeta}$-smooth upper bound on $g_{s,t,i}$ is

$$\hat{g}_{s,t,i}(s_i, t_i) = \begin{cases} \|s_i, t_i\|^2_p & \text{if } \|s_i, t_i\|^2_p \geq \zeta \\ \zeta e^{s_i / \zeta, t_i / \zeta} & \text{otherwise} \end{cases}$$

An upper bound on the derivative sensitivity of $g(s_i, t_i, w_i) = \frac{\|s_i, t_i\|^2_p}{e^{w_i / \zeta}}$ is $c_{g_{s,t,i}}(s_i, t_i, w_i) = \frac{1}{\zeta} \max_{\alpha=1}^n \|s_i, t_i\|^2_p e^{w_i / \zeta}$, and it is $\frac{1}{\zeta}$-smooth because $DS[g_{w,i}](w_i), DS[g_{s,t,i}, g_{s,t,j}, g_{w,j}(w_j), \hat{g}_{s,t,i}(s_i, t_i)]$ are $\frac{1}{\zeta}$-smooth.

A $\frac{1}{\zeta}$-smooth upper bound on $DS[g]$ is

$$c(u) = \frac{\max_{\alpha=1}^n c_{g_{s,t,i}}(s_i, t_i, w_i)}{\max_{\alpha=1}^n \|s_i, t_i\|^2_p e^{w_i / \zeta}}$$

where $u = (s_1, t_1, v_1, \ldots, s_n, t_n, w_n)$.

Now we can use Theorem 3 to compute an $\epsilon$-differentially private version of $g$:

$$h(u) = g(u) + c(u) \cdot \eta$$

$$\epsilon = (\gamma + 1)(b + 1)$$

$$\gamma > 1, b > 0, \eta \sim \text{GenCauchy}(\gamma)$$

To compute an $\epsilon$-differentially private version of $f$, we first transform $(x_1, y_1, v_1, \ldots, x_n, y_n, v_n)$ into $u$ and then compute $h(u)$. \hfill \blacksquare$
5 Application to SQL Queries

In this section, we describe how the theory of Sec. 4 can be applied to SQL queries, computing the smooth upper bounds of their derivative sensitivities, such that an appropriate amount of noise may be added to turn them differentially private. Our theory deals with functions that return a numeric value, so the query should return a single output. We consider queries of the form

\[
\text{SELECT } \text{aggr expr FROM t1 AS } s_1,\ldots,t_n \text{ AS } s_n \text{ WHERE condition},
\]

where:
- \(\text{expr}\) is an expression over table columns, computed as a continuous function.
- \(\text{condition}\) is a boolean expression over predicates \(\{x < 0, x = 0\}\), where \(x\) is an expression of the same form as \(\text{expr}\). Since all functions have to be continuous, these predicates are computed using continuous approximations to the step functions, listed in Table. 1.
- \(\text{aggr}\) is one of the operations \(\text{SUM}, \text{COUNT}, \text{MIN}, \text{MAX}\).

GROUP BY queries can be simulated by generating for each group a separate query, with a filter selecting that particular group. Hence, we can group either by a public or a discrete attribute to get a finite number of groups. We do not support \text{DISTINCT} queries, as we do not know how to efficiently continuously approximate a function that removes repeating elements from a list of arguments.

Let our database have \(n\) tables. Let \(R_i\) be the Banach space of the potential values of rows for the \(i\)-th table, and \(n_i\) the number of rows in this table. Let \(T_i = R_i^{n_i}\) and \(D = T_1 \times \cdots \times T_n\). Starting from the norms on \(R_i\), and applying \(\ell_p\)-norms, define a norm for \(D\).

### 5.1 Derivative Sensitivity of Queries with Respect to a Component

Our ultimate goal is to enforce differential privacy w.r.t. a certain component of a database. A component is a rectangular subset of cells in a table, given by the subsets of columns and rows, into which they belong. The set of all (sensitive) entries is a vector over \(\mathbb{R}^m\). The possible vectors of sensitive entries together with the norm forms a Banach space. As next we describe, how the SQL query is converted into a function from \(\mathbb{R}^m\) to \(\mathbb{R}\). In Sec. 5.2 and 5.3 we explain, how we compute a smooth upper bound of the derivative sensitivity of this function.

#### 5.1.1 Query without a Filter

To make a query on the database, we want to join those \(n\) tables. Consider an input \((t_1,\ldots,t_n) \in D\). Let us first consider the cross product \(t = t_1 \times \cdots \times t_n\), i.e. joining without any filters. First, let us assume that the \(n\) joined tables are distinct, i.e. no table is used more than once. Each row of the cross product is an element of \(R = R_1 \times \cdots \times R_n\), thus \(t\) is an element of \(T = R^{n_1\cdots n_n}\).

The query contains an aggregating function \(f : T \to \mathbb{R}\). All non-sensitive entries of the data tables are treated as constants. Suppose we want to compute the sensitivity of \(f\) w.r.t. a subset \(s\) of rows of \(t_i\). Then the subset of rows of the cross product, filtered through \(s\), is \(t_s = t_1 \times \cdots \times t_{i-1} \times s \times t_{i+1} \times \cdots \times t_n\). Each row in \(t_s\) is depends from exactly one row in \(s\).

Let \(s = r_1,\ldots,r_k\) and let \(t_s = \bigcup_{j=1}^k t_{r_j}\) where \(t_{r_j}\) is the subset of rows that depend from the row \(r_j\). The sets of rows \(t_{r_j}\) are disjoint. Let \(t_{r_j} = \{u_{j1},\ldots,u_{jm_j}\}\). For each row \(u_{jk}\), we select the same (semi)norm as the norm for the row \(t_{r_j}\), with the additional columns not contributing to the norm. The norm for \(t_{r_j}\) is computed by combining the norms for \(u_{j1},\ldots,u_{jm_j}\) using \(\ell_\infty\)-norm. The norm for \(t_s\) is then computed by combining the norms for \(t_{r_j}\) using the norm that combined the norms of the rows of \(t_i\), i.e. \(\ell_p\).

#### 5.1.2 Query with a Filter

A filter that does not depend on sensitive data can be applied directly to the cross product of the input tables, and we may then proceed with the query without a filter. A filter that does depend on sensitive data is treated as a part of the query. We treat this filter as a continuous function, applied in such a way that the discarded rows would be ignored by the aggregating function. We combine sigmoids and tauoids to obtain the approximated value of the indicator \(\sigma(x_i) \in \{0,1\}\), denoting whether the row \(x_i\) satisfies the filter.
Hence, if the filter depends on sensitive data, we have the following set-up:
- There is a set of rows \( \{x_1, \ldots, x_m\} \).
- There is a function \( f_i \) applied to the row \( x_i \), returning a real number. For different rows, this function may be different, e.g. it may be determined by the public cells of the row.
- There is a filtering function \( \sigma_i \) applied to the row \( x_i \). It returns a real number. It approximates a boolean condition, i.e. its values are mostly near 0 and 1.
- There is an aggregation function applied to a subset of the values \( f_1(x_1), \ldots, f_m(x_m) \). Only such \( i \in \{1, \ldots, m\} \) are selected, where the condition holds.

To convert the SQL query into a continuous function, the functions \( f_i \) and \( \sigma_i \) are combined as follows, depending on the aggregation function:
- \( \text{SUM} \): The values \( f_i(\cdot) \) do not affect the sum, hence we compute the result as \( \sum_{i=1}^{m} f_i(x_i) \cdot \sigma_i(x_i) \).
- \( \text{COUNT} \): The values \( f_i(\cdot) \) do not affect the result. We compute the result as \( \sum_{i=1}^{m} \sigma_i(x_i) \), counting all entries for which \( \sigma_i(x_i) = 1 \). The sensitivity of such query only depends on the sensitivity of \( \sigma \).
- \( \text{MIN, MAX} \): If \( \sigma_i(x_i) = 0 \), then we need to replace the actual value \( f_i(x_i) \) with some large [resp. small] value that would not affect the result of MIN [resp. MAX]. Our conversion of the SQL query proceeds by first defining \( \Delta := \text{MAX}(f(x_1), \ldots, f(x_n)) - \text{MIN}(f(x_1), \ldots, f(x_n)) \), and then computing the result by applying MIN to the values \( f_i(x_i) + (1 - \sigma_i(x_i)) \cdot \Delta \). MAX is computed similarly, changing the first “+” into a “−”.

If we know that the compared values are integers and hence \( d(x, x') \geq 1 \) for \( x \neq x' \), we can do better than using sigmoids or tauoids from Table 1, defining precise functions:
- \( x > y \iff \min(1, \max(0, x - y)) \).
- \( x = y \iff 1 - \min(1, \max(0, |x - y|)) \).

An advantage of these functions is that they do not lose precision due to addition and multiplication.

For real numbers, we may bound the precision and assume e.g. that \( d(x, x') \geq 1/k \) for some \( k \geq 1 \), which allows to use similar functions. The sensitivity of such comparisons will be \( k \) times larger than for integers.

### 5.2 Inferring Derivative Sensitivities

Previously we saw, how to convert an SQL query into a continuous function from \( \mathbb{R}^m \) to \( \mathbb{R} \), where each variable refers to the contents of a particular cell in a table of the database. Using the lemmas in Sec. 4.3, we can compute smooth upper bounds to their derivative sensitivities. The results given there can be specified to the concrete computation steps arising in the conversions of SQL queries. These specifications are given below; they are used to convert a SQL query to another query that computes the smooth upper bound of the derivative sensitivity of the original query. Examples of the whole conversion workflow can be seen in App. B, where the database schema is described in App. B.1, the privacy-sensitive parts of the database in App. B.2, the considered SQL queries in App. B.3, and, finally, the conversion results in App. B.5. These conversion results include both the conversion of the original query to a query with smooth semantics, and the query that computes an upper bound on the derivative sensitivity.

Let the write-up DS[\( f \)] \( \leq_N h \sim \beta \) mean that \( h \) is a \( \beta \)-smooth upper bound on the derivative sensitivity of \( f \), according to the norm \( \|\cdot\|_N \) on the domain of \( f \). For compositions, we also need the upper bounds for the (absolute values of) functions \( f \) themselves; let \( f \sim_N \beta \) denote that \( f \) is \( \beta \)-smooth, and \( f \leq g \sim_N \beta \) denote that \( g \) is a \( \beta \)-smooth upper bound of \( |f| \), again according to the norm \( \|\cdot\|_N \) on the domain of \( f \). Table 2 lists the smooth upper bounds of some simple uni- and multivariate functions and their derivative sensitivities (using absolute value as the norm on \( \mathbb{R} \)). These upper bounds are proved in App. C.8. For composite functions, the rules for computing the \( \beta \)-smooth upper bounds are given in Fig. 1. This summarizes the statements of lemmas of Sec. 4.3, which are proven in App. C.9.

### 5.3 Query Norm vs Database Norm

The facts and rules in Table 2 and Fig. 1 are in principle sufficient to compute the smooth upper bounds of derivative sensitivity of functions resulting from SQL queries with respect to all composite norms. Still, when we naïvely apply them, we end up finding the sensitivity for a particular norm that is somehow “natural” for the function. In practice, it may happen that we actually need sensitivity w.r.t. some different norm, because the data owner specified so. For example, we know how to compute the sensitivity w.r.t. the norm \( \|x_1, x_2\|_1 \), but are interested in differential privacy w.r.t. \( \|x_1, x_2\|_2 \).
Let the query norm (denoted $N_{qr}$) be the norm for which we can compute derivative sensitivity. Let the database norm (denoted $N_{db}$) be the norm for which we want to compute derivative sensitivity.

If $N_{qr} \preceq N_{db}$, then the rule ($\succeq D$) allows us to use the computed $DS[f]$ for $N_{qr}$ also with the norm $N_{db}$. But if $N_{qr} \npreceq N_{db}$ then we cannot directly use the sensitivity w.r.t. $N_{qr}$.

According to rule (ND), if $a$ is the upper bound to the derivative sensitivity of the function $f$ is $\beta$-smooth according to $N_{qr}$, then, its $\frac{1}{\alpha}$-times scaled version is $\beta\alpha$-smooth in according to the norm $\alpha \cdot N_{qr}$ for any $\alpha > 0$.

We compute sensitivity w.r.t. such norm $\alpha \cdot N_{qr}$, that $\alpha \cdot N_{qr} \preceq N_{db}$. The sensitivity becomes $\frac{\alpha}{\beta}$-smooth instead of $\beta$-smooth, which affects the amount of noise required to achieve differential privacy.

We show that a suitable $\alpha$ always exists for a composite norm (Def. 11) if $N_{db}$ uses all the variables $x_1, \ldots, x_n$. This assumption is reasonable: any variable that $N_{db}$ does not use is not treated as sensitive, so we may treat it as a constant when computing the sensitivity, reducing the total number of variables. We can find $\alpha$ as follows.

1. Use Lemma 2 to get $a_1, b_1 \geq 0$, $p, q > 0$ satisfying the conditions $\|a_1 x_1, \ldots, a_n x_n\| \geq x_1, \ldots, x_n \|N_{qr}$ and $\|b_1 x_1, \ldots, b_n x_n\| \leq x_1, \ldots, x_n \|N_{db}$. We have $a \cdot \|x_1, \ldots, x_n\|_p \geq \|a_1 x_1, \ldots, a_n x_n\|_p$ for $a = \max(a_1, \ldots, a_n)$, and $b \cdot \|x_1, \ldots, x_n\|_p \leq \|b_1 x_1, \ldots, b_n x_n\|_p$ for $b = \min(b_1, \ldots, b_n)$.

We get $a \cdot \|x_1, \ldots, x_n\|_p \geq \|x_1, \ldots, x_n\|_{N_{qr}}$. Since $N_{db}$ uses all variables $x_1, \ldots, x_n$, we have $b_1 \leq 0$ for all $i$, and hence $b \neq 0$. This allows to write $\|x_1, \ldots, x_n\|_p \leq b \cdot \|x_1, \ldots, x_n\|_{N_{db}}$.

2. If $p \leq q$, we have $a \cdot \|x_1, \ldots, x_n\|_q \leq a \cdot \|x_1, \ldots, x_n\|_p$. If $p > q$, we can use equivalence of $\ell_p$-norms that gives us $a \cdot \|x_1, \ldots, x_n\|_q \leq n^{1/q-1/p} a \cdot \|x_1, \ldots, x_n\|_p$. Let $c = (p \leq q) \cdot n^{1/q-1/p}$. Then $a \cdot \|x_1, \ldots, x_n\|_p \leq c \cdot \|x_1, \ldots, x_n\|_{N_{qr}}$.

3. We have now come up with scalings $a, b, c$ that satisfy $c \cdot a \cdot \|x_1, \ldots, x_n\|_p \geq \|x_1, \ldots, x_n\|_{N_{qr}}$, and $\|x_1, \ldots, x_n\|_p \leq \frac{1}{c} \cdot \|x_1, \ldots, x_n\|_{N_{db}}$. Putting these inequalities together, we get $c \cdot a \cdot \frac{1}{c} \cdot \|x_1, \ldots, x_n\|_{N_{db}} \geq \|x_1, \ldots, x_n\|_{N_{qr}}$. By construction, we always have $c > 0$. It is possible that $a = 0$ only in the case...

Here $\text{pw}^\alpha_p(x) = \left(\frac{\alpha}{\beta}\right)^p \cdot e^{3x}.$

**Table 2.** Upper bounds for uni- and multivariate functions

<table>
<thead>
<tr>
<th>$f(x)$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \cdot N_{qr} \preceq N_{db}$</th>
<th>$\alpha \cdot N_{qr}$</th>
<th>$\alpha \cdot N_{db}$</th>
<th>$\alpha \cdot N_{db}$</th>
<th>$\alpha \cdot N_{db}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e^x$</td>
<td>$</td>
<td>x</td>
<td>$</td>
<td>$</td>
<td>x</td>
<td>$</td>
<td>$e^x$</td>
</tr>
<tr>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
</tr>
<tr>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
<td>$\frac{e^x}{x^2+1}$</td>
</tr>
</tbody>
</table>

**Fig. 1.** Upper bounds for composite functions
6 Precision and Utility

6.1 Choosing the Norm and $\epsilon$

In the standard definition of differential privacy, we are concealing an addition/removal of one table row. In a Banach space, the notion of unit change can be different. Even if we have decided on the set of sensitive rows and columns, it may be unclear whether/which scaling of norm variables is reasonable. For example, if we scale the norm by $a$ and keep noise level the same, the $\epsilon$ will increase proportionally to $a$, so we need to know which $\epsilon$ is "good enough" to choose appropriate $a$. For this, we need to understand what the table attributes mean. For example, if the length is presented in meters, and we want to conceal a change in a kilometer, we scale the location norm by 0.001 to capture a larger change.

To give a better interpretation to $\epsilon$, we may relate it to other security definitions such as guessing probability advantage, as done e.g. in [22]. Adapting this approach to our metrics, we could answer questions like "how likely the attacker guesses that the location of an object is within $X$ miles from the actual location".

Let $X' \subseteq X$ be the subset of inputs for which we consider the attacker guess as successful (e.g. he guesses an object's location coordinates precisely enough). Let the posterior belief of the adversary (after seeing the output) be expressed by the probability distribution $Pr_{post}[:]. Let the prior belief (before seeing the output) be $Pr_{pre}[:].$ and $fx$ the corresponding probability density function, i.e. $Pr_{pre}[X'] = \int_{X'} f_X(x) dx.$ We need an upper bound on $Pr_{post}[X'] = Pr_{pre}[X' \mid y],$ where $y$ is the observed output. Let $fy$ be the probability density function of noisy outputs. Using Bayesian inference, $Pr_{post}[X'] = Pr_{pre}[X' \mid y] \leq \frac{1}{1 + \int_{X''} fy(y|x) f_X(x) dx} \int_{X''} fy(y|x') f_X(x') dx'$

for any $X'' \subseteq X \setminus X'$. Let $X'' = \{ x \mid r < d(x, x') \leq a \}$ for some $a \in \mathbb{R}$ (e.g. $a := \sup_{x \in X} d(x, x')$ if it exists).

Differential privacy gives us $Pr_{pre}[M_{x}(\epsilon') \in Y] \leq e^{-\epsilon a}$ for all $Y$, and hence also $\frac{fy(y|x)}{fy(y|x')} \leq e^{-\epsilon a}$. We get

$$Pr_{post}[X'] \leq \frac{1}{1 + e^{-\epsilon a} Pr_{pre}[X'] / Pr_{pre}[Y]}.$$ 

The optimal value of $a$ depends on the distribution of $X$. In practice, we may use brute force search for $a$.

6.2 Sigmoid Precision

In Sec. 5, we mentioned that we use sigmoids $\frac{e^{ax}}{e^{ax} + 1}$ to approximate filtering of the form $x \geq 0$, where $\alpha > 0$ can be arbitrary. The derivative $\frac{ae^{ax}}{(e^{ax} + 1)^2}$ of a sigmoid is $\alpha$-smooth.

To get a higher precision than that of an ordinary sigmoid but still maintain $\alpha$-smoothness, we use an extra parameter $a$ in addition to $\alpha$. We use the sigmoid $\sigma(x) = \frac{e^{ax}}{e^{ax} + 1}$ (note that the precision parameter is now $a$) but instead of its actual sensitivity $\sigma'(x)$, which would itself be $a$-smooth, we use $c(x) = \frac{ae^{ax}}{(e^{ax} + 1)^2}$, which is an $\alpha$-smooth upper bound on $\sigma'(x)$. The smooth sensitivity is $\frac{a}{\alpha}$ times higher than that of the original sigmoid but the difference from the precise filter value (0 or 1) is $\frac{ae^{ax}}{(e^{ax} + 1)^2}$ times smaller. If the probability density function of $x$ is roughly constant near $x = 0$ then, for all $y \in (0, 1)$, the probability that the difference from the precise filter value is larger than $y$ is

$$Pr(\frac{1}{e^{ax} + 1} > y) = Pr \left( x < \frac{\ln(\frac{1}{y} - 1)}{a} \right) \approx \frac{a}{\alpha} Pr(x < \frac{1}{a} \ln(\frac{1}{y} - 1)) = \frac{a}{\alpha} Pr(\frac{1}{e^{ax} + 1} > y)$$

i.e. $\frac{a}{\alpha}$ times smaller than in the original sigmoid.

The goodness of $a$ depends on both the query and the data. Since a sigmoid error affects each row, precision becomes more important when the number of rows grows large. An optimal $a$ will increase proportionally to $\sqrt{n}$, where $n$ is the number of rows. The details are given in App. A.

6.3 Comparing Laplace and Cauchy noise

While generalized Cauchy distributions have heavy tails, the heaviness can be reduced by increasing the parameter $\gamma$. Taking $\gamma = 4$ seems to give a good balance between tail heaviness and median absolute value of noise. The 99.9999% quantile of $\text{GenCauchy}(4)$ is about 120 times its median which does not seem too
much worse than \([Lap(1)]\), whose 99.9999% quantile is about 20 times its median.

Now let us compare noise magnitudes (which are roughly proportional to median absolute noise). When using Laplace noise, the smoothness \(\beta\) will need to be smaller (for a fixed \(\epsilon\)) than for generalized Cauchy noise. This will in general increase the \(\beta\)-smooth upper bound \(c(x)\). On the other hand, the value \(b\) will also be larger. Whether the noise magnitude \(\frac{c(x)}{b}\) will be smaller for Laplace or for generalized Cauchy noise, depends on the concrete query.

Using Laplace noise does not seem to have enough advantages over generalized Cauchy noise to justify the more complex properties and worse privacy guarantee of \((\epsilon, \delta)\)-DP over \(\epsilon\)-DP. Hence, in this paper we only evaluate generalized Cauchy noise.

7 Implementation and Evaluation

7.1 Implementation

Our analyzer (available on GitHub) has been implemented in Haskell. As an input, it takes an SQL query, a database schema, and a description of the norm w.r.t. which we want to achieve differential privacy. We assume that each table contains a row \(ID\) of unique keys. For each table \(X\), we expect a table named \(X\_sensRows\) that contains the same column \(ID\) of keys, and another column \(sensitive\) of boolean values that tell for each row whether it is sensitive or not.

The analyzer computes another query (as a string) that describes the way in which derivative sensitivity should be computed. This new query represents the function \(c(x)\) such that the additive noise would be \(\frac{c(x)}{b} \cdot \eta\) for \(\eta \leftarrow GenCauchy(\gamma)\), according to Theorem 3. In our analyzer, \(\gamma = 4\) is fixed (as justified in Sec. 6.3), and \(b = \epsilon / (\gamma + 1) - \beta\), where \(\epsilon\) is the desired differential privacy level, and \(\beta\) the smoothness parameter, which is provided as an additional input. The resulting query is fed to a database engine to evaluate the sensitivity on particular data.

7.2 Evaluation

We performed evaluation on 4 x Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz laptop, Ubuntu 16.04.4 LTS, using PostgreSQL 9.5.14.

We have taken the queries of TCP-H set [1] for benchmarking. Most of these queries contain GROUP BY constructions with too many possible groups. We have simplified these queries, adding a filtering that chooses one particular group.

Another challenge comes from the filters. If some filter is “public” (i.e. does not depend on sensitive data), it is easier to apply it beforehand, so that the remaining table with “private” filters (that do depend on sensitive data and hence cannot be applied directly) would be as small as possible. While it is easy to do with a pure AND combination of filters, in practice public and private filters can be mixed, e.g. related by OR. We had to manually rewrite the filters in such a way that public filters would be easily extractable as separate members of an AND combination.

We generated TCP-H data with scale factors (SF) 0.1, 0.5, 1.0, denoting how much data is generated for the sample database. For 1.0, the size of the largest table is ca 6 million rows. The table schema, together with numbers of rows for different tables, is given in App. B.1. To define the database metric, we have considered integer, decimal, and date columns as sensitive, assigning to them different weights, described more precisely in App. B.2. All rows are considered sensitive. Row norms have been combined using \(\ell_1\)-norm, which ensures differential privacy w.r.t. unit change in sensitive attribute of one row.

We adjusted (as described above) the queries \(Q1\) (splitting a single query with 5 aggregations to 5 separate queries), \(Q2\) (splitting it to 2 queries with MIN and MAX respectively), \(Q3\) to \(Q11\), \(Q12\) (splitting 2 aggregations to 2 queries), \(Q16\), \(Q17\), \(Q19\) of the TCP-H dataset to our analyzer. The queries that have been eventually fed to the analyzer are listed in App. B.3. We treat date as an integer, i.e. the number of days passed from the date 1980-01-01. In App. B.4, we present more evaluation results, where we treat date as a floating point number, so that sigmoids can be used for filtering.

We fix \(\epsilon = 1\). For derivative sensitivity experiments, we take sigmoid precision \(\alpha = 5\) and smoothness \(\beta = 0.1\). This choice gives \(b = 0.1\), and the additive noise with 78% probability is below \(10 \cdot c(x)\), where the value 78% comes from analyzing distribution \(GenCauchy(4)\) (we have \(\int_{-1}^{1} GenCauchy(4)(x) \, dx = 0.78\)). Too large value of \(\beta\) makes \(b\) (and hence the noise) larger, and too small \(\beta\) makes the sensitivity larger, so \(\beta\) is a parameter that can in general be optimized.
7.2.1 Time

The time benchmarks are given in Table 3. Let $x$ be the database instance. For each scale factor $SF$, we report the execution time $t_{i}$ of the initial query $q_{i}(x)$, time $t_{m}$ of the modified query $q_{m}(x)$ (i.e. in which filtering is replaced with continuous approximation), and time $t_{s}$ of the sensitivity-computing query $q_{s}(x)$. The time spent to generate the queries $q_{m}$ and $q_{s}$ is negligible (below 20ms), and it does not depend on the database size, so we do not report it. We also do not report the execution time of sampling the noise, as it does not depend on the database size either.

The total time overhead of computing noisy output based on derivative sensitivity is $t_{m} + t_{s}$: since the sensitivity has been computed for $q_{m}$, the noise should also be added to $q_{m}(x)$, and not to $q_{i}(x)$. We estimate the total time overhead for global sensitivity as $t_{i}$, as it is sufficient to execute $q_{i}(x)$, and the computation of global sensitivity does not depend on the database size.

We see that in general $t_{m}$ and $t_{s}$ are larger than $t_{i}$. This overhead comes from filtering. While in $q_{i}$ the database engine may immediately drop all rows that do not satisfy the filter, in $q_{m}$ and $q_{s}$ we need to compute the approximated output and the sensitivity of each row. In overall, the time overhead of $q_{m}$ and $q_{s}$ compared to $q_{i}$ (and hence of derivative sensitivity compared to global sensitivity) depends on the ratio of “number of rows before filtering” and “number of rows after filtering”.

7.2.2 Precision

The precision benchmarks are given in Table 4. For each scale factor $SF$, we report the output $q_{i}(x)$ of the initial query, and $q_{s}(x)$ of the sensitivity-computing query. We report the output $q_{m}(x)$ of the modified query only if it is different from $q_{i}(x)$. The relative error has been computed as $\frac{|q_{m}(x) - q_{i}(x)|}{q_{i}(x)} \cdot 100$, where $\xi = \frac{c(x)}{b} = \frac{q_{s}(x)}{c(x)} = 0.1 \cdot q_{s}(x)$. The additive noise stays below $\xi$ with probability 78% (as discussed above), so the relative error stays below reported value also with probability 78%.

The last two columns of Table 4 demonstrate the global sensitivity of queries, which is the same for all SF values, as it does not depend on data. The left column shows global sensitivity w.r.t. the same metric as the derivative sensitivity (we call it non-standard), and the right column w.r.t. the row difference metric (we call it standard). We compare these with derivative sensitivity.

Global sensitivity w.r.t. non-standard metric. In the first case, we compute the global sensitivity w.r.t. the same metric as the derivative sensitivity. Even using the same metric, we cannot compare global (GS) and derivative sensitivity (DS) directly without taking into account particular noise generating mechanisms. However, in our results we have either GS=$\infty$ or GS=DS. If GS=$\infty$, then the noise would be $\infty$ as well for any noise generating mechanism. If GS=DS, then we expect the noise of GS to be lower, as e.g. employing the same Cauchy mechanism that we use for derivative sensitivity with $\beta \approx 0$ gives 10 times less noise than with $\beta = 0.1$ for the same sensitivity. In our benchmarks, DS gives advantage over GS in the following main cases.

1. When a sensitive attribute $x_{1}$ is multiplied by another attribute $x_{2}$, and there are no bounds on $x_{2}$, we get GS=$\infty$, as $|(x_{1} \pm 1) \cdot x_{2} - x_{1} \cdot x_{2}| = |x_{2}|$.

2. The norms of rows are combined into a table norm using $\ell_{1}$-norm. Hence, $d(t,t') = 1$ covers not only the case where the norm of one row changes by 1, but also the case where each row changes a little. In an extreme case, all rows of $t$ are already very close to the filtering bound, and the filtering function returns $\approx 1$ for all rows in $t$, and $\approx 0$ for all rows in $t'$. This makes no difference for a COUNT query (as in b1_5), as the sum of all these changes is still 1, but we get GS=$\infty$ for the query b1_1, which has the same form as b1_5, except that it is a SUM query.

Global sensitivity w.r.t. standard metric. In the second case, we compute global sensitivity w.r.t. row difference metric. That is, $d(x,x') = 1$ iff there is exactly one sensitive table in databases $x$ and $x'$ such that the respective instances $t$ and $t'$ of that table differ in one row. To make the comparison more fair, we consider an input table sensitive iff the query uses at least one of its attributes that were considered sensitive by the $\ell_{p}$-metric. For SUM, MIN, MAX queries, the effect of adding/removing a row is unbounded, and global sensitivity is $\infty$, as it covers the worst case. For COUNT queries, we may lose advantage as well if we consider a JOIN of tables, where adding/removing a row in an input table may result in adding/removing an unbounded number of rows in the cross product of input tables, as it happens in b4. Row difference metric gives smaller sensitivity in the COUNT-queries b12_1, b12_2, b16. In general, if we filter by a sensitive attribute over a single input table, then row difference metric contributes 1 to the COUNT, while defining the distance as $\ell_{1}$-norm of rows allows to split the unit change among several rows, which may result in higher sensitivity.
<table>
<thead>
<tr>
<th></th>
<th>SF = 0.1</th>
<th></th>
<th>SF = 0.5</th>
<th></th>
<th>SF = 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t_i$</td>
<td>$t_m$</td>
<td>$t_s$</td>
<td>$t_i$</td>
<td>$t_m$</td>
</tr>
<tr>
<td>$b1_1$</td>
<td>152.8</td>
<td>534.59</td>
<td>763.19</td>
<td>731.11</td>
<td>3.17K</td>
</tr>
<tr>
<td>$b1_2$</td>
<td>151.8</td>
<td>559.58</td>
<td>1.04K</td>
<td>1.47K</td>
<td>4.02K</td>
</tr>
<tr>
<td>$b1_3$</td>
<td>168.08</td>
<td>590.1</td>
<td>2.05K</td>
<td>862.07</td>
<td>3.24K</td>
</tr>
<tr>
<td>$b1_4$</td>
<td>184.24</td>
<td>574.28</td>
<td>2.2K</td>
<td>880.35</td>
<td>2.98K</td>
</tr>
<tr>
<td>$b1_5$</td>
<td>149.96</td>
<td>527.38</td>
<td>520.85</td>
<td>744.5</td>
<td>2.69K</td>
</tr>
<tr>
<td>$b2_1$</td>
<td>19.68</td>
<td>45.3</td>
<td>144.78</td>
<td>134.21</td>
<td>294.14</td>
</tr>
<tr>
<td>$b2_2$</td>
<td>29.04</td>
<td>49.37</td>
<td>165.62</td>
<td>158.94</td>
<td>273.06</td>
</tr>
<tr>
<td>$b3$</td>
<td>111.92</td>
<td>117.41</td>
<td>391.47</td>
<td>544.22</td>
<td>623.87</td>
</tr>
<tr>
<td>$b4$</td>
<td>131.52</td>
<td>379.05</td>
<td>774.47</td>
<td>799.9</td>
<td>2.63K</td>
</tr>
<tr>
<td>$b5$</td>
<td>6.66K</td>
<td>204.08</td>
<td>2.18K</td>
<td>696.38</td>
<td>685.59</td>
</tr>
<tr>
<td>$b6$</td>
<td>118.31</td>
<td>3.12K</td>
<td>13.21K</td>
<td>687.4</td>
<td>16.09K</td>
</tr>
<tr>
<td>$b7$</td>
<td>238.74</td>
<td>137.21</td>
<td>713.28</td>
<td>1.19K</td>
<td>861.55</td>
</tr>
<tr>
<td>$b8$</td>
<td>308.08</td>
<td>117.53</td>
<td>782.37</td>
<td>1.3K</td>
<td>1.73K</td>
</tr>
<tr>
<td>$b9$</td>
<td>133.34</td>
<td>128.58</td>
<td>3.82K</td>
<td>1.79K</td>
<td>728.07</td>
</tr>
<tr>
<td>$b10$</td>
<td>131.97</td>
<td>137.03</td>
<td>483.38</td>
<td>882.12</td>
<td>719.65</td>
</tr>
<tr>
<td>$b11$</td>
<td>10.74</td>
<td>10.16</td>
<td>42.12</td>
<td>62.0</td>
<td>62.02</td>
</tr>
<tr>
<td>$b12_1$</td>
<td>215.13</td>
<td>736.64</td>
<td>1.27K</td>
<td>879.2</td>
<td>3.65K</td>
</tr>
<tr>
<td>$b12_2$</td>
<td>148.5</td>
<td>473.72</td>
<td>877.42</td>
<td>846.66</td>
<td>3.26K</td>
</tr>
<tr>
<td>$b16$</td>
<td>22.14</td>
<td>174.35</td>
<td>303.68</td>
<td>127.95</td>
<td>711.93</td>
</tr>
<tr>
<td>$b17$</td>
<td>111.7</td>
<td>88.31</td>
<td>276.69</td>
<td>486.16</td>
<td>455.85</td>
</tr>
<tr>
<td>$b19$</td>
<td>139.16</td>
<td>296.41</td>
<td>1.42K</td>
<td>737.53</td>
<td>1.47K</td>
</tr>
</tbody>
</table>

Table 3. Time benchmarks (ms) for the initial query ($t_i$), modified query ($t_m$), and the sensitivity query ($t_s$). $K$ denotes $10^3$.

<table>
<thead>
<tr>
<th>$q_i(x)$</th>
<th>$q_m(x)$</th>
<th>%noise</th>
<th>$q_i(x)$</th>
<th>$q_m(x)$</th>
<th>%noise</th>
<th>$q_i(x)$</th>
<th>$q_m(x)$</th>
<th>%noise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b1_1$</td>
<td>3.79M</td>
<td>50.0</td>
<td>0.01</td>
<td>18.87M</td>
<td>50.0</td>
<td>0.0026</td>
<td>37.72M</td>
<td>50.0</td>
</tr>
<tr>
<td>$b1_2$</td>
<td>5.34G</td>
<td>95.89K</td>
<td>0.02</td>
<td>27.35G</td>
<td>99.65K</td>
<td>0.0036</td>
<td>56.57G</td>
<td>104.9K</td>
</tr>
<tr>
<td>$b1_3$</td>
<td>5.07G</td>
<td>107.36K</td>
<td>0.02</td>
<td>25.98G</td>
<td>111.18K</td>
<td>0.0043</td>
<td>53.74G</td>
<td>117.34K</td>
</tr>
<tr>
<td>$b1_4$</td>
<td>5.27G</td>
<td>114.87K</td>
<td>0.02</td>
<td>27.02G</td>
<td>119.06K</td>
<td>0.0044</td>
<td>55.89G</td>
<td>124.38K</td>
</tr>
<tr>
<td>$b1_5$</td>
<td>148.3K</td>
<td>1.0</td>
<td>0.0067</td>
<td>739.56K</td>
<td>1.0</td>
<td>0.0014</td>
<td>1.84M</td>
<td>1.0</td>
</tr>
<tr>
<td>$b2_1$</td>
<td>1.07</td>
<td>100.0</td>
<td>93.46K</td>
<td>1.0</td>
<td>100.0</td>
<td>100.0K</td>
<td>1.0</td>
<td>100.0</td>
</tr>
<tr>
<td>$b2_2$</td>
<td>999.98</td>
<td>100.0</td>
<td>1.0K</td>
<td>100.0</td>
<td>100.0</td>
<td>1.0K</td>
<td>1.0</td>
<td>100.0</td>
</tr>
<tr>
<td>$b3$</td>
<td>3.62K</td>
<td>41.28K</td>
<td>11.4K</td>
<td>3.21K</td>
<td>41.1K</td>
<td>12.8K</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$b4$</td>
<td>2.92K</td>
<td>7.0</td>
<td>2.4</td>
<td>14.17K</td>
<td>7.0</td>
<td>0.49</td>
<td>28.07K</td>
<td>7.0</td>
</tr>
<tr>
<td>$b5$</td>
<td>5.37M</td>
<td>260.44K</td>
<td>48.53</td>
<td>25.23M</td>
<td>359.6K</td>
<td>14.25</td>
<td>47.6M</td>
<td>484.12K</td>
</tr>
<tr>
<td>$b6$</td>
<td>17.45M</td>
<td>125.0K</td>
<td>7.14</td>
<td>88.13M</td>
<td>127.0K</td>
<td>1.44</td>
<td>181.93M</td>
<td>130.0K</td>
</tr>
</tbody>
</table>

Table 4. Precision benchmarks for $c = 1$, $\beta = 0.1$, sigmoid $\alpha = 5$, where $q_i(x)$ is the initial query result, $q_m(x)$ the modified query result (if different from $q_i(x)$), $q_s(x)$ is the sensitivity query result, and %noise = $\frac{\text{noise}}{q_i(x)}$. The last two columns show global sensitivity w.r.t. the same non-standard $\ell_p$-induced metric as derivative sensitivity (non-std.) and the standard “row difference” metric (std.). $K$ denotes $10^3$, $M$ denotes $10^6$, and $G$ denotes $10^9$.
8 Discussion

Let us summarize the limits and advantages of the framework proposed in this paper. We compare ℓp-metric vs row distance metric, and local sensitivity vs global sensitivity. In the following, we mark with + the clear advantages, and with − some caveats.

Applicability.
+ Metrics induced by ℓp-norms allow to state different privacy goals, and can be useful in cases where the standard row distance metric is not applicable.
− Computation of derivative sensitivity requires a particular data instance. This is similar to local sensitivity. Since execution of the sensitivity-computing query can be deferred, and the data will anyway be needed at the point where a noisy output is released, we do not treat it as an applicability issue.
− Derivative sensitivity is limited to continuous functions. This is not a problem as far as there exist efficiently computable approximations. We can still cover a wide range of SQL queries.

Complexity.
+ In the first phase of the analysis, the initial query it transformed to sensitivity-computing query. The execution time of this transformation is negligible and does not depend on the data.
− In the second phase of the analysis, when the output is ready to be released, we need to execute the sensitivity-computing query to estimate amount of noise. Compared to the initial query, additional time overhead comes from filtering, as we cannot ignore the rows that have been discarded by the filter.

Amount of noise
+ Changing a numeric attribute of a row in general has smaller effect on the query result than adding/removing an entire row.
+ As global sensitivity always covers the worst-case data instance, it is in general larger than local and derivative sensitivity.
− Compared to global sensitivity w.r.t. standard metric, there are more parameters to be tuned in order to optimize the amount of noise, such as smoothness and sigmoid precision.
− While ℓp-norms allow to define a variety of metrics over databases, they are not a superset of standard metrics, and for some privacy goals we can get less noise using standard metric.

Possible improvements. Adding noise before filtering is the path towards solving the issue of complexity and noise overhead that comes from filtering over sensitive attributes. While we believe that our framework allows to locate the points where the noise has to be added, this is not the topic of the current paper.

So far, similarly to Blowfish [17], we have assumed that the number of rows in the tables is fixed. It is actually possible to define the derivative sensitivity w.r.t. number of rows, treating it as a real number. Our framework would then cover the row difference metric as well. We defer this research to future work.

9 Conclusion

We have started the study of complete norms to define the quantitative privacy properties of information release mechanisms, and have discovered their high expressivity for different kinds of numeric inputs, as well as the principles of the parallel composition of norms in a manner that allows the sensitivity of the information release mechanism to be found. Our results show how the similarity of local sensitivity and the derivative can be exploited in constructing differentially private mechanisms. The result is also practically significant because of the need to precisely model the privacy requirements of data owner(s), for which the flexibility of specifying the metric over possible inputs is a must.

Our results open up the study of the combinations of metrics over more varied types of input data, including categorical and structured data, or data with consistency constraints. Such study would also look for possibilities to express the constraints through suitable combinations of metrics over components. We note that inputs with constraints [20] or with particular structure (sequences indexed by time points) [7, 10] have been considered in the literature. We hope that it is possible to find suitable complete norms that define the metrics used in the privacy definitions for such data, and thereby express these constructions inside our framework.

Acknowledgements. This research has been funded by Estonian Research Council, grant no. IUT27-1, by ERDF through the Centre of Excellence EXCITE, and by the Air Force Research laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under contract FA8750-16-C-0011. The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government.
References


A Sigmoid Precision

Using sigmoids instead of the original filters causes a precision loss, as we use the value \( \frac{e^{\alpha x}}{e^{\alpha x} + 1} \) instead of 0 or 1. Suppose we have a SUM query using \( \ell_1 \)-norm to join row norms and input rows are chosen from a certain distribution that does not change when input size \( n \) changes, which would be a quite common scenario. Then the relative error from added noise goes to 0 but the relative error from sigmoids is roughly constant as \( n \to \infty \). Thus the total error does not go to zero as \( n \to \infty \).

To improve on this, we replace sigmoids with precise sigmoids, keeping \( \alpha \) constant but increasing the second parameter \( a \) as \( n \) increases, allowing a tradeoff that (assuming the probability density function of the input is roughly constant near the pivot point of the filter) increases the relative error from added noise \( \sqrt{n} \) times and decreases that from sigmoids \( \sqrt{n} \) times, making the total error inversely proportional to \( \sqrt{n} \), thus going to zero as \( n \to \infty \). This also holds if more than one sigmoid is used. If no sigmoids are used then no tradeoff is needed and the total error inversely proportional to \( n \).

We can get similar results for tauoids. If the filtered values are integers then we do not have to use sigmoids and tauoids but instead use the precise functions as described before. Then the only error is from adding noise and it would be inversely proportional to \( n \).

B Evaluation Details

B.1 Database Schema

The TPC-H testset [1] puts forth the following database schema, as given below. The tables are (randomly) filled with a number of rows, generated by a program that accompanies the schema. The number of rows depends on the scaling factor \( SF \). The tables, and the numbers of rows in them are the following:

**Part: \( SF \cdot 200,000 \) rows.**

<table>
<thead>
<tr>
<th>column</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PARTKEY</td>
<td>identifier</td>
</tr>
<tr>
<td>P_NAME</td>
<td>text</td>
</tr>
<tr>
<td>P_MFGR</td>
<td>text</td>
</tr>
<tr>
<td>P_BRAND</td>
<td>text</td>
</tr>
<tr>
<td>P_TYPE</td>
<td>text</td>
</tr>
<tr>
<td>P_SIZE</td>
<td>integer</td>
</tr>
<tr>
<td>P_CONTAINER</td>
<td>text</td>
</tr>
<tr>
<td>P_RETAILPRICE</td>
<td>decimal</td>
</tr>
<tr>
<td>P_COMMENT</td>
<td>text</td>
</tr>
</tbody>
</table>

**Customer: \( SF \cdot 150,000 \) rows.**

<table>
<thead>
<tr>
<th>column</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_CUSTKEY</td>
<td>identifier</td>
</tr>
<tr>
<td>C_NAME</td>
<td>text</td>
</tr>
<tr>
<td>C_ADDRESS</td>
<td>text</td>
</tr>
<tr>
<td>C_NATIONKEY</td>
<td>identifier</td>
</tr>
<tr>
<td>C_PHONE</td>
<td>text</td>
</tr>
<tr>
<td>C_ACCTBAL</td>
<td>decimal</td>
</tr>
<tr>
<td>C_MKTSEGMENT</td>
<td>text</td>
</tr>
<tr>
<td>C_COMMENT</td>
<td>text</td>
</tr>
</tbody>
</table>

**Partsupp: \( SF \cdot 800,000 \) rows.**

<table>
<thead>
<tr>
<th>column</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS_PARTKEY</td>
<td>identifier</td>
</tr>
<tr>
<td>PS_SUPPKEY</td>
<td>identifier</td>
</tr>
<tr>
<td>PS_AVAILQTY</td>
<td>integer</td>
</tr>
<tr>
<td>PS_SUPPLYCOST</td>
<td>decimal</td>
</tr>
<tr>
<td>PS_COMMENT</td>
<td>text</td>
</tr>
</tbody>
</table>
A Framework of Metrics for Differential Privacy from Local Sensitivity

### B.2 Sensitive Components

In all tables except `Lineitem`, we consider the change that is the scaled sum of changes in all sensitive attributes. All attributes that are not a part of the norm are considered insensitive. We assumed that textual fields as well as the keys (ordinal data) are not sensitive. The columns of type `date` (e.g. `o_orderdate`) have been converted to a floating-point number, which is the number of months passed from the date 1980-01-01.

- **Part**: \[|p_{size} \cdot 0.01 \cdot p_{retailprice}|\]. The values of `p_retailprice` are measured in hundreds, so we consider larger changes (i.e. make such change causing a change of 1 in the output correspond to unit sensitivity).
- **Partsupp**: \[|ps_{available} \cdot 0.01 \cdot ps_{supplycost}|\].
- **Orders**: \[30 \cdot o_{orderdate} \cdot 0.01 \cdot o_{totalprice}|\].
- **Customer**: \[|0.01 \cdot e_{acctbal}|\].
- **Supplier**: \[|0.01 \cdot s_{acctbal}|\].
- **Region**: no sensitive columns.

### B.3 Benchmark Queries

We list the rewritten queries of TCP-H dataset that were used in benchmarking, to give an impression of what we actually feed to the analyzer. The constant 0.142857 in `b_17` comes from translating an `AVG` query to a `SUM` query, and \(7 = 1/0.142857\) is the number of rows to sum, which is public, as the filter does not use sensitive attributes.
--b1_1
SELECT SUM(lineitem.l_quantity)
FROM lineitem
WHERE
    lineitem.l_shipdate <= 230.3 - 30
    AND lineitem.l_returnflag = 'R'
    AND lineitem.l_linenumber = 'F'
;
--b1_2
SELECT SUM(lineitem.l_extendedprice)
FROM lineitem
WHERE
    lineitem.l_shipdate <= 230.3 - 30
    AND lineitem.l_returnflag = 'R'
    AND lineitem.l_linenumber = 'F'
;
--b1_3
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount))
FROM lineitem
WHERE
    lineitem.l_shipdate <= 230.3 - 30
    AND lineitem.l_returnflag = 'R'
    AND lineitem.l_linenumber = 'F'
;
--b1_4
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount) * (1 + lineitem.l_tax))
FROM lineitem
WHERE
    lineitem.l_shipdate <= 230.3 - 30
    AND lineitem.l_returnflag = 'R'
    AND lineitem.l_linenumber = 'F'
;
--b2_1
SELECT MIN(partsupp.ps_supplycost)
FROM partsupp, supplier, nation, region, part
WHERE
    part.p_partkey = partsupp.ps_partkey
    AND supplier.s_suppkey = partsupp.ps_suppkey
    AND supplier.s_nationkey = nation.n_nationkey
    AND nation.n_regionkey = region.r_regionkey
    AND region.r_name = 'ASIA'
;
--b3
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount))
FROM customer, orders, lineitem
WHERE
    customer.c_mktsegment = 'BUILDING'
    AND customer.c_custkey = orders.o_custkey
    AND lineitem.l_orderkey = orders.o_orderkey
    AND orders.o_orderdate < 190
    AND lineitem.l_shipdate > 190
    AND lineitem.l_orderkey = '162'
    AND orders.o_shippriority = '0'
;
--b4
SELECT COUNT(*)
FROM orders, lineitem
WHERE
    orders.o_orderdate >= 180
    AND orders.o_orderdate < 180 + 3
    AND lineitem.l_orderkey = orders.o_orderkey
    AND lineitem.l_commitdate < lineitem.l_receiptdate
    AND orders.o_orderpriority = '1-URGENT'
;
--b5
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount))
FROM customer, orders, lineitem, supplier, nation, region
WHERE
    customer.c_custkey = orders.o_custkey
    AND lineitem.l_orderkey = orders.o_orderkey
    AND lineitem.l_suppkey = supplier.s_suppkey
    AND customer.c_nationkey = supplier.s_nationkey
    AND supplier.s_nationkey = nation.n_nationkey
    AND nation.n_regionkey = region.r_regionkey
    AND region.r_name = 'ASIA'
    AND orders.o_orderdate >= 213.3
    AND orders.o_orderdate < 213.3 + 12
    AND nation.n_name = 'JAPAN'
;
--b6
SELECT SUM(lineitem.l_extendedprice * lineitem.l_discount)
FROM lineitem
WHERE
    lineitem.l_shipdate >= 170.5
    AND lineitem.l_shipdate < 170.5 + 12
    AND lineitem.l_discount BETWEEN 0.09 - 0.01
    AND 0.09 + 0.01
    AND lineitem.l_quantity < 24
;
--b7
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount))
FROM lineitem
WHERE
    lineitem.l_shipdate <= 170.5
    AND lineitem.l_shipdate < 170.5 + 12
    AND lineitem.l_discount BETWEEN 0.09 - 0.01
    AND 0.09 + 0.01
    AND lineitem.l_quantity < 24
FROM supplier, lineitem, orders, customer, nation as n1, nation as n2
WHERE
supplier.s_suppkey = lineitem.l_suppkey
AND orders.o_orderkey = lineitem.l_orderkey
AND customer.c_custkey = orders.o_custkey
AND supplier.s_nationkey = n1.n_nationkey
AND customer.c_nationkey = n2.n_nationkey
AND (n1.n_name = 'JAPAN' and n2.n_name = 'INDONESIA')
OR (n1.n_name = 'INDONESIA' and n2.n_name = 'JAPAN')
AND lineitem.l_shipdate between 182.6 and 207
;
--b8
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount))
FROM part, supplier, lineitem, orders, customer,
nation AS n1, nation AS n2, region
WHERE
part.p_partkey = lineitem.l_partkey
AND supplier.s_suppkey = lineitem.l_suppkey
AND lineitem.l_orderkey = orders.o_orderkey
AND orders.o_custkey = customer.c_custkey
AND customer.c_nationkey = n1.n_nationkey
AND n1.n_regionkey = region.r_regionkey
AND region.r_name = 'ASIA'
AND supplier.s_suppkey = n2.n_nationkey
AND orders.o_orderdate >= 5478
AND orders.o_orderdate <= 6210
AND part.p_type = 'MEDIUM BRUSHED COPPER'
AND n2.n_name = 'JAPAN'
;
--b9
SELECT SUM(lineitem.l_extendedprice * (1 - lineitem.l_discount))
- partsupp.ps_supplycost * lineitem.l_quantity)
FROM part, supplier, lineitem, partsupp, orders,
nation
WHERE
supplier.s_suppkey = lineitem.l_suppkey
AND partsupp.ps_suppkey = lineitem.l_suppkey
AND partsupp.ps_partkey = lineitem.l_partkey
AND part.p_partkey = lineitem.l_partkey
AND orders.o_orderkey = lineitem.l_orderkey
AND supplier.s_nationkey = nation.n_nationkey
AND part.p_name LIKE '%violet%'
AND nation.n_name = 'UNITED KINGDOM'
;
--b10
SELECT COUNT(*)
FROM orders, lineitem
WHERE
orders.o_orderkey = lineitem.l_orderkey
AND (orders.o_orderpriority <> '1-URGENT'
OR orders.o_orderpriority <> '2-HIGH')
AND lineitem.l_shipmode in ('TRUCK', 'SHIP')
AND lineitem.l_commitdate < lineitem.l_receiptdate
AND lineitem.l_receiptdate >= 183.3
AND lineitem.l_receiptdate < 183.3 + 12
;
--b11
SELECT SUM(partsupp.ps_supplycost * partsupp.ps_availqty * 0.2)
FROM partsupp, supplier, nation
WHERE
partsupp.ps_suppkey = supplier.s_suppkey
AND supplier.s_nationkey = nation.n_nationkey
AND nation.n_name = 'JAPAN'
;
--b12_1
SELECT COUNT(*)
FROM orders, lineitem
WHERE
orders.o_orderkey = lineitem.l_orderkey
AND (orders.o_orderpriority = '1-URGENT'
OR orders.o_orderpriority = '2-HIGH')
AND lineitem.l_shipmode in ('TRUCK', 'SHIP')
AND lineitem.l_commitdate < lineitem.l_receiptdate
AND lineitem.l_receiptdate >= 183.3
AND lineitem.l_receiptdate < 183.3 + 12
;
--b12_2
SELECT COUNT(*)
FROM orders, lineitem
WHERE
orders.o_orderkey = lineitem.l_orderkey
AND (orders.o_orderpriority <> '1-URGENT'
OR orders.o_orderpriority <> '2-HIGH')
AND lineitem.l_shipmode in ('TRUCK', 'SHIP')
AND lineitem.l_commitdate < lineitem.l_receiptdate
AND lineitem.l_receiptdate >= 183.3
AND lineitem.l_receiptdate < 183.3 + 12
;
--b16
SELECT COUNT(partsupp.ps_suppkey)
FROM partsupp, part, supplier
WHERE
part.p_partkey = partsupp.ps_partkey
AND partsupp.ps_suppkey = supplier.s_suppkey
AND part.p_brand = 'Brand#14'
AND part.p_type = 'LARGE ANODIZED TIN'
;
B.4 Integer vs Float Type Filtering

Since the date datatype of SQL is measured within day precision, it makes sense to treat it as an integer. However, we could as well represent it as a floating-point number. This allows us to use sigmoids, as discussed in Sec. 5.1.2. For sigmoids, we have to choose precision in such a way that the noise would be smaller. Since precision itself cannot depend on the data, we have empirically evaluated appropriate precision level on an independently generated TCP-H instance with scale factor SF=0.05. As described in Sec. A, the precision has to be increased proportionally with $\sqrt{n}$, where $n$ is the number of analyzed rows. Hence, the sigmoid precisions for the cases of SF 0.1, 0.5, 1.0 had to be multiplied with $\sqrt{2}$, $\sqrt{10}$ and $\sqrt{20}$ respectively.

While Table 3 and Table 4 use integer approximation for date filtering, the tables Table 5 and Table 6 show the results for sigmoid approach. The results have been computed for different $\beta$ and $\alpha$ values, where $\beta \approx 0$ means that the sensitivity could be computed for an arbitrarily small $\beta$, and the third column shows the base $\alpha$ that has been computed for SF=0.05. The computing time for a modified query is much higher for floating points, since the SQL engine now needs to compute exponentiation for each row and each private filter, so for the most complicated queries we present the results up to SF=0.5. We see that, except the queries b12_1 and b12_2, the error gets smaller compared to integer datatype approach. The problem of b12_1 and b12_2 seems to be that the sigmoid precision that we found for SF=0.05 is not the best for SF=0.1, which indeed may happen as the final result depends not only on the number of rows, but also on the actual data, so even though the sensitivity of these queries is smaller, they suffer from precision error. The disadvantages of sigmoid approach are that it takes more time to execute the modified query, and that exponentiations tend to cause overflow errors in PSQL engine when the exponents get large. The time overheads are more significant for the cases with many private filters, where sigmoid error gets larger due to multiplication, so it seems more reasonable to use integer datatype there.

B.5 Examples of Analyzer Output

We give some examples of shorter queries that have been output by the analyzer. We present the modified query and the sensitivity query, demonstrating where computation overhead may come from.

Query b1_1

Modified query.

```
SELECT sum((lineitem.l_quantity * (exp((0.1 * (200.3 + ((-1.0) * lineitem.l_shipdate)))) / (exp((0.1 * (200.3 + ((-1.0) * lineitem.l_shipdate)))) + 1.0))))
FROM lineitem
WHERE (lineitem.l_linenumber = 'F')
AND (lineitem.l_returnflag = 'R');
```

Sensitivity query.
Table 5. Time benchmarks (ms) for the initial query ($t_i$), modified query ($t_m$), and the sensitivity query ($t_s$). $K$ denotes $10^3$, and $M$ denotes $10^6$.

<table>
<thead>
<tr>
<th></th>
<th>$t_i$</th>
<th>$t_m$</th>
<th>$t_s$</th>
<th>$t_i$</th>
<th>$t_m$</th>
<th>$t_s$</th>
<th>$t_i$</th>
<th>$t_m$</th>
<th>$t_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SF = 0.1$</td>
<td>144.36</td>
<td>11.43K</td>
<td>1.74K</td>
<td>761.36</td>
<td>157.38K</td>
<td>9.49K</td>
<td>1.47K</td>
<td>535.97K</td>
<td>18.51K</td>
</tr>
<tr>
<td>$SF = 0.5$</td>
<td>141.57</td>
<td>11.43K</td>
<td>1.74K</td>
<td>742.35</td>
<td>163.81K</td>
<td>9.57K</td>
<td>1.46K</td>
<td>518.45K</td>
<td>19.08K</td>
</tr>
<tr>
<td>$SF = 1.0$</td>
<td>154.66</td>
<td>11.75K</td>
<td>1.89K</td>
<td>886.02</td>
<td>157.73K</td>
<td>9.93K</td>
<td>1.67K</td>
<td>538.28K</td>
<td>21.61K</td>
</tr>
</tbody>
</table>

**SELECT** max(abs(sdsg)) FROM {
  SELECT sum(abs(greatest(abs((exp((0.1 * (200.3 + ((-1.0) * lineitem.l_shipdate)))) + 1.0),
  case when (abs(lineitem.l_quantity) >=
  200.3 + ((-1.0) * lineitem.l_shipdate)) + 1.0) - 1.0) + 1.0))) / (0.1 * lineitem.l_shipdate))))) / ((exp((0.1 * (200.3 + ((-1.0) * lineitem.l_shipdate)))) + 1.0)) * 0.03) = 0.0 then 0.0 else ((0.1 * (200.3 + ((-1.0) * lineitem.l_shipdate)))) / ((exp((0.1 * (200.3 + ((-1.0) * lineitem.l_shipdate)))) + 1.0) - 2.0)) * 0.03) * case when (abs(lineitem.l_quantity) >=
10.0) then abs(lineitem.l_quantity) else (exp((0.1 * abs(lineitem.l_quantity)) - 10.0)) / (0.1) end) (end))) AS sdsg
FROM lineitem, lineitem_sensRows WHERE (lineitem.l_linestatus = 'F') AND (lineitem.l_returnflag = 'R') AND lineitem_sensRows.ID = lineitem.ID AND lineitem_sensRows.sensitive
GROUP BY lineitem_sensRows.ID) AS sub;

**Query b1_5**

**Modified query.**

SELECT sum(abs((0.1 * (200.3 + ((-1.0) * lineitem_l_shipdate))) - 1.0))) / (exp((0.1 * (200.3 + ((-1.0) * lineitem_l_shipdate)))) + 1.0)))
FROM lineitem
WHERE (lineitem_l_linestatus = 'F')
AND (lineitem_l_returnflag = 'R');

**Sensitivity query.**

SELECT max(abs(sdsg)) FROM {
  SELECT sum(abs(greatest(abs((exp((0.1 * (200.3 + ((-1.0) * lineitem_l_shipdate)))) + 1.0),
  case when (abs(lineitem_l_quantity) >=
  200.3 + ((-1.0) * lineitem_l_shipdate)) + 1.0) - 1.0) + 1.0))) / (0.1 * lineitem_l_shipdate))))) / ((exp((0.1 * (200.3 + ((-1.0) * lineitem_l_shipdate)))) + 1.0) - 2.0)) * 0.03) = 0.0 then 0.0 else ((0.1 * (200.3 + ((-1.0) * lineitem_l_shipdate)))) / ((exp((0.1 * (200.3 + ((-1.0) * lineitem_l_shipdate)))) + 1.0) - 2.0)) * 0.03) * case when (abs(lineitem_l_quantity) >=
10.0) then abs(lineitem_l_quantity) else (exp((0.1 * abs(lineitem_l_quantity)) - 10.0)) / (0.1) end) (end))) AS sdsg
FROM lineitem, lineitem_sensRows WHERE (lineitem_l_linestatus = 'F') AND (lineitem_l_returnflag = 'R') AND lineitem_sensRows.ID = lineitem.ID AND lineitem_sensRows.sensitive
GROUP BY lineitem_sensRows.ID) AS sub;

**Query b16**

**Modified query.**

SELECT sum(abs(((2.0 / (exp((0.1 * (part_p_size - 10.0)) + exp((0.1 * (part_p_size - 10.0)))) + (2.0 / (exp((0.1 * (part_p_size - 15.0)) + exp((0.1 * (part_p_size - 15.0)))) + (2.0 / (exp((0.1 * (part_p_size - 20.0)) + exp((0.1 * (part_p_size - 20.0))))))) + (2.0 / (exp((0.1 * (part_p_size - 25.0)) + exp((0.1 * (part_p_size - 25.0))))))) + (2.0 / (exp((0.1 * (part_p_size - 30.0)) + exp((0.1 * (part_p_size - 30.0))))))) + (2.0 / (exp((0.1 * (part_p_size - 35.0)) + exp((0.1 * (part_p_size - 35.0)))))))) + (2.0 / (exp((0.1 * (part_p_size - 40.0)) + exp((0.1 * (part_p_size - 40.0))))))) + (2.0 / (exp((0.1 * (part_p_size - 50.0)) + exp((0.1 * (part_p_size - 50.0))))))))) AS sdsg
FROM lineitem
WHERE (lineitem_l_linestatus = 'F') AND (lineitem_l_returnflag = 'R') AND lineitem_sensRows.ID = lineitem.ID
AND lineitem_sensRows.sensitive
GROUP BY lineitem_sensRows.ID) AS sub;
Table 6. Precision benchmarks for \(\epsilon = 1\), where \(q_i(x)\) is the initial query result, \(q_m(x)\) the modified query result (if different from \(q_i(x)\)), \(q_s(x)\) the sensitivity query result, and \(\%\text{noise} = \frac{|q_m(x) - q_i(x)|}{q_i(x)} \cdot 100\). K denotes \(\cdot 10^3\), M denotes \(\cdot 10^6\), and G denotes \(\cdot 10^9\).

<table>
<thead>
<tr>
<th>(\beta)</th>
<th>sigmoid prec. (\alpha)</th>
<th>(q_i(x))</th>
<th>(q_m(x))</th>
<th>(%\text{noise})</th>
<th>(q_i(x))</th>
<th>(q_m(x))</th>
<th>(%\text{noise})</th>
<th>(q_i(x))</th>
<th>(q_m(x))</th>
<th>(%\text{noise})</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1_1</td>
<td>(\approx 0)</td>
<td>0.1</td>
<td>3.79M</td>
<td>1.8</td>
<td>0.0002</td>
<td>18.87M</td>
<td>4.03</td>
<td>0.0001</td>
<td>37.72M</td>
<td>5.7</td>
</tr>
<tr>
<td>b1_2</td>
<td>(\approx 0)</td>
<td>0.1</td>
<td>5.34G</td>
<td>10.0K</td>
<td>0.0009</td>
<td>27.35G</td>
<td>10.0K</td>
<td>0.0002</td>
<td>56.57G</td>
<td>11.8K</td>
</tr>
<tr>
<td>b1_3</td>
<td>(\approx 0)</td>
<td>0.1</td>
<td>5.07G</td>
<td>19.0K</td>
<td>0.0019</td>
<td>25.98G</td>
<td>19.0K</td>
<td>0.0004</td>
<td>53.74G</td>
<td>21.2K</td>
</tr>
<tr>
<td>b1_4</td>
<td>0.1</td>
<td>0.1</td>
<td>5.27G</td>
<td>12.1K</td>
<td>0.0023</td>
<td>27.02G</td>
<td>12.1K</td>
<td>0.0004</td>
<td>55.89G</td>
<td>13.9K</td>
</tr>
<tr>
<td>b1_5</td>
<td>(\approx 0)</td>
<td>0.05</td>
<td>148.3K</td>
<td>0.02</td>
<td>6e-05</td>
<td>739.56K</td>
<td>0.04</td>
<td>2.7e-05</td>
<td>1.48M</td>
<td>0.06</td>
</tr>
<tr>
<td>b2_1</td>
<td>0.1</td>
<td>0.1</td>
<td>1.07</td>
<td>100.00</td>
<td>93.46K</td>
<td>1.0</td>
<td>100.00</td>
<td>100.00</td>
<td>1.0</td>
<td>100.00</td>
</tr>
<tr>
<td>b2_2</td>
<td>0.1</td>
<td>0.1</td>
<td>999.98</td>
<td>100.00</td>
<td>1.0K</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>1.0K</td>
<td>100.00</td>
</tr>
<tr>
<td>b3</td>
<td>(\approx 0)</td>
<td>0.01</td>
<td>3.62K</td>
<td>19.0K</td>
<td>2.58K</td>
<td>3.21K</td>
<td>19.0K</td>
<td>2.93K</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>b4</td>
<td>(\approx 0)</td>
<td>0.5</td>
<td>2.92K</td>
<td>1.24</td>
<td>1.73</td>
<td>14.17K</td>
<td>3.32</td>
<td>0.76</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>b5</td>
<td>0.1</td>
<td>0.01</td>
<td>5.37M</td>
<td>11.21K</td>
<td>8.1</td>
<td>25.23M</td>
<td>11.21K</td>
<td>1.5</td>
<td>47.6M</td>
<td>11.21K</td>
</tr>
<tr>
<td>b6</td>
<td>(\approx 0)</td>
<td>0.4</td>
<td>17.45M</td>
<td>105.0K</td>
<td>5.67</td>
<td>88.13M</td>
<td>105.0K</td>
<td>1.81</td>
<td>(46.88M)</td>
<td>–</td>
</tr>
<tr>
<td>b7</td>
<td>(\approx 0)</td>
<td>0.1</td>
<td>22.07M</td>
<td>19.0K</td>
<td>0.06</td>
<td>95.63M</td>
<td>19.0K</td>
<td>0.1</td>
<td>212.11M</td>
<td>21.24K</td>
</tr>
<tr>
<td>b8</td>
<td>0.1</td>
<td>0.1</td>
<td>470.8K</td>
<td>112.1K</td>
<td>23.83</td>
<td>2.74M</td>
<td>13.64K</td>
<td>5.31</td>
<td>3.29M</td>
<td>20.01K</td>
</tr>
<tr>
<td>b9</td>
<td>0.1</td>
<td>0.1</td>
<td>30.32M</td>
<td>40.0K</td>
<td>1.32</td>
<td>137.73M</td>
<td>49.2K</td>
<td>0.36</td>
<td>283.82M</td>
<td>49.2K</td>
</tr>
<tr>
<td>b10</td>
<td>0.1</td>
<td>0.1</td>
<td>100.31K</td>
<td>126.5K</td>
<td>125.84</td>
<td>149.6K</td>
<td>31.48K</td>
<td>206.34</td>
<td>0.0</td>
<td>34.94K</td>
</tr>
<tr>
<td>b11</td>
<td>(\approx 0)</td>
<td>0.1</td>
<td>1.63G</td>
<td>199.98K</td>
<td>0.12</td>
<td>7.73G</td>
<td>199.98K</td>
<td>0.03</td>
<td>15.18G</td>
<td>199.98K</td>
</tr>
<tr>
<td>b12_1</td>
<td>(\approx 0)</td>
<td>0.5</td>
<td>3.12K</td>
<td>0.53</td>
<td>8.43</td>
<td>15.4K</td>
<td>1.19</td>
<td>7.32</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>b12_2</td>
<td>(\approx 0)</td>
<td>0.5</td>
<td>1.29K</td>
<td>0.53</td>
<td>8.88</td>
<td>6.2K</td>
<td>1.19</td>
<td>7.61</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>b16</td>
<td>0.1</td>
<td>0.1</td>
<td>9.95K</td>
<td>4.0</td>
<td>0.4</td>
<td>49.35K</td>
<td>4.0</td>
<td>0.08</td>
<td>98.97K</td>
<td>4.0</td>
</tr>
<tr>
<td>b17</td>
<td>(\approx 0)</td>
<td>0.5</td>
<td>31.54K</td>
<td>253K</td>
<td>68.66</td>
<td>256.24K</td>
<td>5.65K</td>
<td>9.83</td>
<td>531.93K</td>
<td>7.99K</td>
</tr>
<tr>
<td>b19</td>
<td>0.1</td>
<td>0.1</td>
<td>155.25K</td>
<td>651.72K</td>
<td>4.2K</td>
<td>1.1M</td>
<td>813.52K</td>
<td>738.04</td>
<td>1.73M</td>
<td>827.69K</td>
</tr>
</tbody>
</table>

FROM part, partsupp, supplier
WHERE NOT((supplier.s_comment LIKE 'Customer%Complaints%'))
AND NOT((part.p_type LIKE '%COPPER%'))
AND NOT((part.p_brand = 'Brand#34'))
AND (part.p_partkey = partsupp.ps_partkey)
AND (partsupp.ps_suppkey = supplier.s_suppkey);

Sensitivity query.

SELECT max(sdsg) FROM (  SELECT sum(abs(...)) AS sub;

FROM part, partsupp, supplier
WHERE NOT((supplier.s_comment LIKE 'Customer%Complaints%'))
AND NOT((part.p_type LIKE '%COPPER%'))
AND NOT((part.p_brand = 'Brand#34'))
AND (part.p_partkey = partsupp.ps_partkey)
AND (partsupp.ps_suppkey = supplier.s_suppkey)
AND part_sensRows.ID = part.ID
AND part_sensRows.sensitive
GROUP BY part_sensRows.ID) AS sub;
C Postponed Proofs

C.1 Useful Facts and Lemmas

Fact 10. For all \((x_1, \ldots, x_m) \in \mathbb{R}^m\), \(p \geq q\), we have
\[
\|x_1, \ldots, x_m\|_p \leq \|x_1, \ldots, x_m\|_q \leq m^{1/q - 1/p} \cdot \|x_1, \ldots, x_m\|_p.
\]

Fact 11. For all \(x \in \mathbb{R}^+\), \(\|x\|_p = x\), and \(\|x\|_\infty = x\).

Fact 12 (a part of Lemma 2.3 of [24]). Let \(f : X \to \mathbb{R}\). For \(\beta > 0\), a \(\beta\)-smooth upper bound on \(f\) is
\[
g_{f, \beta} = \max_{x' \in X} (f(x) \cdot e^{-\beta d(x, x')}).
\]

Lemma 13. Let \(x = (x_1, \ldots, x_k) \in \mathbb{R}^k\), \(y = (y_1, \ldots, y_n) \in \mathbb{R}^n\), \(z = (z_1, \ldots, z_m) \in \mathbb{R}^m\). If \(p \geq q \geq 1\), then
1. \(\|x\|_q, \|y\|_q, z_1, \ldots, z_m \leq \|x\|_q, z_1, \ldots, z_m\|_p,\)
2. \(\|x\|_p, \|y\|_p, z_1, \ldots, z_m \|_q \geq \|x\|_p, z_1, \ldots, z_m\|_q,\)
where \(x|y\) denotes concatenation. If \(p = q\), then the inequalities become equalities.

Proof. Since \(p, q \geq 1\), we may raise both sides of equations to the powers \(p \) or \(q\). The main inequalities that we use in the proof are \(a^n + b^n \leq (a+b)^n\) for \(n \geq 1\), and \(a^n + b^n \geq (a+b)^n\) for \(n \leq 1\).

\[
\begin{align*}
\|x\|_q, \|y\|_q, z_1, \ldots, z_m \|_p^p &= \left( \sum_{i=1}^k x_i^q \right)^{\frac{p}{q}} + \left( \sum_{i=1}^n y_i^q \right)^{\frac{p}{q}} + \sum_{i=1}^m z_i^p \\
&\leq \left( \sum_{i=1}^k x_i^q + \sum_{i=1}^n y_i^q \right)^{\frac{p}{q}} + \sum_{i=1}^m z_i^p \\
&= \|x\|_p^q + \sum_{i=1}^m z_i^q = \|x\|_p, z_1, \ldots, z_m \|_q^q.
\end{align*}
\]

If \(p = q\), then all inequalities in these derivations are equalities.

Lemma 14. For all \(x \in \mathbb{R}\), \((\alpha_1, \ldots, \alpha_k) \in \mathbb{R}^k\), \((y_1, \ldots, y_m) \in \mathbb{R}^m\):
\[
\sum_{i=1}^k \alpha_i^p x_i, y_1, \ldots, y_m\|_p = \left( \sum_{i=1}^k \alpha_i^p x_i, y_1, \ldots, y_m \right)^p.
\]

Proof. Since an \(\ell_p\)-norm is defined for \(p \geq 1\), we may raise both sides of equation to the power \(p\). We use the definition of \(\ell_p\)-norm and rewrite the term.

\[
\begin{align*}
\|\alpha_1 x, \ldots, \alpha_k x, y_1, \ldots, y_m\|_p^p &= \sum_{i=1}^k (\alpha_i x)^p + \sum_{i=1}^m y_i^p \\
&= \left( \sum_{i=1}^k \alpha_i^p x + \sum_{i=1}^m y_i^p \right)^p \\
&= \left( \sum_{i=1}^k \alpha_i^p x, y_1, \ldots, y_m \right)^p.
\end{align*}
\]

Putting \(\alpha_i = 1\) for all \(i \in [n]\), we get the following corollary.

Corollary 15. We have
\[
\left\| \sqrt[k]{x_1, \ldots, x, y_1, \ldots, y_m} \right\|_p = \left( \sum_{i=1}^k \sqrt[k]{x_i} \right)^{\frac{1}{\sqrt[k]{p}}}.
\]

Lemma 16. Let \(X_i\) for \(i \in \{1, \ldots, n\}\) be Banach spaces, \(f_i : X_i \to \mathbb{R}\). Let \(x = (x_1, \ldots, x_n)\), and let \(f(x, \ldots, x_n) = \|f_1(x), \ldots, f_n(x)\|_p\). Then \(\frac{\partial f_i}{\partial x_i}(x) = \frac{\partial f_i}{\partial x_i}(x_i) \cdot \left( \frac{f_i(x)}{f(x)} \right)^{p-1} \leq \frac{\partial f_i}{\partial x_i}(x_i)\).

Proof. Let \(y_i = f_i(x_i)\) and \(y = (y_1, \ldots, y_n)\). We have
\[
\frac{\partial f}{\partial x_i}(x) = \frac{\partial f}{\partial y_i}(y) \cdot \frac{\partial f_i}{\partial x_i}(x_i) = \left( \frac{y_i}{\|y\|_p} \right)^{p-1} \cdot \frac{\partial f_i}{\partial x_i}(x_i).
\]

Since \(\frac{y_i}{\|y\|_p} = \frac{f_i(x)}{f(x)} \leq \left( \frac{f_i(x)}{f(x)} \right)^{p-1} \leq 1\), and hence also \(\frac{f_i(x)}{f(x)} \leq 1\), getting \(\frac{\partial f}{\partial x_i}(x) \leq \frac{\partial f_i}{\partial x_i}(x)\).

Lemma 17. Let \(X_i\) for \(i \in \{1, \ldots, n\}\) be Banach spaces. Let \(x = (x_1, \ldots, x_n)\), and let \(f(x) = \|f_1(x), \ldots, f_n(x)\|_p\). Then \(\frac{\partial f}{\partial x_i}(x) = \sum_{j=1}^n \frac{f_j(x)}{f(x)} \frac{\partial f_j}{\partial x_i}(x)\). This can be upper bounded as:
1. \(\sum_{j=1}^n \frac{f_j(x)}{f(x)} \frac{\partial f_j}{\partial x_i}(x)\)
2. \(\max_{j=1}^n \frac{f_j(x)}{f(x)} \frac{\partial f_j}{\partial x_i}(x)\).
Proof. Let \( y_j = f_j(x) \), \( z = \sum_{j=1}^{n} y_j \). We have

\[
\frac{\partial f}{\partial x_i}(x) = \frac{\partial f}{\partial z}(z) \sum_{j=1}^{n} \left( \frac{\partial f_j}{\partial y_j}(y_j) \cdot \frac{\partial f_j}{\partial x_i}(x) \right) 
\]

\[
= \sum_{j=1}^{n} \left( \frac{y_j}{\|y\|_p} \right)^{p-1} \cdot \frac{\partial f_j}{\partial x_i}(x) 
\]

\[
= \sum_{j=1}^{n} \left( \frac{f_j(x)}{f(x)} \right)^{p-1} \cdot \frac{\partial f_j}{\partial x_i}(x) 
\]

As in the proof of Lemma 16, \( \left( \frac{f_j(x)}{f(x)} \right)^{p-1} \leq 1 \). We get \( \frac{\partial f}{\partial x_i}(x) \leq \sum_{j=1}^{n} \frac{f_j(x)}{f(x)} \). We can proceed with the inequality in another way.

\[
\sum_{j=1}^{n} \left( \frac{f_j(x)}{f(x)} \right)^{p-1} \cdot \frac{\partial f_j}{\partial x_i}(x) = \sum_{j=1}^{n} \frac{f_j(x)^{p-1}}{f(x)^{p-1}} \cdot \frac{\partial f_j}{\partial x_i}(x) 
\]

\[
\leq \max_{j=1}^{n} \left( \frac{1}{f_j(x)} \cdot \frac{\partial f_j}{\partial x_i}(x) \right) \cdot \sum_{j=1}^{n} \frac{f_j(x)^{p-1}}{f(x)^{p-1}} 
\]

\[
= \max_{j=1}^{n} \left( \frac{f(x)}{f_j(x)} \cdot \frac{\partial f_j}{\partial x_i}(x) \right) 
\]

C.2 DP from Cauchy Noise

Proof. Let \( \eta \) be defined as in Sec. 3.1. The construction of differentially private mechanisms based on adding noise distributed according to a generalized Cauchy distribution, with the magnitude depending on the smooth upper bounds on the derivative sensitivity of the original query, was given in Theorem 3. Let \( \eta \sim \text{GenCauchy}(\gamma) \). The generalized Cauchy distribution is relatively stable under shifts and stretchings, satisfying the following inequalities for all \( a_1, a_2, c_1, c_2 \in \mathbb{R} \) [24]:

\[
d_{\text{dp}}(a_1 + c_1 \cdot \eta, a_2 + c_1 \cdot \eta) \leq (\gamma + 1) \cdot \frac{|a_2 - a_1|}{c_1} 
\]

\[
d_{\text{dp}}(c_1 \cdot \eta, c_2 \cdot \eta) \leq (\gamma + 1) \cdot \frac{|c_2 - c_1|}{c_1} 
\]

The combination of these two inequalities gives

\[
d_{\text{dp}}(a_1 + c_1 \cdot \eta, a_2 + c_2 \cdot \eta) \leq (\gamma + 1) \cdot \left( \frac{|a_2 - a_1|}{\max\{|c_1|, |c_2|\}} + \frac{|c_2 - c_1|}{c_1} \right) 
\]

C.3 DP from Laplace Noise

C.3.1 A Metric for \((\epsilon, \delta)\)-Differential Privacy

In Sec. 3.1, we defined a distance \( d_{\text{dp}} \) that is related to \( \epsilon \)-differential privacy. We would like to define a similar distance \( d_{\text{DP}} \) that is related to \((\epsilon, \delta)\)-differential privacy.

Usually, we measure the distances with non-negative real numbers, but it is possible to be more general. In principle, the distances may come from any set equipped with addition and a partial order, where the addition is monotone and has the neutral element that is also the least element in this set. With this signature, we can state all axioms of a metric.

Consider the set \( \mathbb{R}_+ \times \mathbb{R}_+ \) of pairs of non-negative real numbers, where addition and ordering is componentwise. This set can serve as the set of distances, and we could consider it as the range of \( d_{\text{DP}} \); the components somehow corresponding to \( \epsilon \) and \( \delta \). However,
two probability distributions can be $(\epsilon, \delta)$-far from each other for different values of $\epsilon$ and $\delta$; one can be traded for the other.

For a partially ordered set $V$, let $F(V)$ denote the set of all upwards closed subsets of $V$. I.e. $U \subseteq V$ is an element of $F(V)$ if $u \in U$ and $u \leq v$ imply $v \in U$ for all $u, v \in V$. For an arbitrary $U \subseteq V$ we let $\uparrow U$ denote the upwards closure of $U$, i.e. the smallest upwards closed set that contains $U$ as a subset.

If $V$ is a set of distances, then $F(V)$ can also be turned into a set of distances. Let $Z_1, Z_2 \in F(V)$. We have $Z_1 \subseteq Z_2$ iff $Z_2 \subseteq Z_1$. In this way, the entire set $V$ is the only element of $F(V)$ that contains the least element of $V$ is the least element. The addition on $F(V)$ is defined by

$$Z_1 + Z_2 = \uparrow \{ v_1 + v_2 \mid v_1 \in Z_1, v_2 \in Z_2 \}.$$  

(6)

It is easy to see that the operation $+$ is associative, commutative, has the zero element $F(V)$ and is compatible with ordering (meaning that $Z_1 \subseteq Z_2$ implies $Z_1 + Z_3 \subseteq Z_2 + Z_3$ for any $Z_3$).

We let $F(\mathbb{R}_+ \times \mathbb{R}_+)$ be the range of $d_{DP}$. Note that for this set, the $\uparrow$-operation can be left out from (6), due to the continuousness of $\mathbb{R}$. If $X$ is a set, and $\chi, \chi' \in \mathcal{D}(X)$, then define

$$d_{DP}(\chi, \chi') = \bigcap_{X' \subseteq X} \{(\epsilon, \delta) \mid P(X', \chi, \chi', \epsilon, \delta)\},$$  

(7)

where

$$P(X', \chi, \chi', \epsilon, \delta) = \Pr[x \in X' \mid x \leftarrow \chi] \leq e^\epsilon \left( \Pr[x \in X' \mid x \leftarrow \chi'] + \delta \right)$$

$\land \Pr[x \in X' \mid x \leftarrow \chi'] \leq e^\epsilon \left( \Pr[x \in X' \mid x \leftarrow \chi'] + \delta \right).$

Clearly, $d_{DP}(\chi, \chi') \in F(\mathbb{R}_+ \times \mathbb{R}_+)$. Now, a mapping $f : X \rightarrow \mathcal{D}(Y)$ from a metric space $X$ is $(\epsilon, \delta)$-differentially private, if it is $\uparrow \{(\epsilon, \delta)\}$-sensitive for the distance $d_{DP}$ being used on $\mathcal{D}(Y)$.

The following proposition shows that $d_{DP}$ satisfies the triangle inequality.

**Proposition 18.** Let $\chi_1, \chi_2, \chi_3 \in \mathcal{D}(X)$. Then $d_{DP}(\chi_1, \chi_3) \leq d_{DP}(\chi_1, \chi_2) + d_{DP}(\chi_2, \chi_3)$.

**Proof.** Let $(\epsilon_1, \delta_1) \in d_{DP}(\chi_1, \chi_2)$ and $(\epsilon_2, \delta_2) \in d_{DP}(\chi_2, \chi_3)$. According to the definition of $\uparrow$, the pair $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ is a member of $d_{DP}(\chi_1, \chi_2) + d_{DP}(\chi_2, \chi_3)$. We have to show that it is also a member of $d_{DP}(\chi_1, \chi_3)$ according to (7). Let $X' \subseteq X$. Then

\[
\Pr[x \in X' \mid x \leftarrow \chi_1] \\
\leq e^{\epsilon_1} \left( \Pr[x \in X' \mid x \leftarrow \chi_2] + \delta_1 \right) \\
\leq e^{\epsilon_1} \left( e^{\epsilon_2} \left( \Pr[x \in X' \mid x \leftarrow \chi_3] + \delta_2 \right) + \delta_1 \right) \\
= e^{\epsilon_1 + \epsilon_2} \Pr[x \in X' \mid x \leftarrow \chi_3] + e^{\epsilon_1 + \epsilon_2} \delta_2 + e^{\epsilon_1} \delta_1 \\
\leq e^{\epsilon_1 + \epsilon_2} (\Pr[x \in X' \mid x \leftarrow \chi_3] + \delta_2 + \delta_1)
\]

as necessary.

\[\square\]

### C.3.2 Relating the Metric to $(\epsilon, \delta)$-Differential Privacy

Comparing (7) to the definition of $(\epsilon, \delta)$-DP (Def. 1), we see the difference in one important aspect. Namely, in (7), the quantity $\delta$ is multiplied with $e^\epsilon$, while in (1), it is not. While the difference of the factor $e^\epsilon$ seems small in first glance, it is not if we start considering “group privacy”, i.e. distances in $X$ different from 1. Let $d_X(x, x') = L$. If $f : X \rightarrow \mathcal{D}(Y)$ is $\uparrow \{(\epsilon, \delta)\}$-sensitive with respect to the distance $d_{DP}$ on $\mathcal{D}(Y)$, then we know that $(L\epsilon, L\delta) \in d_{DP}(f(x), f(x'))$. But if $f$ is $(\epsilon, \delta)$-differentially private, then we only get

$$\Pr[f(x) \in Y'] \leq e^{L\epsilon} \Pr[f(x') \in Y'] + \frac{e^{L\epsilon} - 1}{e^\epsilon - 1} \delta$$

from Definition 1.

It is not difficult to show that if we do not multiply $\delta$ with $e^\epsilon$, then $d_{DP}$ is no longer a distance; in particular, it would not satisfy the triangle inequality. For example, let us pick

$$\chi_1 = \text{Ber}(0.01) \quad \chi_2 = \text{Ber}(0.03) \quad \chi_3 = \text{Ber}(0.07) \quad \epsilon = 0.01 \quad \delta = 0.01.$$

Here $\text{Ber}(p)$ is the Bernoulli distribution. It returns 1 with probability $p$ and 0 with probability $1-p$. We have $(\epsilon, \delta) \in d_{DP}(\chi_1, \chi_2)$ and also $(\epsilon, \delta) \in d_{DP}(\chi_2, \chi_3)$, but not $(2\epsilon, 2\delta) \in d_{DP}(\chi_1, \chi_3)$. Indeed,

$$\Pr[x = 1 \mid x \leftarrow \chi_2] = 0.03 = 2 - 0.01 + 0.01$$

$$\Pr[x = 1 \mid x \leftarrow \chi_1] = 0.07 = 2 - 0.03 + 0.01$$

$$\Pr[x = 1 \mid x \leftarrow \chi_3] = 0.07 > 4 - 0.01 + 0.02$$

$$\Pr[x = 1 \mid x \leftarrow \chi_3] = e^{2\epsilon} \cdot \Pr[x = 1 \mid x \leftarrow \chi_1] + 2\delta.$$

### C.3.3 Self-similarity of Laplace Distribution

Theorem 4 makes use of the distribution $Lap(1) \in \mathcal{D}(\mathbb{R})$, defined by $Lap(1)(x) \propto e^{-|x|}$. We first have to state the
results about the self-similarity of Lap(1) under shifting and stretching.

**Lemma 19.** Let \( \eta \sim \text{Lap}(1) \). Let \( a_1, a_2 \in \mathbb{R}, c_1, c_2 \in \mathbb{R}_+, c_1 \leq c_2 \). Define \( \beta = \ln(c_2/c_1) \) and let \( \epsilon \geq \beta \). Let \( \delta \geq e^{-\epsilon(\epsilon+\beta)/(\epsilon^2-1)} \). Then the following holds.

\[
\left( \frac{|a_2 - a_1|}{c_1}, 0 \right) \in d_{DP}(a_1 + c_1 \cdot \eta, a_2 + c_1 \cdot \eta) \Rightarrow (\epsilon, \delta) \in d_{DP}(c_1 \cdot \eta, c_2 \cdot \eta).
\]

**Proof.** The probability density functions (pdf) and the cumulative density functions (cdf) of the distributions named above are the following:

\[
\begin{align*}
\text{pdf}_{c_1 \cdot \eta}(x) &= \frac{1}{2c_1} e^{-|x|/c_1} \\
\text{pdf}_{a_1 + c_1 \cdot \eta}(x) &= \frac{1}{2c_1} e^{-|x-a_1|/c_1} \\
\text{pdf}_{c_2 \cdot \eta}(x) &= \frac{1}{2c_2} e^{-|x|/c_2} \\
\text{pdf}_{a_2 + c_1 \cdot \eta}(x) &= \frac{1}{2c_1} e^{-|x-a_2|/c_1}
\end{align*}
\]

and

\[
\begin{align*}
\text{cdf}_{c_1 \cdot \eta}(x) &= \begin{cases} 
  e^{x/c_1}/2, & \text{if } x < 0 \\
  1 - e^{-x/c_1}/2, & \text{if } x \geq 0
\end{cases} \\
\text{cdf}_{c_2 \cdot \eta}(x) &= \begin{cases} 
  e^{x/c_2}/2, & \text{if } x < 0 \\
  1 - e^{-x/c_2}/2, & \text{if } x \geq 0
\end{cases}
\]

The first claim of the lemma is shown by

\[
\max_{x \in \mathbb{R}} \left| \log \frac{\text{pdf}_{a_1 + c_1 \cdot \eta}(x)}{\text{pdf}_{a_2 + c_1 \cdot \eta}(x)} \right| = \max_{x \in \mathbb{R}} \left| \log \frac{e^{-|x-a_1|/c_1}}{e^{-|x-a_2|/c_1}} \right| \leq \ln \left( \frac{|a_2 - a_1|}{c_1} \right) = \frac{|a_2 - a_1|}{c_1},
\]

showing that \( \frac{|a_2 - a_1|}{c_1} \geq d_{DP}(a_1 + c_1 \cdot \eta, a_2 + c_1 \cdot \eta) \). To show the second claim, consider the following function \( f \):

\[
f(x) = \left| \log \frac{\text{pdf}_{c_1 \cdot \eta}(x)}{\text{pdf}_{c_2 \cdot \eta}(x)} \right|.
\]

We are interested in the set of \( x \)-s that satisfy \( f(x) \leq \epsilon \). We have

\[
f(x) = \left| \log \left( \frac{c_2}{c_1} \cdot e^{x/c_2 - |x|/c_1} \right) \right| = \beta - \frac{c_2 - c_1}{c_1c_2} |x|.
\]

The condition \( f(x) \leq \epsilon \) is equivalent to

\[
|x| \leq (\epsilon + \beta) \frac{c_1c_2}{c_2 - c_1}.
\]

To obtain the distance \( \epsilon \{(\epsilon, \delta)\} \), it is sufficient to take \( \delta \) equal to \( e^{-\epsilon} \) times the probability that either \( x \), when sampled according to either \( c_2 \cdot \eta \) or \( c_1 \cdot \eta \), does not satisfy (9). This probability is larger for \( c_2 \cdot \eta \) because \( c_2 \geq c_1 \).

Let us compute this probability.

\[
\Pr[x < 0 \land f(x) > \epsilon | x \leftarrow c_2 \cdot \eta] = \frac{1}{2} e^{(\epsilon+\beta)\frac{c_2}{c_2-c_1}} = \frac{1}{2} e^{(\epsilon+\beta)\frac{c_1}{c_2-c_1}}.
\]

The probability that we are looking for is twice the quantity above. Multiplying it with \( e^{-\epsilon} \) gives us the statement of the lemma.

The next lemma provides a more coarse, but simpler upper bound for the DP-distance between stretched versions of the Laplace distribution.

**Lemma 20.** Let \( \eta \sim \text{Lap}(1) \). Let \( c_1, c_2 \in \mathbb{R}_+, c_1 \leq c_2 \). Define \( \beta = \ln(c_2/c_1) \). Let \( \epsilon \geq \beta \). Let \( k = 1 + \epsilon/\beta \). Then \((\epsilon, e^{-k}) \in d_{DP}(c_1 \cdot \eta, c_2 \cdot \eta)\).

**Proof.** Let \( \delta = e^{-\epsilon - \frac{\epsilon+\beta}{\epsilon^2-1}} \). By the previous lemma, \((\epsilon, \delta) \in d_{DP}(c_1 \cdot \eta, c_2 \cdot \eta)\). We will now show that \( e^{-k} \geq \delta \).

Indeed,

\[
\delta \leq e^{-k} \iff e^{-\epsilon - \frac{\epsilon+\beta}{\epsilon^2-1}} \leq e^{-k} \iff -\epsilon - \frac{\epsilon+\beta}{\epsilon^2-1} \leq -k
\]

\[
\iff (k-1)\beta + \frac{k}{\epsilon^2-1} \geq k
\]

\[
\iff \frac{1}{\epsilon^2-1} \geq \frac{1}{\beta} \geq \frac{k-1}{k}
\]

\[
\iff \frac{1}{\epsilon^2-1} \geq \frac{1}{\beta} \geq \frac{k-1}{k} \iff \frac{1}{\beta} \leq \frac{1}{\epsilon^2-1} + \frac{1}{\epsilon^2+1} \geq 2,
\]

where the \( \sim \) claim holds because \( k \geq 2 \). Consider now the function \( f(x) = x + 4/(e^x+1) \). We have \( f(0) = 2 \).

Also, \( f \) is a monotone function. Hence the claim \( \beta + 4/(e^\beta + 1) \geq 2 \) holds.

### C.3.4 Proof of Theorem 4

Let us restate the theorem about achieving \((\epsilon, \delta)\)-DP using Laplace noise.

**Theorem 4.** Let \( b, \beta, \epsilon \in \mathbb{R}_+, b > 0, b + \beta \leq \epsilon \). Define \( k = 1 + (\epsilon - b) / \beta \). Let \( \delta = e^{-k} \). Let \( \eta \) be a random variable distributed according to \( \text{Lap}(1) \). Let \( c \) be a \( \beta \)-smooth upper bound on \( DS[f] \) for a function \( f : X \to \mathbb{R} \), where \( X \) is Banach space and \( d_X \) is the distance corresponding to the norm of \( X \). Define \( g(x) := f(x) + \frac{c}{\beta} \cdot \eta \). Then

\[
\text{for any } x_1, x_2 \in X, (\epsilon, L, 2\delta) \in d_{DP}(g(x_1), g(x_2)),
\]

where \( L = d_X(x_1, x_2) \).
Lemma 20 required that \( \epsilon \geq \beta \). This condition is translated to

\[
(\epsilon - b) L_1 \geq \ln \frac{c(x_\mu)}{c(x_1)}.
\]

Let us verify that it holds:

\[
\ln \frac{c(x_\mu)}{c(x_1)} \leq \beta \cdot d_X(x_1, x_\mu) \leq \beta L_1 \leq (\epsilon - b) L_1.
\]

We also lower-bound the value of \( k \) from Lemma 20, in order to simplify its expression:

\[
1 + (\epsilon - b) L_1 \geq 1 + \frac{(\epsilon - b) L_1}{\beta L_1} = k.
\]

We obtain

\[
((\epsilon - b) L_1, e^{-k}) \in \text{dDP}(\chi_1, \chi_2).
\]

Similarly, we can obtain

\[
((\epsilon - b) L_2, e^{-k}) \in \text{dDP}(\chi_3, \chi_4).
\]

Using the triangle inequality, we can combine \( \text{dDP}(\chi_i, \chi_{i+1}) \) for \( i \in \{1, 2, 3\} \):

\[
(\epsilon L, e^{-k}) \in \text{dDP}(\chi_1, \chi_4)
\]

as required. If \( d_X(x_1, x_2) = 1 \), then \( (\epsilon, 2e^{-k}) \in \text{dDP}(\chi_1, \chi_2) \), i.e. \( g \) is \( (\epsilon, 2\delta) \)-differentially private.

If \( c \) is monotone along the path \( h \), then the point \( x_\mu \) coincides with either \( x_1 \) or \( x_2 \). W.l.o.g. assume \( x_\mu = x_1 \).

Then \( \chi_1 = \chi_2 \) and \((0,0) \in \text{dDP}(\chi_1, \chi_2)\). The triangle inequality now gives \((\epsilon L, e^{-k}) \in \text{dDP}(\chi_1, \chi_4)\).

\( \square \)

### C.4 Domination between Norms

#### C.4.1 Proof of Lemma 1

Let \( N = N'(V_1, \ldots, V_m) \). The relation \( V_i \preceq W_i \) implies \( \|x_1, \ldots, x_n\|_{V_i} \leq \|x_1, \ldots, x_n\|_{W_i} \) for all \( x_1, \ldots, x_n \in \mathbb{R}^n \).

Define a new norm \( M = N'(W_1, \ldots, W_m) \). By definition of a composite norm, we have the three cases for \( N' \):

- If \( N' = |x_j| \) for some \( j \in [n] \), then \( m = 0 \), and hence \( \|x_1, \ldots, x_n\|_N = \|x_1, \ldots, x_n\|_{V_1} \), and \( \|x_1, \ldots, x_n\|_M = \alpha \|x_1, \ldots, x_n\|_{V_1} \), so \( N \preceq M \).

- If \( N' = \alpha z \), then \( m = 1 \), and we have \( \|x_1, \ldots, x_n\|_N = \alpha \|x_1, \ldots, x_n\|_{V_1} \) and \( \|x_1, \ldots, x_n\|_M = \alpha \|x_1, \ldots, x_n\|_{W_1} \), so \( N \preceq M \).

- If \( N' = \preceq \), then \( \|x_1, \ldots, x_n\|_N \preceq \|x_1, \ldots, x_n\|_{V_1} \), \( \|x_1, \ldots, x_n\|_{V_1} \preceq \|x_1, \ldots, x_n\|_{V_2} \), \( \|x_1, \ldots, x_n\|_{V_2} \preceq \|x_1, \ldots, x_n\|_{V_3} \), \( \|x_1, \ldots, x_n\|_p \preceq \|x_1, \ldots, x_n\|_{W_1} \), \( \|x_1, \ldots, x_n\|_{W_1} \preceq \|x_1, \ldots, x_n\|_{W_2} \), \( \|x_1, \ldots, x_n\|_p \preceq \|x_1, \ldots, x_n\|_{W_2} \), \( \|x_1, \ldots, x_n\|_p \preceq \|x_1, \ldots, x_n\|_{W_3} \), so \( N \preceq M \).

In any case, we get \( N \preceq M \), which is equivalent to \( N'(V_1, \ldots, V_m) \preceq N'(W_1, \ldots, W_m) \).
C.4.2 Proof of Lemma 2

Without loss of generality, we assume that all scalings in $N$ are applied directly to the variables, as we can always apply the equality $\alpha |x| = |\alpha x|$ to push all scalings as deep as possible, directly in front of variables. Let the variable $x_i$ occur $k_i$ times in $N$, and let $\alpha_{ij}$ be the scaling of the $j$-th occurrence of $x_i$. We define $\alpha_i$ and $\beta_i$ of Lemma 2 as follows.

- $\alpha_i = \sqrt[k_i]{\sum_{j=1}^{k_i} \alpha_{ij}}$.
- $\beta_i = \sqrt[k_i]{\sum_{j=1}^{k_i} \alpha_{ij}}$.

We prove the first inequality, and the proof would be analogous for the second one. Let $N = \|M_1, \ldots, M_k\|_r$.

Since $p$ is the largest $\ell_p$-norm used as a term constructor of $N$, we have $\|M_1, \ldots, M_k\|_r \geq \|M_1, \ldots, M_k\|_p$. Repeat the same procedure with all $M_1, \ldots, M_k$ recursively, substituting all instances of $\ell_r$ with $\ell_p$. By Lemma 1, each step of the transformation keeps the resulting norm smaller (or equal). Finally, we are left with a composite norm $N'$ that only contains $\|\cdot\|_p$ for the same $p \geq 1$ as a term constructor. We can now apply Lemma 13 and get a norm of the form $N' = \|\alpha_{11}x_1, \ldots, \alpha_{nk_n}x_n\|_p$, such that $N' \leq N$.

Some variables $x_i$ used by $N'$ may repeat if they were repeating in $N$ before. We may now use Lemma 14 to merge repeating variables into one, rewriting

$$\left\|\alpha_{11}x_1, \ldots, \alpha_{1k_1}x_1, \ldots, \alpha_{nk_n}x_n\right\|_p = \left\|\alpha_{11}x_1, \ldots, \sum_{j=1}^{k_i} \alpha_{ij}x_{ij}, \ldots, \alpha_{nk_n}x_n\right\|_p$$

After doing it for all $i \in [n]$, we get a norm $N'' = \|\alpha_1x_1, \ldots, \alpha_nx_n\|_p$, which satisfies $N'' \leq N$.

C.5 Basic Results of Derivative Sensitivity

C.5.1 Proof of Lemma 3

Let $\nabla f(x) = (a_i)_{i=1}^n$. Assuming $a_i \neq 0$ for all $i$ (otherwise remove the indices $i$ for which $a_i = 0$ from the summations containing $a_i$):

$$|df_x(y)| = |\nabla f(x) \cdot y| \leq \sum_{i=1}^{n} |a_i||y_i|$$

$$\leq \left(\sum_{i=1}^{n} |a_i|^p\right)^{1/p} \left(\sum_{i=1}^{n} |y_i|^p\right)^{1/p}$$

$$= \left|\nabla f(x)\right|_q \cdot \|y\|_p$$

for all $y \in X$. The second inequality used here is the weighted power means inequality with exponents 1 and $p$. Equality is achievable (and not only for $y = 0$): for example, by taking $y_i = |a_i|^{-1/p}$. Thus $\|\nabla f(x)\|_q$ is the smallest value of $c$ such that for all $y$, $|df_x(y)| \leq c \cdot \|y\|_p$, i.e. it is the operator norm $\|df_x\|$.

The cases $p = 1$ and $p = \infty$ can be achieved as limits of the general case.

C.5.2 Proof of Lemma 4

The derivative sensitivity of $f$ at $x$ is the operator norm of a particular linear operator $df_x$. It is equal to the minimal possible $c$, such that for all vectors $y$, the absolute value of of $df_x(y) \in \mathbb{R}$ is at most $c$ times larger than the norm $\|y\|_N$. If we replace $N$ with $a \cdot N$, then the norm $\|y\|_{a \cdot N}$ is increased by $a$ times. Hence we may now reduce $c$ by $a$ times and still have the inequality.

C.5.3 Proof of Lemma 5

(a) We first prove that $(V, \|\cdot\|_V)$ is a normed vector space. We prove only the triangle inequality. The rest of the properties of norm are easy to check.

$$\|(v_1, v_2) + (v'_1, v'_2)\|_V = \|(v_1 + v'_1, v_2 + v'_2)\|_V \leq$$

$$\|v_1 + v'_1\|_V + \|v_2 + v'_2\|_V$$

The first inequality uses the triangle inequalities of $\|\cdot\|_V$ and $\|\cdot\|_{V_2}$ and the monotonicity of $\|\cdot\|_p$ in the absolute values of the coordinates of its argument vector. The second inequality uses the triangle inequality of $\|\cdot\|_p$.

Thus $(V, \|\cdot\|_V)$ is a normed vector space. It remains to prove that it is complete. Consider a Cauchy sequence $\{x_n\}$ in $V$. Then

$$\forall \epsilon > 0. \exists N \in \mathbb{N}. \forall m,n > N. \|x_m - x_n\|_V < \epsilon$$
Let \( x_n = (y_n, z_n) \) where \( y_n \in V_1 \) and \( z_n \in V_2 \). Note that
\[
\|y_m - y_n\|_V = \|(y_m - y_n, 0)\|_V \\
\leq \|(y_m - y_n, z_m - z_n)\|_V = \|x_m - x_n\|_V
\]
Thus
\[
\forall \epsilon > 0, \exists N \in \mathbb{N}, \forall m, n > N, \|y_m - y_n\|_V < \epsilon
\]
i.e. \( \{y_n\} \) is a Cauchy sequence in \( V_1 \). Because \( V_1 \) is a Banach space, there exists \( y \in V_1 \) such that
\[
\lim_{n \to \infty} y_n - y = 0 \|V_1\|
\]
Similarly, we get that there exists \( z \in V_2 \) such that
\[
\lim_{n \to \infty} z_n - z = 0 \|V_2\|
\]
Let \( x = (y, z) \). Note that
\[
\|x_n - x\|_V = \|(y_n - y, z_n - z)\|_V \\
= \|(y_n - y, y_1 - y_1, z_n - z, z_2 - z_2)\|_p
\]
Then, because \( \|\cdot\|_p \) is continuous,
\[
\lim_{n \to \infty} \|x_n - x\|_V = 0 \|V\|
\]
Thus \( V \) is a Banach space.

(b) Let \( c_1 = \|dg_{v_1}\| \), \( c_2 = \|dh_{v_2}\| \). Note that
\[
\lim_{x_1 \to 0} \|g(v_1 + x_1) - g(v_1) - df_v(x_1, 0)\|_V = 0
\]
\[
\lim_{x_1 \to 0} \|f(v_1 + x_1, v_2) - f(v_1, v_2) - df_v(x_1, 0)\|_V = 0
\]
\[
\lim_{x_1 \to 0} \|f(v + (x_1, 0)) - f(v) - df_v(x_1, 0)\|_V = 0
\]
The last equality holds by the definition of Fréchet derivative. The equality before that holds because the limit on the right-hand side exists. Then, again by the definition of Fréchet derivative, we get that the linear map that maps \( x_1 \) to \( df_v(x_1, 0) \), is \( dg_{v_1} \). Thus \( df_v(x_1, 0) = dg_{v_1}(x_1) \). Similarly, we get \( df_v(0, x_2) = dh_{v_2}(x_2) \).

The last inequality follows from the weighted power means inequality, similarly to the proof of Lemma 3. Equality is also achievable: because \( c_1 = \|dg_{v_1}\| \) and \( c_2 = \|dh_{v_2}\| \), there exist \( x_1 \) and \( x_2 \) that achieve equality in the second inequality. Then scale \( x_1 \) and \( x_2 \) by constants such that \( \|x_1\|_V \) and \( \|x_2\|_V \) (which scale by the same constants) achieve equality in the third in equality. To achieve equality in the first inequality, we may further need to multiply \( x_1 \) and/or \( x_2 \) by \(-1\). Thus \( \|df_v\| = \|(c_1, c_2)\|_q \).

### C.6 Alternative Definition of Smoothness

In this Section, we prove Lemma 6. By Def. 13, the mapping \( f \) is \( \beta \)-smooth, if \( f(x) \leq e^{\beta \|x' - x\|} \cdot f(x') \) for all \( x, x' \in X \). We may rewrite it as \( \frac{f(x) - f(x')}{{\|x' - x\|}} \leq e^{\beta \|x' - x\|} \cdot f(x') \). Applying \( \ln \) to both sides, it suffices to prove that \( \ln(f(x)) - \ln(f(x')) \leq \beta \cdot \|x' - x\| \), which is
\[
\frac{\ln(f(x)) - \ln(f(x'))}{\|x' - x\|} \leq \beta. \]

Applying mean value theorem to the function \( \ln \circ f : X \to \mathbb{R} \), we get \( \|\ln(f(x)) - \ln(f(x'))\| = \|d(\ln \circ f)\|_v \) for some \( v \in X \). Applying derivative chain rule, since \( \frac{d(\ln \circ f)}{dx} (x) = \frac{1}{f(x)} \), we get \( \|d(\ln \circ f)\| = \|\frac{df}{dx}\| = \|\frac{df}{f'(f(x))} \|_v \leq \beta \), where the last inequality comes from the lemma statement.

### C.7 Smoothness of Composite Functions

#### C.7.1 Proof of Lemma 7

We have:
1. \[
\frac{(f(x) + g(x))}{f(x)} \leq \frac{f(x) + g'(x)}{f(x) + g(x)} \leq \max \left( \frac{f'(x)}{f(x)}, \frac{g'(x)}{g(x)} \right) \leq \max(\beta_f, \beta_g).
\]
2. \[
\frac{f(x)g(x)}{f(x)g(x)} \leq \frac{f(x)g(x) + f(x)g(x)}{f(x)g(x)} \leq \beta_f + \beta_g.
\]
3. \[
\frac{(f(x)g(x))}{f(x)g(x)} \leq \frac{f(x)g(x) - f(x)g(x)}{f(x)g(x)} \leq \beta_f + \beta_g.
\]

#### C.7.2 Proof of Lemma 8

Let \( X = \prod_{i=1}^n X_i \) and \( x = (x_1, \ldots, x_n) \). Let \( \beta = \max \beta_i \).
By Lemma 16, an upper bound on \( \frac{df}{dx}(x) \) is \( c_i(x) = \ldots \)
We have
\[ |c_i(x)| = |f'_i(x)| \leq DS[f_i](x) = |f_i(x)| \cdot \frac{DS[f_i](x)}{|f_i(x)|} \leq |f_i(x)| \cdot \beta_i. \]

By Lemma 3 and Lemma 5, the derivative sensitivity of \( f \) in \( (X, \ell_p) \) is
\[ DS[f](x) = \|(c_1(x), \ldots, c_n(x))\|_p \]
Using inequality \( |f_i(x)| \leq |f(x)| \), we get
\[ DS[f](x) \leq \frac{\|(|f_i(x)| \cdot \beta_i)_i \|_p}{|f(x)|} \leq \frac{|f(x)| \cdot \|(|\beta_i)_i \|_p}{|f(x)|} \leq \|(|\beta_i)_i \|_p. \]

On the other hand, using inequality \( \beta_i \leq \beta \), we get
\[ DS[f](x) \leq \frac{\|(|f_i(x)| \cdot \beta_i^n)_{i=1}^n \|_p}{|f(x)|} \leq \beta \cdot \|(|f_i(x)| \cdot \beta_i^n)_{i=1}^n \|_p = \beta \]

**C.7.3 Proof of Lemma 9**

Let \( X = \prod_{i=1}^n X_i \) and \( x = (x_1, \ldots, x_n) \). By Lemma 17, an upper bound on \( \frac{\partial f}{\partial x_i}(x) = c_i(x) = \max_j f_j(x) / f_j(x) \cdot \frac{\partial f_i}{\partial x_i}(x) \).
We have
\[ |c_i(x)| = \max_j \left| f_j(x) \cdot \frac{\partial f_i}{\partial x_i}(x) \right| \leq |f(x)| \cdot \max_j \left| \frac{\partial f_i}{\partial x_i}(x) \cdot \frac{1}{f_j(x)} \right| \leq |f(x)| \cdot \max_j \beta_i^j \cdot \beta_i. \]

By Lemma 3 and Lemma 5, the derivative sensitivity of \( f \) in \( (X, \ell_{\text{dual}(y)}) \) is
\[ DS[f](x) = \|(c_1(x), \ldots, c_n(x))\|_p \]
We get
\[ DS[f](x) \leq \frac{|f(x)| \cdot \left\| \left( \max_j \beta_i^j \right)_{i=1}^n \right\|_p}{|f(x)|} \leq \left\| \left( \max_j \beta_i^j \right)_{i=1}^n \right\|_p. \]

**C.8 Derivative Sensitivity of Simple Functions**

The following functions were considered in Table 2.

**Power function.** Let \( f(x) = x^r, r \in \mathbb{R}_+, x > 0 \). We have
\[ \frac{f'(x)}{f(x)} = \frac{x^{r-1}}{x^r} = \frac{r}{x}; \quad \left| \frac{r}{x} \right| \leq \beta \iff x \geq \frac{|r|}{\beta}. \]
For \( x \leq \frac{|r|}{\beta} \), the function \( f'(x) \) achieves its maximum at the point \( \frac{r}{\beta} \). By Lemma 12, a \( \beta \)-smooth upper bound on \( f \) is
\[ \text{UB}_{f}(x) = \begin{cases} \frac{x^r}{e^{\beta x-r}} \left( \frac{r}{\beta} \right)^{r-1} \quad \text{if } x \geq \frac{r-1}{\beta} \\ \frac{r e^{\beta x-(r-1) \left( \frac{r-1}{\beta} \right)^{r-1}}} {e^{\beta x-r}} \quad \text{otherwise} \end{cases} \]
A \( \beta \)-smooth upper bound on \( DS[f] \) is
\[ \text{UB}_{DS}[f] = \begin{cases} \frac{rx^{r-1}}{e^{\beta x-r}} \left( \frac{r}{\beta} \right)^{r-1} \quad \text{if } x \geq \frac{r-1}{\beta} \\ \frac{r e^{\beta x-(r-1) \left( \frac{r-1}{\beta} \right)^{r-1}}} {e^{\beta x-r}} \quad \text{otherwise} \end{cases} \]

**Exponent.** Let \( f(x) = e^{rx}, r \in \mathbb{R}, x \in \mathbb{R} \). We have
\[ DS[f](x) = |f'(x)| = |r|e^{rx}, \quad \text{hence:} \quad \left| \frac{f'(x)}{f(x)} \right| = \frac{e^{rx}}{e^{rx}} = r; \quad \left| \frac{f'(x)}{f(x)} \right| = \frac{e^{rx}}{e^{rx}} = r. \]
Thus both \( f \) and \( DS[f] \) are \( \beta \)-smooth if \( |r| \leq \beta. \)

**Sigmoid.** Consider the (sigmoid) function \( \sigma(x) = \frac{e^{\alpha x}}{1 + e^{\alpha x}} \). This function can be viewed as a continuous approximation of the indicator function \( 1_{(0,1)} : \mathbb{R} \rightarrow \{0,1\} \), which is less precise for values close to 0, and the error decreases when \( \alpha \) increases. We have:
- \( \sigma'(x) = \frac{\alpha e^{\alpha x} e^{\alpha x}}{(1 + e^{\alpha x})^2}; \quad \sigma''(x) = \frac{\alpha^2 e^{\alpha x} e^{\alpha x} (e^{\alpha x} - 1)}{(1 + e^{\alpha x})^3}; \)
- \( \left| \frac{\sigma'(x)}{\sigma(x)} \right| \leq \left| \alpha \cdot \frac{1}{e^{\alpha x} - 1} \right| \leq \alpha; \quad \left| \frac{\sigma''(x)}{\sigma(x)} \right| \leq \left| \alpha \cdot \frac{e^{\alpha x} - 1}{e^{\alpha x} - 1} \right| \leq \alpha. \]
Thus both \( \sigma(x) \) and \( DS[\sigma(x)] \) are \( \alpha \)-smooth. If we want less DP noise, we should decrease \( \alpha \), which in turn makes the sigmoid itself less precise, so there is a tradeoff.

**Tauoid.** Consider the function \( \tau(x) = \frac{2}{e^{\alpha x} + e^{-\alpha x}} \) (let us call it a tauoid). This function can be viewed as a continuous approximation of the indicator function \( I_{(0,1)} : \mathbb{R} \rightarrow \{0,1\} \), which works similarly to a sigmoid.
We have:
\[
\tau'(x) = -\frac{2\alpha(e^{\alpha x} - e^{-\alpha x})}{(e^{\alpha x} + e^{-\alpha x})^2} = \frac{2\alpha(e^{\alpha x} - e^{-\alpha x})}{e^{\alpha x} + e^{-\alpha x}} = \frac{2\alpha(e^{\alpha x}(1 - e^{2\alpha x}))}{(1 + e^{2\alpha x})^2}
\]
\[
|\tau'(x)| \leq |\alpha| - |e^{-\alpha x} - e^{\alpha x}| \leq |\alpha|
\]
\[
|\tau'(x)| \leq \frac{2|\alpha|e^{\alpha x}}{1 + e^{2\alpha x}} = \frac{|\alpha|\tau(x) = \text{UB}_{\text{DS}, \alpha}(x)}
\]

Thus both \(\tau\) itself and \(\text{UB}_{\text{DS}, \alpha}\) is a bound upper on its derivative sensitivity, is \(\alpha\)-smooth.

An \(\ell_p\)-norm. Consider the function \(f(x) = \|x\|_p = (\sum x_i^p)^{1/p}\), \(x \in \mathbb{R}^n, x = (x_1, \ldots, x_n)\). We have
\[
\frac{\partial f}{\partial x_i} = \frac{p x_i^{p-1}}{p (\sum x_i^p)^{(p-1)/p}} = \left(\frac{x_i^{p-1}}{\sum x_i^p} \right)^{(p-1)/p}
\]

By Lemma 3, the derivative sensitivity of \(f\) in \((\mathbb{R}^n, \ell_p)\) is
\[
\text{DS}[f](x) = \left(\sum x_i^{p-1}/\sum x_i^p\right)^{p-1} = 1 \quad \text{for}\quad \|x\|_p = 1
\]

This is constant and thus \(\beta\)-smooth for all \(\beta\). The function \(f\) itself is \(\beta\)-smooth if \(\frac{1}{\|x\|_p} \leq \beta\), i.e. if \(\|x\|_p \geq \frac{1}{\beta}\).

By Lemma 12, a \(\beta\)-smooth upper bound on \(f\) is
\[
\text{UB}_f(x) = \begin{cases} 1 & \text{if } \|x\|_p \geq \frac{1}{\beta} \wedge \text{argmax}_j |x_j| \leq \beta \\ \text{undefined} & \text{otherwise} \end{cases}
\]

This also holds for \(p = \infty\).

The \(\ell_\infty\)-norm. Let \(f(x) = \|x\|_\infty = \max_i |x_i|\). We have
\[
\frac{\partial f}{\partial x_i} = \begin{cases} 1 & \text{if } i = \text{argmax}_j |x_j| \\ \text{undefined} & \text{if } \text{argmax}_j |x_j| \neq \text{unique} \wedge 0 \\ 0 & \text{otherwise} \end{cases}
\]

The derivative sensitivity of \(f\) in \((\mathbb{R}^n, \ell_\infty)\) is
\[
\text{DS}[f](x) = \begin{cases} 1 & \text{if } \text{argmax}_j |x_j| \text{ is unique} \\ \text{undefined} & \text{if } \text{argmax}_j |x_j| \text{ is not unique} \end{cases}
\]

Because we are interested in upper bounds on the derivative sensitivity, we define
\[
\text{DS}[f]\left(x_0\right) := \limsup_{x \to x_0} \text{DS}[f](x) = 1
\]

for those \(x_0\) for which \(\text{DS}[f](x_0)\) is undefined. Thus \(\text{DS}[f](x) = 1\), which is constant and \(\beta\)-smooth for all \(\beta\). The smooth upper bound on the function \(f\) itself can be found similarly to the \(\ell_p\)-norm case.

C.9 Composing Derivative Sensitivity

The following constructions are considered in Fig. 1. **Product.** Let \(f : \prod_{i=1}^n X_i \to \mathbb{R}, f(x_1, \ldots, x_n) = \prod_{i=1}^n f_i(x_i)\) where \(X_i\) are Banach spaces. Let \(X = \prod_{i=1}^n X_i\) and \(x = (x_1, \ldots, x_n)\). First, suppose that variables \(x_i\) are independent. We have \(\frac{\partial f}{\partial x_i}(x) = \prod_{j \neq i} f_j(x_j) \cdot f'_i(x_i),\) and \(\frac{\partial f}{\partial x_i}(x) \cdot \frac{1}{f_i(x_i)} = f'_i(x_i)/f_i(x_i),\)

We have \(f_i(x_i)\) constant and \(\beta\)-smooth.

Thus both \(\tau\) itself and \(\text{UB}_{\text{DS}, \alpha}\), an upper bound on its derivative sensitivity, are \(\alpha\)-smooth.

**The \(\ell_\infty\)-norm.** Let \(f(x) = \|x\|_\infty = \max_i |x_i|\). We have
\[
\frac{\partial f}{\partial x_i} = \begin{cases} 1 & \text{if } i = \text{argmax}_j |x_j| \\ \text{undefined} & \text{if } \text{argmax}_j |x_j| \text{ is unique} \wedge 0 \\ 0 & \text{otherwise} \end{cases}
\]

The derivative sensitivity of \(f\) in \((\mathbb{R}^n, \ell_\infty)\) is
\[
\text{DS}[f](x) = \begin{cases} 1 & \text{if } \text{argmax}_j |x_j| \text{ is unique} \\ \text{undefined} & \text{if } \text{argmax}_j |x_j| \text{ is not unique} \end{cases}
\]

Because we are interested in upper bounds on the derivative sensitivity, we define
\[
\text{DS}[f]\left(x_0\right) := \limsup_{x \to x_0} \text{DS}[f](x) = 1
\]

for those \(x_0\) for which \(\text{DS}[f](x_0)\) is undefined. Thus \(\text{DS}[f](x) = 1\), which is constant and \(\beta\)-smooth for all \(\beta\). The smooth upper bound on the function \(f\) itself can be found similarly to the \(\ell_p\)-norm case.
Consider the case where \( f(x) = \sum_{i=1}^{n} g_i(x) \) where \( g_i : X \to \mathbb{R} \) and \( X \) is a Banach space. Then

\[
DS[f](x) = \sum_{i=1}^{n} DS[g_i](x)
\]

By Lemma 7, if all \( DS[g_i] \) are \( \beta \)-smooth then \( DS[f] \) is \( \beta \)-smooth. If all \( g_i \) are non-negative and \( \beta \)-smooth then \( f \) is \( \beta \)-smooth. This shows the correctness of the rules \((+p)\) and \((+s)\).

**Min / max.** Let \( f : \prod_{i=1}^{n} X_i \to \mathbb{R}, f(x_1, \ldots, x_n) = \min_{i=1}^{n} f_i(x_i) \) where \( X_i \) are Banach spaces (the case with max instead of min is similar). Let \( X = \prod_{i=1}^{n} X_i \) and \( x = (x_1, \ldots, x_n) \). Let the variables \( x_i \) be independent.

If for all \( i, f_i \) is \( \beta \)-smooth in \( X_i \), then \( f \) is \( \beta \)-smooth in \( (X, \ell_p) \). The same holds with max instead of min.

The derivative sensitivity of \( f \) w.r.t. \( x_i \) is \( DS[f_i](x_i) \) if \( i = \arg\min f_i(x_i) \) and 0 otherwise. The derivative sensitivity of \( f \) in \( (X, \ell_p) \) is \( DS[f](x) = DS[f_i](x_i) \) where \( i = \arg\min f_i(x_i) \). In general, \( DS[f] \) is discontinuous at points where \( \arg\min f_i(x_i) \) is not unique.

A possible valid \( \beta \)-smooth (in \( (X, \ell_p) \)) upper bound on \( DS[f] \) is \( \max c_i(x_i) \) where \( c_i \) is a \( \beta \)-smooth upper bound on \( DS[f_i] \). This shows the correctness of the rules \((\min_D^p)\) and \((\max_D^p)\).

**Norm scaling.** Let \( f : X \to \mathbb{R} \) in the Banach space \((X, \| \cdot \|)\). Scaling the norm by \( a \) scales the derivative \( f'(x) \) by \( \frac{1}{a} \) while keeping the value of \( f(x) \) the same.

Hence, if \( f \) is \( \beta \)-smooth in \((X, \| \cdot \|)\) then it is \( \frac{\beta}{a} \)-smooth in \((X, a \cdot \| \cdot \|)\). Hence the rule \((\times a)\) is correct.

Let \( c(x) \) be a \( \beta \)-smooth upper bound on the derivative sensitivity of \( f \) at \( x \) in \((X, \| \cdot \|)\). Then \( \frac{c(x)}{a} \) is a \( \frac{\beta}{a} \)-smooth upper bound on the derivative sensitivity of \( f \) at \( x \) in \((X, a \cdot \| \cdot \|)\) by Lemma 4. Hence the rule \((\div a)\) is correct.

**Sensitivity w.r.t. a larger norm.** Let \( f : X \to \mathbb{R} \) in the Banach space \((X, \| \cdot \|_N)\). Let \( \| \cdot \|_M \geq \| \cdot \|_N \).

If \( f \) is \( \beta \)-smooth in \((X, \| \cdot \|_N)\), then \( f(x) \leq e^\beta \|x-x'\|_N \cdot f(x') \leq e^\beta \|x-x'\|_M \cdot f(x') \) for all \( x, x' \in X \), so \( f \) is also \( \beta \)-smooth in \((X, \| \cdot \|_M)\). The same holds about any function that is \( \beta \)-smooth in \((X, \| \cdot \|_N)\), including a \( \beta \)-smooth upper bound on the derivative sensitivity of \( f \).

Let us show that the derivative sensitivity of \( f \) w.r.t. \( \| \cdot \|_N \) is a valid upper bound on the derivative sensitivity of \( f \) w.r.t. \( \| \cdot \|_M \). First, note that \( \| \cdot \|_{\text{dual}(N)} \geq \| \cdot \|_{\text{dual}(M)} \).

Indeed, by definition of a dual norm, \( \|T\|_{\text{dual}(M)} = \sup \{ |T(x)| \mid \|x\|_M \leq 1 \} \) for an operator \( T \) from the dual space \( X \to \mathbb{R} \). Since \( \|x\|_N \leq \|x\|_M \), we have \( \forall x : \{ |T(x)| \mid \|x\|_N \leq 1 \} \geq \{ |T(x)| \mid \|x\|_M \leq 1 \} \).

Hence, \( \|T\|_{\text{dual}(N)} \geq \sup \{ |T(x)| \mid \|x\|_N \leq 1 \} \geq \|T\|_{\text{dual}(M)} \) for all \( x \).

By definition, we have \( DS[f](x) = \|df_x\|_{\text{dual}(N)} \), where \( df_x \) is the Fréchet derivative of \( f \) at \( x \). Since \( \|df_x\|_{\text{dual}(N)} \geq \|df_x\|_{\text{dual}(M)} \), we have \( \|df_x\|_{\text{dual}(N)} \geq \|ds_x\|_{\text{dual}(M)} \) for all \( x \). Hence the rule \((\div D)\) is correct.

**Composition with a real function.** Let \( f(x) = h(g(x)), x \in X \) where \( g : X \to \mathbb{R}, h : \mathbb{R} \to \mathbb{R} \) and \( X \) is a Banach space.

\[
DS[f](x) = |h'(g(x))| \cdot DS[g](x)
\]

\[
DS[f](x) = \frac{|h'(g(x))|}{|f'(x)|} \cdot DS[g](x)
\]

Suppose that \( h \) is \( \beta_h \)-smooth and \( DS[g](x) \leq B \) for all \( x \). Then \( f \) is \( \beta_h B \)-smooth. We have

\[
DS[DS[f]](x) = |h''(g(x))|(DS[g](x))^2 + |h'(g(x))| \cdot DS[DS[g]](x),
\]

\[
\frac{DS[DS[f]](x)}{DS[f](x)} = \frac{|h''(g(x))|}{|h'(g(x))|} \cdot DS[g](x) + \frac{DS[DS[g]](x)}{DS[g](x)}.
\]

By Lemma 7, if \( h' \) is \( \beta_h \)-smooth, \( DS[g] \) is \( \beta_g \)-smooth, and \( DS[g](x) \leq B \) for all \( x \) then \( DS[f] \) is \( \beta_h B + \beta_g \)-smooth. Hence the rules \((\diamond a)\) and \((\diamond D)\) are correct.