<table>
<thead>
<tr>
<th>Title</th>
<th>Exploiting metric structure for efficient private query release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Huang, Z; Roth, A</td>
</tr>
<tr>
<td>Issued Date</td>
<td>2014</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10722/201110">http://hdl.handle.net/10722/201110</a></td>
</tr>
<tr>
<td>Rights</td>
<td>Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014. Copyright © Association for Computing Machinery.; ©ACM, 2014. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014.; This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
Exploiting Metric Structure for Efficient Private Query Release

Zhiyi Huang∗ Aaron Roth†

Abstract
We consider the problem of privately answering queries defined on databases which are collections of points belonging to some metric space. We give simple, computationally efficient algorithms for answering distance queries defined over an arbitrary metric. Distance queries are specified by points in the metric space, and ask for the average distance from the query point to the points contained in the database, according to the specified metric. Our algorithms run efficiently in the database size and the dimension of the space, and operate in both the online release setting, and the offline setting in which they must in polynomial time generate a fixed data structure which can answer all queries of interest. This represents one of the first subclasses of linear queries for which efficient algorithms are known for the private query release problem, circumventing known hardness results for generic linear queries.

1 Introduction
Consider an online retailer who is attempting to recommend products to customers as they arrive. The retailer may have a great deal of demographic information about each customer, both from cookies and from data obtained from tracking networks. Moreover, the retailer will also have information about what other, demographically similar customers have purchased in the past. If the retailer can identify which cluster of customers the new arrival most resembles, then it can likely provide a useful set of recommendations. Note that this problem reduces to computing the average distance from the new arrival to past customers in each demographic cluster, where the distance metric may be complex and domain specific.1

For legal reasons (i.e. to adhere to its stated privacy policy), or for public relations reasons, the retailer may not want the recommendations given to some customer i to reveal information about any specific past customer j ≠ i. Therefore, it would be helpful if the retailer could compute these distance queries while guaranteeing that these computations satisfy differential privacy. Informally, this means that the distances computed from each new customer to the demographic clusters should be insensitive to the data of any single user in the database of past customers.

Distance queries are a subclass of linear or predicate queries, which are well studied in the differential privacy literature [2, 7, 8, 13, 19]. For example, the data analyst could answer k such queries from an ℓ-dimensional metric space, on a database of size n using the private multiplicative weights mechanism of Hardt and Rothblum [13] with error that scales as O(poly(log(k), ℓ)/√n).2 However, none of these mechanisms is computationally efficient, and even for the best of these mechanisms, the running time per query will be exponential in ℓ, the dimension of the space. What’s more, there is strong evidence that there do not exist computationally efficient mechanisms that can usefully and privately answer more than O(n^2) general predicate queries [7, 21]. A major open question in differential privacy is to determine whether there exist interesting subclasses of predicate queries for which efficient algorithms do exist.

In this paper, we show that distance queries using an arbitrary metric are one such class. We give simple, efficient algorithms for answering exponentially many distance queries defined over any metric space with bounded diameter. In the online query release setting, our algorithms run in time nearly linear in the dimension of the space and the size of the private database per query. Our algorithms remain efficient even in the offline query release setting, in which the mechanism must in one shot (and with only polynomial running time) privately generate a synopsis which can answer

∗Stanford University. Email: hzhiyi@stanford.edu. This work was done while the author was a graduate student at University of Pennsylvania. Supported in part by a Simons Graduate Fellowship for Theoretical Computer Science (No. 252128).
†University of Pennsylvania. Supported in part by an NSF CAREER award, NSF grant CNS-1065060, and a Google Focused Research award. Email: aaroth@cis.upenn.edu.
1Note that the most natural metric for this problem may not be defined by an ℓp norm, but may be something more combinatorial, like edit distance on various categorical features.
2All of the mechanisms for answering predicate queries [2, 7, 8, 11, 12, 13, 19] are defined over discrete domains X and have an error dependence on log |X|. In contrast, these queries are defined over continuous ℓ-dimensional domains, and so it is not clear that this previous work even applies. However, metric queries are Lipschitz, and so these mechanisms can be run on a discrete grid with roughly n^Ω(ℓ) points, giving a polynomial dependence on ℓ in the error bounds, but an exponential dependence on ℓ in the running time.
all of the (possibly exponentially many) queries of interest. This represents one of the first high dimensional classes of predicate queries which are known to have computationally efficient private query release mechanisms which can answer large numbers of queries.

**Our Techniques:** At a high level, our mechanism is based on the reduction from online learning algorithms to private query release mechanisms developed in a series of papers [11, 12, 13, 19]. Specifically, we use the fact that an online mistake-bounded learning algorithm for learning the function \( F : C \rightarrow \mathbb{R} \), which maps queries \( f \in C \) to their answers \( f(D) \) on the private database \( D \) generically gives the existence of a private query release mechanism in the interactive setting, where the running time per query is equal to the update time of the learning algorithm.

We observe that if the queries are metric distance queries over some continuous \( \ell_p \) metric space \( X \), then \( F : X \rightarrow \mathbb{R} \) is a convex, Lipschitz-continuous function. Motivated by this observation, we give a simple mistake-bounded learning algorithm for learning arbitrary convex Lipschitz-continuous functions over the unit interval \([0, 1]\) by approximating \( F \) by successively finer piecewise linear approximations. Our algorithm has a natural generalization to the \( \ell \)-dimensional rectangle \([0, 1]^\ell\), but unfortunately the mistake bound of this generalization necessarily grows exponentially with \( \ell \).

Instead, we further observe that if \( X = [0, 1]^\ell \) and is endowed with the \( \ell_1 \) metric, then \( F \) can be decomposed into \( \ell_1 \) 1-dimensional functions, each defined over the unit interval \([0, 1]\). Hence, for the \( \ell_1 \) metric, our learning algorithm can be extended to \([0, 1]^\ell\) with only a linear increase in the mistake bound. In other words, the \( \ell_1 \) metric is an easy metric for differential privacy. In fact, for \( \ell_1 \) distance queries, our algorithm achieves per-query error \( O(\text{poly}(\log(k), \ell)/n^{4/5}) \), improving on the worst-case error guarantees given by inefficient generic query release mechanisms like [13].

Finally, we show that our algorithm can be used to answer distance queries for any metric space that can be embedded into \( \ell_1 \) space using a low sensitivity embedding. A sensitivity \( s \) embedding maps any pair of databases that differ in 1 element into a pair of projected databases that differ in at most \( s \) elements. Oblivious embeddings, such as the almost-isometric embedding from \( \ell_2 \) into \( \ell_1 \) are 1-sensitive [10, 16]. On the other hand, generic embeddings, such as the embedding from an arbitrary metric space into \( \ell_1 \) from Bourgain’s theorem [4, 18] can have sensitivity as high as \( n \).

We observe, however, that for our purposes, the embedding does not need to preserve distances between pairs of database points, or between pairs of query points, but rather only between database points and query points. Therefore, we are able to prove a variant of Bourgain’s theorem, which only preserves distances between query points and database points. This gives a 1-sensitive embedding from any metric space into \( \log k \) dimensional \( \ell_1 \) space, with distortion \( \log k \), which works for any collection of \( k \) distance queries. In particular, this gives us an efficient offline algorithm for answering \( k \) distance queries defined over an arbitrary bounded diameter metric that has multiplicative error \( O(\log k) \) and additive error \( O(\text{polylog}(k)/n^{4/5}) \). Our use of metric embeddings is novel in the context of differential privacy, and we believe that they will be useful tools for developing efficient algorithms in the future as we identify other privacy-friendly metrics in addition to \( \ell_1 \).

**Related Work:** Differential privacy was developed in a series of papers [1, 5, 6], culminating in the definition by Dwork et al. [6]. It is accompanied by a vast literature which we do not attempt to survey.

Dwork et al. [6] also introduced the Laplace mechanism, which together with the composition theorems of Dwork, Rothblum, and Vadhan [8] gives an efficient, interactive method for privately answering nearly \( n^2 \) arbitrary low-sensitivity queries on a database of size \( n \) to non-trivial accuracy. On the other hand, it has been known since Blum, Ligett, and Roth [2] that it is information theoretically possible to privately answer nearly exponentially many predicate queries to non-trivial accuracy, but the mechanism of [2] is not computationally efficient. A series of papers [2, 7, 8, 11, 12, 13, 19] has extended the work of [2], improving its accuracy, running time, and generality. The state of the art is the private multiplicative weights mechanism of Hardt and Rothblum [13]. However, even this mechanism has running time that is linear in the size of the data universe, or in other words exponential in the dimension of the data. Finding algorithms with error bounds similar to [13] while running in time polynomial in the size of the database and the data dimension has been a major open question in the literature since at least [2], who explicitly ask this question.

Unfortunately, a striking recent result of Ullman [21], building on the beautiful work of Dwork, Naor, Reingold, Rothblum, and Vadhan [7], shows that assuming the existence of one way functions, no polynomial time algorithm can answer more than \( O(n^2) \) arbitrary predicate queries. In other words, the Laplace mechanism of [6] is nearly optimal among all computationally efficient algorithms for privately answering queries at a comparable level of generality. This result suggests that to make progress on the problem of computationally effi-
cient private query release, we must abandon the goal of designing mechanisms which can answer arbitrary predicate queries, and instead focus on classes of queries that have some particular structure that we can exploit.

Before this work, there were very few efficient algorithms for privately releasing classes of “high dimensional” predicate queries with worst case error guarantees. Most closely related to our work are the results of Feldman et al. [9], which gave efficient algorithms for releasing Euclidean k-medians queries in a constant dimensional unit ball. Note that when we restrict our attention to Euclidean metric spaces, our queries correspond to 1-median queries. In contrast to [9], we can handle arbitrary metrics, and our algorithms are efficient also in the dimension of the metric space. Blum, Ligett, and Roth [2] gave efficient algorithms for two low dimensional classes of queries: constant dimensional axis aligned rectangles, and large margin halfspaces3. Blum and Roth [3] gave an efficient algorithm for releasing predicate queries defined over predicates with extremely sparse truth tables, but such queries are very rare.

Only slightly more is known for average case error. Gupta et al. [11] gave a poly-time algorithm for releasing the answers (to linear, but non-trivial error) to conjunctions, where the error is measured in the average case on conjunctions drawn from a product distribution. Hardt, Rothblum, and Servedio [14] gave a poly-time algorithm for releasing answers to parity queries, where the error is measured in the average case on parities drawn from a product distribution. Although it is known how to convert average case error to worst-case error using the private boosting technique of Dwork, Rothblum, and Vadhan [8], the boosting algorithm itself is not computationally efficient when the class of queries is large, and so cannot be applied in this setting where we are interested in poly-time algorithms. For the special case of privately releasing conjunctions in ℓ dimensions, Thaler, Ullman, and Vadhan [20], building on the work of Hardt, Rothblum, and Servedio [14], give an algorithm that runs in time $O(2^{\sqrt{\ell}})$, improving on the generic bound of $O(2^\ell)$. Finding a poly-time algorithm for releasing conjunctions remains an open problem.

Metric embeddings have proven to be a useful technique in theoretical computer science, particularly when designing approximation algorithms. See [15] for a useful survey. The specific embeddings that we use in this paper are the nearly isometric embedding from $\ell_2$ into $\ell_1$ using random projections [10, 16], and a variant of Bourgain’s theorem [4, 18], which allows the embedding of an arbitrary metric into $\ell_1$. Our use of metric embeddings is slightly different than its typical use in approximation algorithms. Typically, metric embeddings are used to embed some problem into a metric in which some optimization problem of interest is tractable. In our case, we are embedding metrics into $\ell_1$, for which the information theoretic problem of query release is simpler, since a $d$ dimensional $\ell_1$ metric can be decomposed into $d$ 1-dimensional metric spaces. On the one hand, for privacy, we have a stronger constraint on the type of metric embeddings we can employ: we require them to be low sensitivity embeddings, which map neighboring databases to databases of bounded distance (in the hamming metric). The embedding corresponding to Bourgain’s theorem does not satisfy this property. On the other hand, we do not require that the embedding preserve the distances between pairs of database points, or pairs of query points, but merely between query points and database points. This allows us to prove a variant of Bourgain’s theorem that is 1-sensitive. We think that metric embeddings may prove to be a useful tool in the design of efficient private query release algorithms, and in particular, identifying other privacy-friendly metrics, and the study of other low sensitivity embeddings is a very interesting future direction.

2 Preliminaries

2.1 Model Let $(X, d)$ be an arbitrary metric space. Let $D \in X^n$ be a database consists of $n$ points in the metric space. For the sake of presentation, we will focus on metric spaces with diameter 1 in the rest of this paper. This is simply a matter of scaling: all of our error bounds hold for arbitrary diameter spaces, with a linear dependence on the diameter.

We will consider the problem of answering distance queries while preserving the privacy of the elements in the database, where each query is a point $y \in X$ in the metric space and the answer for a given query $y$ is the average distance from $y$ to the elements in the database, i.e., $\frac{1}{|D|} \sum_{x \in D} d(x, y)$. Let $k$ denote the number of queries and $Q \in X^k$ be the set of distance queries asked by the data analyst. We will let $D(Q) \in \mathbb{R}^k$ denote the exact answer to the queries in $Q$ with respect to database $D$. We will usually use $x_i$’s to denote data points and $y_j$’s to denote query points.

Query Release Mechanisms We will consider two settings for query release in this paper: The first setting is the interactive setting, where the queries are not given upfront but instead arrive online. An interactive query release mechanism needs to provide
3 Releasing \(\ell_1\)-Distance Queries

In this section, we consider \(\ell_1\) distance queries. Let \(\mathcal{X} \subset [0,1]^{\ell} \) and \(d = \|\cdot\|_1\) such that the diameter of \(\mathcal{X}\) (with respect to \(\ell_1\)) is at most 1. We present private, computationally efficient mechanisms for releasing the answers to \(\ell_1\) distance queries in both the interactive and offline setting. These mechanisms for releasing \(\ell_1\) distances will serve as important building blocks for our results for other metrics. First, let us formally state our results. (Recall that \(k\) is the number of queries.)

**Theorem 3.1.** There is an interactive \((\epsilon, \delta)\)-differentially private mechanism for answering distance queries w.r.t. \((\mathcal{X}, \|\cdot\|_1)\) that is \((\alpha, \beta)\)-accurate with

\[
\alpha = O \left( \frac{\epsilon^{9/5} \text{polylog}(\delta^{-1}, k, \beta^{-1})}{(n\epsilon)^{1/5}} \right).
\]

There is also an interactive \(\epsilon\)-differentially private mechanism that is \((\alpha, \beta)\)-accurate for

\[
\alpha = O \left( \frac{\epsilon^{7/3} \text{polylog}(k, \beta^{-1})}{(n\epsilon)^{2/3}} \right).
\]

Both mechanisms run in time \(O(\ell n)\) per query.

**Theorem 3.2.** There is a non-interactive \((\epsilon, \delta)\)-differentially private mechanism for answering distance queries w.r.t. \((\mathcal{X}, \|\cdot\|_1)\) that is \((\alpha, \beta)\)-accurate with

\[
\alpha = O \left( \frac{\epsilon^{9/5} \text{polylog}(\delta^{-1}, n, \ell, \beta^{-1})}{(n\epsilon)^{1/5}} \right).
\]

There is also an non-interactive \(\epsilon\)-differentially private mechanism that is \((\alpha, \beta)\)-accurate for

\[
\alpha = O \left( \frac{\epsilon^{7/3} \text{polylog}(n, \ell, \beta^{-1})}{(n\epsilon)^{2/3}} \right).
\]

Both mechanisms run in time \(O(\ell^3 n^2 \alpha^{-1})\).

**Remark 1.** Note that the non-interactive mechanism has no dependence on the number of queries asked, in either the accuracy or the running time. It in one shot produces a data structure that can be used to accurately answer all \(\ell_1\) queries.

**Proof Overview:** To prove Theorem 3.1, we will use the connection between private query release and online learning, which was established in \([19, 13, 12, 17]\). We will review this connection in Section 3.1. Based on this connection, it suffices to provide an online learning algorithm that learns the function mapping queries to their answers w.r.t. the database using a small number of updates. So we will shift our viewpoint by viewing the
database as a 1-Lipschitz and convex function that maps the query points to real values in $[0, 1]$. The structure of the $\ell_1$ metric allows us to reduce the problem to learning $\ell$ different one dimensional 1-Lipschitz and convex functions, for which we propose in Section 3.2 an online learning algorithm that only needs $O(\alpha^{-1/2})$ updates to achieve an additive error bound $\alpha$. Finally, we combine these ingredients to give an interactive differentially private mechanism for releasing answers for $\ell_1$ distance queries in Section 3.3 to prove Theorem 3.1. Roughly speaking, the interactive mechanism will always maintain a hypothesis function that maps queries to answers and it will update the hypothesis function using the online learning algorithm whenever the hypothesis function makes a mistake. Finally, we show that there is an explicit set of different one dimensional 1-Lipschitz and convex functions, for which we propose in Section 3.2 updates to achieve an additive error bound $O(1/n)$ sensitive in the private database $D$. Concretely, our variant of IDC is defined as follows.

**Definition 3.** ([19, 13, 12]) Given an error bound $\alpha > 0$ and an error tolerance $1 > c > 0$, an iterative database construction algorithm with respect to $\ell_1$ distance queries plays the following game with an adversary in a sequence of rounds $(t = 1, 2, \ldots)$:

1. The algorithm maintains a hypothesis function $\hat{F}_t : X \rightarrow \mathbb{R}$ for each round $t$ on which it can evaluate queries. $\hat{F}_1$ is initialized to be some default function at the beginning of round 1.

2. In each round $t \geq 1$, the adversary (adaptively) chooses a query $y_i \in X$, at which point the algorithm predicts a query value $\hat{F}_t(y_i)$. If $|\hat{F}_t(y_i) - F_D(y_i)| > \alpha$, then we say the algorithm has made a mistake. At this point, the algorithm receives $\ell + 1$ values $a_0, a_1, \ldots, a_\ell \in \mathbb{R}$ such that

   \[
   a_0 \in [F_D(y_i) - \alpha, F_D(y_i) + \alpha]
   \]

   and, $\forall i \in [\ell]$,

   \[
   a_i \in [\partial_i F_D(y_i) - \alpha, \partial_i F_D(y_i) + \alpha] .
   \]

   The algorithm then updates its hypothesis function to obtain $\hat{F}_{t+1}$ using this information.

**Definition 4.** (Mistake Bound) An iterative database construction algorithm has a mistake bound $m : \mathbb{R}_+ \rightarrow \mathbb{N}_+$, if for any given error bound $\alpha$, no adversary can (adaptively) choose a sequence of queries to force the algorithm to make $m(\alpha) + 1$ mistakes.

**Lemma 3.1.** ([19, 13, 12]) If there is an iterative database construction algorithm for releasing $\ell_1$ distance queries with mistake bound $m(\alpha)$ and error tolerance $c$, then there is an $(\epsilon, \delta)$-differentially private mechanism in the interactive setting that is $(\alpha, \beta)$-accurate, for $\alpha$ satisfying

\[
\alpha = \frac{1}{n\epsilon} \cdot 3000 \cdot \ell \cdot m(\alpha) \log(4/\delta) \log(k/\beta) .
\]

There is also an $\epsilon$-differentially private mechanism in the interactive setting that is $(\alpha, \beta)$-accurate, for $\alpha$ satisfying

\[
\alpha = \frac{1}{n\epsilon} \cdot 3000 \cdot \ell \cdot m(\alpha) \log(k/\beta) .
\]

\[\text{As we apply this framework to the $\ell_1$ case, the subgradient of } F_D \text{ is well-defined. When } F_D \text{ is not differentiable at the query point, we assume that the learning algorithm learns an arbitrary subgradient.}\]
Moreover, the per-query running time of the mechanism equals (up to constant factors) the running time of the per-round running time of the iterative database construction algorithm.

Representing $\ell_1$ Databases as Decomposable Convex Functions Consider a database $D$ where the universe is the $\ell$-dimensional unit cube $\mathcal{X} = [0, 1]^\ell$ endowed with the $\ell_1$ metric. Then the function mapping queries $y \in \mathcal{X}$ to their answers takes the form: $F_D(y) = \frac{1}{n} \sum_{x \in D} ||x - y||_1$, which is a $1$-Lipschitz convex function of $y$. We wish to proceed by providing an iterative database construction for $\ell_1$ distance queries using these properties. Observe that because we are working with the $\ell_1$ metric, we can write:

$$F_D(y) = \sum_{i=1}^{\ell} F_D^{(i)}(y), \text{ where } F_D^{(i)}(y) = \frac{1}{n} \sum_{x \in D} |x_i - y_i| .$$

Observe that each function $F_D^{(i)}(y)$ is $1$-Lipschitz and convex, and has a $1$-dimensional range $[0, 1]$. Therefore, to learn an approximation to $F_D(y)$ up to some error $\varepsilon$, it suffices to learn an approximation to each $F_D^{(i)}(y)$ to error $\varepsilon/\ell$. This is the approach we take.

### 3.2 Learning 1-Lipschitz Convex Functions

In this section we study the problem of iteratively constructing an arbitrary continuous, $1$-Lipschitz, and convex function $G: [0, 1] \mapsto [0, 1]$ up to some additive error $\alpha_1$ with noisy oracle access to the function. Here, the oracle can return the function value $G(x)$ and the derivative $G'(x)$ given any $x \in [0, 1]$ up to an additive error of $\alpha_1/4$. Here, we assume the derivative $G'$ is well defined in $[0, 1]$: If $G$ is not differentiable at $x$, then we assume the derivative $G'(x)$ is (consistently) defined to be any value between the left and right derivatives at $x$.

**Learning 1-D Functions with an Accurate Oracle** We will maintain a hypothesis piece-wise linear function $\hat{G}(x)$ via the algorithm given in Figure 1. It remains to analyze the number of updates needed by this algorithm in order to learn a piece-wise linear function $\hat{G}$ that approximates $G$ everywhere up to additive error $\alpha_1$.

For any $1$-Lipschitz (possibly non-convex) function $G$ and any error bound $\alpha_1 > 0$, the algorithm in Figure 1 will make at most $1/\alpha_1$ mistakes. This is because the Lipschitz condition implies that the tangent line at each update point $x_t^*$ is a good approximation (up to error $\alpha_1$) in the neighborhood $[x_t^* - \alpha_1, x_t^* + \alpha_1]$, and hence the update points are at least $\alpha_1$ away from each other. Further, it is easy to construct examples where this bound is tight up to a constant. By using the convexity of function $G$, we can improve the mistake bound to $O\left(\frac{1}{\sqrt{\alpha_1}}\right)$.

**Lemma 3.2.** For any $1$-Lipschitz convex function $G$ and any given error bound $\alpha_1 \in (0, 1)$, the algorithm in Figure 1 will make at most $\frac{3}{\alpha_1}$ updates.

**Proof.** Consider any two update points $x_t^*$ and $x_t^*$. Let $\alpha < t$. Then, by our assumption, the tangent line at $x_t^*$ does not approximate the function value of $f$ at $x_t^*$ up to an additive error of $\alpha$. Therefore, we get that

$$\alpha_1 < G(x_t^*) - (G'(x_t^*)(x_t^* - x_1^*) + G(x_1^*)) = G(x_t^*) - G(x_1^*) = G'(x_t^*)(x_t^* - x_1^*) \leq G'(x_t^*)(x_t^* - x_1^*) - G'(x_1^*)(x_1^* - x_1^*) = (G'(x_t^*) - G'(x_1^*)) \left(\frac{x_t^* - x_1^*}{\sqrt{\alpha_1}}\right) ,$$

where the second inequality is by the convexity of $f$.

Next, consider a maximal set of update points in sorted order: $-1 \leq x_1 < \cdots < x_T \leq 1$. Since $G$ is convex and $1$-Lipschitz, we have that $-1 < G'(x_t^*) < \cdots < G'(x_T^*) \leq 1$. Therefore, we get that

$$2 \cdot 1 \geq \frac{G'(x_T^*) - G'(x_1^*)}{(x_T^* - x_1^*)} = \sum_{t=1}^{T-1} \frac{G'(x_{t+1}^*) - G'(x_t^*)}{(x_{t+1}^* - x_t^*)} \geq \left(\sum_{t=1}^{T-1} \frac{1}{\sqrt{\alpha_1}}\right)^2 = \left(\frac{\alpha_1}{\sqrt{\alpha_1}}\right)^2 .$$

Here, the first inequality is by $G'(x_t^*) \in [-1, 1]$ and $x_t^* \in [0, 1]$ for $t = 1, \ldots, T$; the second inequality is a simple application of the Cauchy-Schwartz inequality; the last inequality is by equation (3.1). So by the above inequality, the number of mistakes is at most $T \leq \frac{\sqrt{2}}{\sqrt{\alpha_1}} + 1 < \frac{3}{\sqrt{\alpha_1}}$.

**Learning 1-D Functions with a Noisy Oracle** Note that the domain of the function is $[0, 1]$. So if the tangent line at $x'$ approximates the function value at $x$ up to additive error $\frac{\alpha_1}{2}$, i.e.,

$$G(x) - G(x') + G'(x')(x - x') \leq \frac{\alpha_1}{2} ,$$

then a noisy version of the tangent line

$$\tilde{G}(x') + \tilde{G}'(x')(x - x') ,$$

where

$$\tilde{G}(x') \in [G(x') - \frac{\alpha_1}{4}, G(x') + \frac{\alpha_1}{4}].$$
Learning a 1-Lipschitz and Convex Function

Maintain \( \hat{G}(x) = \max \{a_k \cdot x + b_k\} \) where \( a_k \in [-1, 1] \) and \( b_k \in \mathbb{R} \) define a set of linear functions.

**Initial Step:** Let \( a_0 = 0 \) and \( b_0 = 0 \).

**Update Step** \( t \geq 1 \): While the update generator returns a distinguishing point \( x_t^* \), we shall add the tangent line at \( x_t^* \) with respect to function \( g \) to the set of linear functions, i.e., \( a_t = G'(x_t^*) \) and \( b_t = G(x_t^*) - G(x_t^*) \cdot x_t^* \).

Figure 1: An algorithm for learning a 1-Lipschitz and convex one-dimensional function by approximating it with a piece-wise linear function. The algorithm always predicts according to \( \hat{G} \). When it makes a mistake, it is given an update point \( x_t^* \) together with \( G(x_t^*) \) and \( G'(x_t^*) \).

\[ G'(x') \in [G'(x') - \frac{\alpha_1}{4}, G'(x') + \frac{\alpha_1}{4}] \]

will approximate the value at \( x \) up to additive error \( \alpha_1 \). Hence, the mistake bound of the algorithm in Figure 1 for learning a 1-Lipschitz and convex function up to additive error \( \alpha_1 \) using a noisy oracle is no more than the mistake bound for learning the same function up to additive error \( \frac{\alpha_1}{2} \) with an accurate oracle. Hence, the mistake bound is still of order \( O(\frac{1}{\sqrt{\alpha_1}}) \).

**Lemma 3.3.** For any 1-Lipschitz convex function \( g \) and any given error bound \( \alpha_1 \in (0, 1) \), the algorithm in Figure 1 will make at most \( O(\frac{1}{\sqrt{\alpha_1}}) \) updates with an \( \frac{\alpha_1}{4} \)-noisy oracle.

**Learning Decomposable Functions** Suppose we want to learn an \( \ell \)-dimension decomposable convex function \( F_D = \sum_{i=1}^{\ell} F_D^{(i)}(y_i) \) up to additive error \( \alpha \), where each \( F_D^{(i)} \) is convex and 1-Lipschitz. Then, it suffices to learn the 1-Lipschitz convex functions \( F_D^{(i)} \) for each coordinate up to error \( \alpha_1 = \frac{\alpha}{\ell} \). So as a simple corollary of Lemma 3.3, we have the following lemma:

**Lemma 3.4.** For any function \( F_D : [0, 1]^{\ell} \rightarrow \mathbb{R} \) such that:

1. \( F_D(y) = \sum_{i=1}^{\ell} F_D^{(i)}(y_i) \) where each \( F_D^{(i)} : [0, 1] \rightarrow [0, 1] \) is 1-Lipschitz and convex, and:

2. For every \( y \in [0, 1]^{\ell} \) and every \( i \in [\ell] \): \( F_D^{(i)}(y) \) and \( (F_D^{(i)}(y))^\prime \) are \( 1/n \)-sensitive in \( D \)

there is an iterative database construction algorithm for \( F_D \) using a collection of \( 2\ell \) functions \( S \) with respect to an error tolerance \( 1/(4\ell) \) that has a mistake bound of \( m(\alpha) = O(\ell^{3/2}/\alpha^{1/2}) \).

**Proof.** Let \( \alpha_1 = \frac{\alpha}{\ell} \). Consider the following algorithm:

1. The algorithm maintains a hypothesis function \( \hat{F}_t = \sum_{i=1}^{\ell} F_t^{(i)} \) by maintaining \( \ell \) one-dimensional piecewise-linear hypothesis functions \( \hat{F}_t^{(i)} : [0, 1] \rightarrow [0, 1] \) for each \( i \in [\ell] \) via the one-dimension learning algorithm with error tolerance \( \alpha_1 \), and letting \( F_t = \sum_{i=1}^{\ell} F_t^{(i)} \).

2. If the algorithm makes a mistake on query \( y_i \in [0, 1]^{\ell} \), then the algorithm asks query \( y_i \) to each of the one-dimensional learning algorithms. On any of the one-dimensional learning algorithms \( i \) on which a mistake is made, the algorithm queries two values: \( F_t^{(i)}(y_i) \) and \( (F_t^{(i)}(y_i))^\prime \), tolerating additive error up to \( \alpha_1/(4\ell) \), and updates the hypothesis \( \hat{F}_t^{(i)} \), \( i = 1, \ldots, \ell \), accordingly using the one dimensional learning algorithm. Note that this leads to at most \( |S| = 2\ell \) queries per update.

3.3 **Proofs of Theorem 3.1 and Theorem 3.2**

**Proof.** [Theorem 3.1] Since the \( \ell_1 \) distance function in each coordinate has range \( [0, 1] \) and is 1-Lipschitz, we get that \( F_t^{(i)} \) and the derivative \( (F_t^{(i)})^\prime \) are \( O(1/n) \)-sensitive. So by Lemma 3.4, there is an iterative database construction algorithm for releasing answers to \( \ell_1 \) distance queries that uses a set \( S \) of \( 2\ell \) \( O(1/n) \)-sensitive queries with error \( \alpha_1 \), and the algorithm has mistake bound \( O(\ell^{3/2}/\alpha^{1/2}) \).

By plugging the parameters of the above iterative database construction algorithm to Lemma 3.1, we get that there is an \((\epsilon, \delta)\)-differentially private mechanism in the interactive setting that is \((\alpha, \beta)\)-accurate for
releasing distance queries with respect to metric space $([0, 1]^d, \|\cdot\|_1)$, for $\alpha$ satisfying

$$\frac{\alpha}{\ell} = O\left(\frac{1}{n^c} \sqrt{\frac{\ell^5/2}{\alpha 1/2}} \log(4/\delta) \log(k/\beta)\right).$$

Solving the above we get that

$$\alpha = O\left(\frac{\ell^{5/2}}{n^c} \alpha^{1/2} \log(k/\beta)\right).$$

We also get that there is an $\epsilon$-differentially private mechanism in the interactive setting that is $(\alpha, \beta)$-accurate, for $\alpha$ satisfying

$$\frac{\alpha}{\ell} = O\left(\frac{1}{n^c} \log(k/\beta)\right).$$

Solving the above we get that

$$\alpha = O\left(\frac{\ell^{5/2}}{n^c} \log^2(\beta/k)\right).$$

The analysis of the running time per query is straightforward and hence omitted.

**Proof.** [Theorem 3.2] Consider running the online query release mechanism with accuracy $\alpha' = \alpha/2$. To give an offline mechanism, we simply describe a fixed set of $\ell/\alpha'$ queries that we can make to each of the $\ell$ one-dimensional learning algorithms maintaining $\hat{F}^{(i)}_D$ that guarantees that for each $y \in [0, 1]$,

$$|\hat{F}^{(i)}_D(y) - F^{(i)}_D(y)| \leq \alpha/\ell.$$

Once we have this condition, we know that for any $y \in [0, 1]^d$:

$$|\hat{F}_D(y) - F_D(y)| \leq \alpha.$$

The queries are simple: we just take our query set to be a grid: $T = \{0, \alpha'/\ell, 2\alpha'/\ell, 3\alpha'/\ell, \ldots, 1\}$. By the guarantees of the 1-dimensional learning algorithm, we have that for every $y \in T$, $|\hat{F}^{(i)}_D(y) - F^{(i)}_D(y)| \leq \alpha'/\ell$. Moreover, by the fact that $\hat{F}^{(i)}_D$ is 1-Lipschitz, and for every $y \in [0, 1]$, $d(y, T) \leq \alpha'/\ell$, we have that for every $y \in [0, 1]$,

$$|\hat{F}^{(i)}_D(y) - F^{(i)}_D(y)| \leq 2\alpha'/\ell = \alpha/\ell,$$

which is the condition we wanted. In total, we make $2\ell^2/\alpha$ queries, and the theorem follows by instantiating the guarantees of the online mechanism with $k = 2\ell^2/\alpha$.

### 4 Releasing Arbitrary Distance Queries

In this section, we will show how to answer distance queries with respect to other metric spaces. We will reduce the problem to answering to $\ell_1$ distance queries via metric embeddings. An embedding from a metric space $(X, d)$ to another metric space $(Y, d')$ is a mapping $\pi : X \times X \rightarrow Y$, where $\pi(x, X)$ is the embedding of $x$ w.r.t. a subset of points $X \subseteq X$. The usefulness of an embedding is measured by how much the embedding distorts the distance between any pair of points in $X$.

Note that for the purpose of answering distance queries, the usual definition of distortion is too strong in the sense that it considers the worst-case distortion for every pair of points in the metric space while we only need to preserve the distance between every data-query pair. Therefore, we will consider the following weaker notion of expansion, contraction, and distortion. (Recall that $D$ and $Q$ are the set of data points and the set of query points respectively.)

**Definition 5.** The expansion of an embedding $\pi$ from $(X, d)$ to another metric space $(Y, d')$ is

$$\max_{x \in X, \pi x \in Y, y \in Q} \frac{d'(\pi(x, X), \pi(y, X))}{d(x, y)}.$$

The contraction of the embedding is

$$\max_{x \in X, \pi x \in Y, y \in Q} \frac{d(x, y)}{d'(\pi(x, X), \pi(y, X))}.$$

The distortion of an embedding is the product of its expansion and contraction.

In the rest of this section, we will always choose $X = D$ and hence omit the second parameter of the embedding (i.e., $\pi(x) = \pi(x, D)$) when appropriate for brevity. We will let the target metric $(Y, d')$ to be the $\ell_1$ metric $([0, 1]^d, \|\cdot\|_1)$ and scale the embedding such that the expansion is 1.

#### 4.1 1-Sensitive Metric Embeddings

Suppose we have an embedding from $(X, d)$ to $([0, 1]^d, \|\cdot\|_1)$ with expansion 1 and contraction $C$. In some cases, the dimension $\ell$ of the target $\ell_1$ space may depend on the contraction $C$. We will embed both the data points and the query points into $([0, 1]^d, \|\cdot\|_1)$ and release distance queries via the the mechanism for $\ell_1$. Concretely, consider mechanisms $M_{\epsilon, \delta}$ and $M_c$ given in Figure 2.

We first consider the accuracy of the mechanisms: they lose a multiplicative factor due to the embedding and an additive factor due to answering the $\ell_1$ queries privately. So we have the following:
Releasing distance queries via embedding into $\ell_1$

**Input:** Database $\mathcal{D}$. Set of queries $\mathcal{Q}$. 1-sensitive embedding $\pi$ from $(X,d)$ to $([0,1]^d, \| \cdot \|_1)$.

1. Construct a proxy database $\mathcal{D}'$ for releasing $\ell_1$ distances by letting $\pi(x) \in \mathcal{D}'$ for all $x \in \mathcal{D}$.
2. For every $y \in \mathcal{Q}$, let $\epsilon, \delta$-differentially private mechanism (resp., $\epsilon$-differentially private mechanism) for $\ell_1$ answer $\frac{1}{\pi} \sum_{x \in \mathcal{D}} \| \pi(x) - \pi(y) \|_1$, and release it as the answer to query $y$.

Figure 2: An $(\epsilon, \delta)$-differentially private mechanism $M_{\epsilon, \delta}$ (resp., $\epsilon$-differentially private mechanism $M$) for releasing distance queries via embedding into $\ell_1$

**Theorem 4.1.** If $\pi$ has expansion 1 and contraction $C$, and we use the $(\epsilon, \delta)$-differentially private mechanism for $\ell_1$ in Figure 2, then with probability at least $1 - \beta$,

$$\forall y \in \mathcal{Q}: \quad \frac{1}{\pi} \sum_{x \in \mathcal{D}} d(x, y) - \tilde{O} \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) \leq M_{\epsilon, \delta}(\mathcal{D}, y)$$

$$\leq \sum_{x \in \mathcal{D}} d(x, y) + \tilde{O} \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) ,$$

If we instead use the $\epsilon$-differentially private mechanism for $\ell_1$, then with probability at least $1 - \beta$,

$$\forall y \in \mathcal{Q}: \quad \frac{1}{\pi} \sum_{x \in \mathcal{D}} d(x, y) - O \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) \leq M_{\epsilon}(\mathcal{D}, y)$$

$$\leq \sum_{x \in \mathcal{D}} d(x, y) + O \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) .$$

**Remark 3.** If the embedding is nearly isometric, i.e., we can achieve contraction $1 + \alpha$ for any small $\alpha > 0$ by embedding into an $\ell$-dimensional $\ell_1$ space, then we will choose the optimal additive error bound $\alpha_{\epsilon, \delta}$ and $\alpha_{\epsilon}$ for $(\epsilon, \delta)$- and $\epsilon$-differential privacy respectively such that

$$\alpha_{\epsilon, \delta} = \tilde{O} \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) \quad \text{and} \quad \alpha_{\epsilon} = \tilde{O} \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) .$$

**Proof.** Let us prove the error bound for $\epsilon$-differential privacy. The proof of the error bound for $(\epsilon, \delta)$-differential privacy is similar. We will view the embedding $\pi$ as from $(X,d)$ to $(\pi(X), \| \cdot \|_1)$. Since the embedding $\pi$ has expansion 1, the image $\pi(x)$ of $X$ has diameter 1 as well. Let $M_{\epsilon}^{\ell_1}$ denote the $\epsilon$-differentially private mechanism for releasing answers to the $\ell_1$ distance queries. Then we have that

$$M_{\epsilon}(y, D) = M_{\epsilon}^{\ell_1}(\pi(y), D')$$

$$\leq \sum_{x \in \mathcal{D}} \| \pi(x) - \pi(y) \|_1 + \tilde{O} \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right)$$

$$\leq \sum_{x \in \mathcal{D}} d(x, y) + \tilde{O} \left( \frac{\epsilon^{1/\delta} / n^{2/3} \epsilon^{3/4}}{\pi} \right) .$$

The proof of the lower bound is similar, hence omitted.

Next we will turn to the privacy guarantee of the mechanism. Since we are using either an $(\epsilon, \delta)$-differentially private mechanism or an $\epsilon$-differentially private mechanism for releasing answers to the $\ell_1$ distance queries with respect to the proxy database $D'$, it suffices to ensure that the embeddings of neighboring databases remain neighboring databases. In general, the embedding of some point $x$ may be defined in terms of other data points $y$, which would violate this condition. Formally, we want our embeddings to be 1-sensitive:

**Definition 6.** An embedding $\pi$ from $(X,d)$ to $([0,1]^d, \| \cdot \|_1)$ is 1-sensitive if changing a data point $x_i \in \mathcal{D}$ will only change the embedding $\pi(x_i, \mathcal{D})$ of $x_i$ and will not affect the embedding $\pi(x_j, \mathcal{D})$ of other $x_j \in \mathcal{D}$ for any $j \neq i$.

**Theorem 4.2.** If the embedding $\pi$ is 1-sensitive, and if we use the $(\epsilon, \delta)$-differentially private mechanism (resp., $\epsilon$-differentially private mechanism) for releasing answers to the $\ell_1$ distance queries, then the mechanism $M_{\epsilon, \delta}$ (resp., $M_{\epsilon}$) is $(\epsilon, \delta)$-differentially private (resp., $\epsilon$-differentially private).

**Proof.** For any two neighboring databases $\mathcal{D}_1$ and $\mathcal{D}_2$, the resulting proxy databases $\mathcal{D}_1'$ and $\mathcal{D}_2'$ in Figure 2 will either be the same or be neighboring databases since the embedding $\pi$ is 1-sensitive. Since we are using an $\epsilon$-differentially private mechanism for releasing $\ell_1$ distances over the proxy databases, we get that for any set of queries $\mathcal{Q}$ and for any subset $S$ of possible answers,

$$\Pr[M_{\epsilon}(\mathcal{D}, \mathcal{Q}) \in S] = \Pr[M_{\epsilon}(\mathcal{D}_1', \pi(\mathcal{Q})) \in S]$$

$$\leq \exp(\epsilon) \Pr[M_{\epsilon}(\mathcal{D}_2', \pi(\mathcal{Q})) \in S]$$

$$= \exp(\epsilon) \Pr[M_{\epsilon}(\mathcal{D}_2, \mathcal{Q}) \in S] .$$

So mechanism $M_{\epsilon}$ is $\epsilon$-differentially private. The proof for $(\epsilon, \delta)$-differential privacy is similar.

**Remark 4.** In principle, we can also consider $s$-sensitive embeddings for small $s$. However, we are not aware of any useful embeddings of this kind. So we will focus on 1-sensitive embeddings.

**Remark 5.** If the embedding $\pi$ is independent of the set $\mathcal{Q}$ of queries, then the mechanisms in Figure 2 can
be made interactive or non-interactive by using the interactive or non-interactive mechanisms respectively for releasing answers to the \( \ell_1 \) distance queries. If the embedding is a function of the query set, then the mechanism will be non-interactive, because potentially all of the queries may be needed to construct the embedding of the database.

4.2 Releasing Euclidean Distance via an Oblivious Embedding

An embedding is oblivious if the embedding function \( \pi \) is defined independent on the data points. Any oblivious embedding is 1-sensitive.

Let us consider releasing distance queries with respect to Euclidean distance. From the metric embedding literature we know that there exists an almost isometric embedding from \( \ell_2 \) space to the \( \ell_1 \) space with distortion \( \exp(\sqrt{\log(1/\alpha)}) \). Unfortunately, this embedding is not oblivious and does not have low sensitivity. However, recall that for the purpose of releasing distance queries, we only need to preserve the distances between all data-query pairs. In other words, it is okay to have the distances between data points (and likewise, between query points) to be highly distorted. Further, we show that for this weaker notion of embedding, there is a variant of Bourgain’s theorem using an embedding that is oblivious to the data points, and hence 1-sensitive.

Concretely, we will consider the embedding given in Figure 3. The idea is to define the embedding only using the query points and we will show this is enough to preserve the distances from any point in the metric space to the query points with high probability.

**Theorem 4.4.** (1-Sensitive Bourgain) Let \( \pi \) be the embedding in Figure 3. For any data point \( x \in X \) and any query point \( y \in \mathcal{Q} \), with probability at least \( 1 - \frac{1}{m^2 k} \),

\[
\frac{1}{64 \log k} d(x, y) \leq \|\pi(x) - \pi(y)\|_1 \leq d(x, y).
\]

The proof of the above theorem is very similar to one of the proofs for Bourgain’s original theorem. The expansion bound is identical. The contraction bound will only guarantee the embedded distance of two pair of points \( x \in D \) and \( y \in Q \) satisfies \( d'(\pi(x), \pi(y)) \geq O(\frac{1}{\log k})(d(x, y) - d(x, Q)) \). We observe that the additive loss of \( O(\frac{1}{\log k})d(x, Q) \) can be avoided by using an additional \( O(\log k + \log n) \) dimensions in the embedding. We include the proof below for completeness.

**Proof.** Expansion: By triangle inequality,

\[
|\pi_{ij}(x) - \pi_{ij}(y)| = \frac{1}{K \log k} |d(x, S_{ij}) - d(y, S_{ij})| 
\leq \frac{1}{K \log k} d(x, y).
\]

Summing over \( 0 \leq i \leq \log k \) and \( 1 \leq j \leq K \) we have

\[
\sum_{i=0}^{\log k} \sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \leq d(x, y).
\]

**Contraction:** Let us first define some notation. Let \( r_i \) and \( r_i' \) denote the smallest radius such that the closed ball (with respect to metric \( (X, d) \), similar hereafter) \( B(x, r_i) \) and \( B(y, r_i') \) respectively contains at least \( 2^{i-1} \) query points. Let \( r^*_i = \max\{r_i, r'_i\} \). We will have that \( r^*_i \) is non-decreasing in \( i \). Let \( s' \) denote the largest index such that \( r^*_i + r^*_{i-1} \leq d(x, y) \). Redefine \( r^*_i \) to be \( d(x, y) - r^*_{i-1} \). We have \( r^*_i \geq \frac{d(x, y)}{2} \). We will need to following lemmas.
1-sensitive variant of Bourgain: Embedding an arbitrary metric space \((X, d)\) into \(\ell_1\)

Pre-processing: For \(1 \leq i \leq \log k\) and \(1 \leq j \leq K\), where \(K\) is chosen to be \(512(\log k + \log n)\), choose a random subset \(S_{ij}\) of the query points by picking each query point \(y\) independently with probability \(2^{-(i-1)}\).

Embedding: Given \(x\) in the metric space \((X, d)\), embed it into \(\{\pi_{ij}(x)\}_{0 \leq i \leq \log k, 1 \leq j \leq K}\) in the \(O(K \log k)\)-dimension \(\ell_1\) space by letting \(\pi_{ij}(x) = \frac{1}{K \log k} d(x, S_{ij})\).

Figure 3: A randomized 1-sensitive embedding of an arbitrary metric space \((X, d)\) into an \(O(\log^2 k + \log k \log n)\)-dimension \(\ell_1\) space with \(O(\log k)\) distortion

**Lemma 4.2.** For any \(1 \leq i \leq i'\), we have

\[
\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \geq \frac{1}{32 \log k} (r_i^* - r_{i-1}^*) \quad \text{with probability at least } 1 - \frac{1}{n^{K \log k}}.
\]

**Proof.** Suppose \(r_i^* = r_i\) (the other case is similar). Consider the open ball \(B^o(x, r_i^*)\) and the closed ball \(B(y, r_{i-1}^*)\). By definition, the number of query points in \(B^o(x, r_i^*)\) is less than \(2^{i-1}\), and the number query points in \(B(y, r_{i-1}^*)\) is at least \(2^{i-2}\). Since for each \(1 \leq j \leq K\), the set \(S_{ij}\) pick each query point independently with probability \(2^{-(i-1)}\), the probability that \(S_{ij} \cap B^o(x, r_i^*) = \emptyset\) is at least \((1 - 2^{-(i-1)})^{2^{i-1}} \geq \frac{1}{2}\), while the probability that \(S_{ij} \cap B(y, r_{i-1}^*) \neq \emptyset\) is at least \(1 - (1 - 2^{-(i-1)})^{2^{i-2}} \geq 1 - e^{-\frac{1}{2}}\). In sum, with probability at least \(\frac{1}{2}(1 - e^{-\frac{1}{2}}) > \frac{1}{32}\), we have both \(S_{ij} \cap B^o(x, r_i^*) = \emptyset\) and \(S_{ij} \cap B(y, r_{i-1}^*) \neq \emptyset\), which indicates that \(d(x, S_{ij}) \geq r_i^*\) and \(d(y, S_{ij}) \leq r_{i-1}^*\) and therefore

\[
(4.2) \quad |\pi_{ij}(x) - \pi_{ij}(y)| \geq \frac{1}{K \log k} (r_i^* - r_{i-1}^*).
\]

Further, by the additive form of Chernoff-Hoeffding theorem, we get that with probability at least \(1 - 2^{-32} \leq 1 - \frac{1}{n^{K \log k}}\), (4.2) holds for at least \(\frac{1}{2}\) of the \(\pi_{ij}\)'s. So we conclude that with probability at least \(1 - \frac{1}{n^{K \log k}}\),

\[
\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \geq \frac{1}{32 \log k} (r_i^* - r_{i-1}^*).
\]

**Lemma 4.3.** \(\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| = \frac{1}{K \log k} r_i^*\).

**Proof.** It is easy to see that \(r_i^* = 0\) because \(y\) itself is a query point and \(r_i^* = r_i = d(x, Q)\). Note that for every \(j\), \(S_{ij}\) equals the set of query points. So we always have \(d(x, S_{ij}) = d(x, Q) = r_i^*\) and \(d(y, S_{ij}) = 0\). Therefore, \(|\pi_{ij}(x) - \pi_{ij}(y)| = \frac{1}{K \log k} r_i^*\) for \(1 \leq j \leq K\), and summing up completes the proof.

By Lemma 4.2 and union bound, with probability at least \(1 - \frac{1}{n^{K \log k}}\), we have

\[
\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \geq \frac{1}{32 \log k} (r_i^* - r_{i-1}^*)
\]

for all \(1 \leq i \leq i'\). By Lemma 4.3 we have

\[
\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \geq \frac{1}{32 \log k} r_i^* \geq \frac{1}{64 \log k} d(x, y).
\]

Summing them up we get that

\[
\sum_{i=1}^{K} \sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \geq \frac{1}{32 \log k} r_i^* \geq \frac{1}{64 \log k} d(x, y).
\]

As a corollary of Theorem 4.4 and union bound we have

**Corollary 4.1.** Let \(\pi\) be the embedding in Figure 3. Then with probability at least \(1 - \frac{1}{n}\), we have that for any data point \(x \in D\) and any query points \(y \in Q\),

\[
\frac{1}{64 \log k} d(x, y) \leq ||\pi(x) - \pi(y)||_1 \leq d(x, y).
\]

Hence, there exists an embedding of an arbitrary metric to an \(\ell_1\) metric with distortion \(O(\log k)\) that is 1-sensitive because it is oblivious to the data points. The dimension of the resulting \(\ell_1\) metric is \(O(\log^2 k + \log k \log n)\). We remark that the expansion guarantee may fail with some small probability, in which case the diameter of our embedding may be greater than 1. This would appear to require us to move to an \((\epsilon, \delta)\)-privacy guarantee, but it does not: when computing \(\ell_1\) distances between points \(x, y\), we can instead compute \(\min(1, |\pi(x) - \pi(y)|_1)\). In the high probability event in which the expansion guarantee of the embedding holds, this will be exactly equal to the true distance between the embeddings of the points \(x\) and \(y\). In the small probability event in which the expansion guarantee fails, the resulting queries will remain \(1/n\) sensitive in the private data. So by combining Theorem 4.1, Theorem 4.2, and Theorem 4.4 we have the following theorem.

**Theorem 4.5.** For any metric space \((\mathcal{X}, d)\), there is a non-interactive \((\epsilon, \delta)\)-differentially private mechanism \(M_{\epsilon, \delta}\) for answering any \(k\) distance queries w.r.t. \((\mathcal{X}, d)\), such that with high probability, for any query \(y \in Q\),

\[
O\left(\frac{1}{\log k}\right) \frac{1}{n} \sum_{x \in D} d(x, y) - \hat{O}\left(\frac{1}{n^{\log^{1/4} k}}\right) \leq M_{\epsilon, \delta}(y) \leq \frac{1}{n} \sum_{x \in D} d(x, y) + \hat{O}\left(\frac{1}{n^{\log^{1/4} k}}\right).
\]
There is also a non-interactive $\epsilon$-differentially private mechanism $M_\epsilon$, such that with high probability, for any query $y \in \mathcal{Q}$,

$$O\left(\frac{1}{\log k}\right)\sum_{x \in \mathcal{D}} d(x, y) - \tilde{O}\left(\frac{1}{n^{1/2} \epsilon^2 \log^2 n}\right) \leq M_\epsilon(y) \leq \sum_{x \in \mathcal{D}} d(x, y) + \tilde{O}\left(\frac{1}{n^{1/2} \epsilon^2 \log^2 n}\right).$$

Both mechanisms run in time $\text{poly}(n, k)$.

References


