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 query 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 it's 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 $j \neq i$. 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 in 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 $O(\text{poly}(\log(k), \ell)/\sqrt{n})$.[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.

[1] Note that the most natural metric for this problem may not be defined by an ℓ_p norm, but may be something more combinatorial, like edit distance on various categorical features.

[2] All 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 $n^{\Omega(\ell)}$ 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 $F : C \to \mathbb{R}$, 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 ℓ_p metric space $\mathcal{X}$, then $F : \mathcal{X} \to \mathbb{R}$ 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 ℓ-dimensional rectangle $[0, 1]^\ell$, but unfortunately the mistake bound of this generalization necessarily grows exponentially with ℓ.

Instead, we further observe that if $\mathcal{X} = [0, 1]^\ell$ and is endowed with the ℓ_1 metric, then F can be decomposed into ℓ 1-dimensional functions, each defined over the unit interval $[0, 1]$. Hence, for the ℓ_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 ℓ_1 metric is an easy metric for differential privacy.* In fact, for ℓ_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 ℓ_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 ℓ_2 into ℓ_1 are 1-sensitive [10, 16]. On the other hand, generic embeddings, such as the embedding from an arbitrary metric space into ℓ_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 ℓ_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 ℓ_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 halfspaces[3]. 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 ℓ_2 into ℓ_1 using random projections [10, 16], and a variant of Bourgain's theorem [4, 18], which allows the embedding of an *arbitrary* metric into ℓ_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 ℓ_1, for which the *information theoretic* problem of query release is simpler, since a d dimensional ℓ_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.

[3]Note that halfspace queries are in general high dimensional, but the large-margin assumption implies that the data has intrinsic dimension only roughly $O(\log n)$, since the dimensionality of the data can be reduced using the Johnson-Lindenstrauss lemma without affecting the value of any of the halfspace predicates.

2 Preliminaries

2.1 Model Let $(\mathcal{X}, d)$ be an arbitrary metric space. Let $\mathcal{D} \in \mathcal{X}^n$ 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 $y \in \mathcal{X}$ 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., $\sum_{x \in \mathcal{D}} \frac{1}{n} d(x, y)$. Let k denote the number of queries and $\mathcal{Q} \in \mathcal{X}^k$ be the set of distance queries asked by the data analyst. We will let $\mathcal{D}(\mathcal{Q}) \in \mathbb{R}^k$ denote the exact answer to the queries in $\mathcal{Q}$ with respect to database $\mathcal{D}$. 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

an answer for each query as it arrives. The answer can depend on the query, the private database, and the state of the mechanism, but not on future queries. An interactive query release mechanism is said to be efficient if the *per-query* running time is polynomial in the description size of the database. (E.g., if we consider an ℓ-dimensional ℓ_p metric, then it shall be polynomial in n and the dimension ℓ.)

The second setting is the *non-interactive setting*. A *non-interactive query release mechanism* takes the database as input and outputs an algorithm that can answer *all* queries without further access to the database. We say a non-interactive query release mechanism is efficient if both the running time of the mechanism and the running time per query of the algorithm it constructs are polynomial in n and ℓ.

2.2 Differential Privacy Let $\|\mathcal{D}_1 - \mathcal{D}_2\|_H$ denote the Hamming distance between two databases $\mathcal{D}_1$ and $\mathcal{D}_2$ (i.e., the number of entries that are different in $\mathcal{D}_1$ and $\mathcal{D}_2$). Two databases are adjacent if the Hamming distance between them is 1 (i.e. they differ in a single element). Let $n = |\mathcal{D}|$ be the size of the database. We will consider the by now standard privacy solution concept of "differential privacy" [6].

DEFINITION 1. *A mechanism M is* (ϵ, δ)-differentially private *if for all adjacent databases $\mathcal{D}_1$ and $\mathcal{D}_2$, any set of queries $\mathcal{Q}$, and for all subsets of possible answers $S \subset \mathbb{R}^k$, we have* $\Pr[M(\mathcal{D}_1, \mathcal{Q}) \in S] \leq \exp(\epsilon) \Pr[M(\mathcal{D}_2, \mathcal{Q}) \in S] + \delta$. *If $\delta = 0$, then we say that M is ϵ-differentially private.*

A function $f : \mathcal{X}^n \to \mathbb{R}$ is said to have sensitivity Δ with respect to the private database if $\max_{\mathcal{D}_1, \mathcal{D}_2} |f(\mathcal{D}_1) - f(\mathcal{D}_2)| \leq \Delta$, where the max is taken over all pairs of adjacent databases.

When we talk about the privacy of interactive mechanisms, the range of the mechanism is the entire transcript of queries and answers communicated between the data analyst and the mechanism (see [8, 13] for a more precise formalization of the model). An interactive mechanism is (ϵ, δ)-differential private if the probability that the transcript falls into any chosen subset differs by at most an $\exp(\epsilon)$ multiplicative factor and a δ additive factor for any two adjacent databases.

Given a mechanism, we will measure its accuracy in terms of answering distance queries:

DEFINITION 2. (ACCURACY) *A mechanism M is* (α, β)-accurate *if for any database $\mathcal{D}$ and any set of queries $\mathcal{Q}$, with probability at least $1 - \beta$, the M answers every query up to additive error α, i.e.,* $\Pr[\|M(\mathcal{D}, \mathcal{Q}) - \mathcal{D}(\mathcal{Q})\|_\infty \leq \alpha] \geq 1 - \beta$.

3 Releasing ℓ_1-Distance Queries

In this section, we consider ℓ_1 distance queries. Let $\mathcal{X} \subset [0,1]^\ell$ and $d = \|.\|_1$ such that the diameter of $\mathcal{X}$ (with respect to ℓ_1) is at most 1. We present private, computationally efficient mechanisms for releasing the answers to ℓ_1 distance queries in both the interactive and offline setting. These mechanisms for releasing ℓ_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* (ϵ, δ)*-differentially private mechanism for answering distance queries w.r.t.* $(\mathcal{X}, \|.\|_1)$ *that is* (α, β)*-accurate with*

$$\alpha = O\left(\frac{\ell^{9/5}\text{polylog}(\delta^{-1}, k, \beta^{-1})}{(n\epsilon)^{4/5}}\right) .$$

There is also an interactive ϵ-differentially private mechanism that is (α, β)-accurate for

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

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

THEOREM 3.2. *There is an non-interactive* (ϵ, δ)*-differentially private mechanism for answering distance queries w.r.t.* $(\mathcal{X}, \|.\|_1)$ *that is* (α, β)*-accurate with*

$$\alpha = O\left(\frac{\ell^{9/5}\text{polylog}(\delta^{-1}, n, \ell, \beta^{-1})}{(n\epsilon)^{4/5}}\right) .$$

There is also an non-interactive ϵ-differentially private mechanism that is (α, β)-accurate for

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

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

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 ℓ_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 ℓ_1 metric allows us to reduce the problem to learning ℓ 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 α. Finally, we combine these ingredients to give an interactive differentially private mechanism for releasing answers for ℓ_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 $O(\ell^2/\alpha)$ queries such that asking these queries to the interactive mechanism is sufficient to guarantee that the hypothesis function is accurate with respect to all queries. So Theorem 3.2 follows because the non-interactive mechanism can first ask these queries to the interactive mechanism and then release the hypothesis function.

Remark 2. *We note that once we reduce the problem to releasing* 1*-dimensional Lipschitz queries, there are other techniques that we could have used: for example, we could use the interval release algorithm of [2], or the private multiplicative weights mechanism of [13] after an appropriate discretization of the universe. However, these approaches would inherit inferior error bounds of* $O(1/\sqrt{n})$ *in comparison to the bound of* $O(1/n^{4/5})$ *that we achieve, in addition to inferior dependencies on the dimension* ℓ*. By exploiting convexity, we give improved bounds, which become more important later on when we reduce the general problem of metric query release to the problem of* ℓ_1 *distance query release: because the metric embeddings that we will use only blow up the final error bounds, it is important to get as little error as possible when solving the "base" problem.*

3.1 Query Release from Iterative Database Construction In this section, we introduce (a variant of) the *Iterative Database Construction (IDC)* framework in [12], which generalizes the median mechanism and the multiplicative weights mechanism [19, 13]. Roughly speaking, the IDC framework considers the database $\mathcal{D}$ as a mapping $F_{\mathcal{D}} : \mathcal{X} \mapsto \mathbb{R}$ from queries to their answers evaluated on $\mathcal{D}$, i.e., $\forall y \in \mathcal{X}, F_{\mathcal{D}}(y) = \mathcal{D}(y)$. It first learns an approximate version of this mapping privately by making a small number of queries to the mapping, and then answers all queries by the data analysts using the approximate mapping. Our variant of IDC allows the learning algorithm to also learn the subgradient of $F_{\mathcal{D}}$[4] (denoted as $\partial F_{\mathcal{D}}$) in addition to the value of $F_{\mathcal{D}}$ in the learning stage. Note that each coordinate of the subgradient of $F_{\mathcal{D}}$ is $O(1/n)$ sensitive in the private database $\mathcal{D}$. 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* ℓ_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* $\hat{F}_t : \mathcal{X} \to \mathbb{R}$ *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* $y_t \in \mathcal{X}$*, at which point the algorithm predicts a query value* $\hat{F}_t(y_t)$*. If* $|\hat{F}_t(y_t) - F_{\mathcal{D}}(y_t)| > \alpha$*, then we say the algorithm has* made a mistake*. At this point, the algorithm receives* $\ell + 1$ *values* $a_0, a_1, \ldots, a_\ell \in \mathbb{R}$ *s.t.*

$$a_0 \in [F_{\mathcal{D}}(y_t) - c\alpha, F_{\mathcal{D}}(y_t) + c\alpha]$$

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

$$a_i \in [\partial_i F_{\mathcal{D}}(y_t) - c\alpha, \partial_i F_{\mathcal{D}}(y_t) + c\alpha] \ .$$

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 $m : \mathbb{R}_+ \mapsto \mathbb{N}_+$*, if for any given error bound* α*, 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* ℓ_1 *distance queries with mistake bound* $m(\alpha)$ *and error tolerance* c*, then there is an* (ϵ, δ)*-differentially private mechanism in the interactive setting that is* (α, β)*-accurate, for* α *satisfying*

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

There is also an ϵ*-differentially private mechanism in the interactive setting that is* (α, β)*-accurate, for* α *satisfying*

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

[4] As we apply this framework to the ℓ_1 case, the subgradient of $F_{\mathcal{D}}$ is well-defined. When $F_{\mathcal{D}}$ 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 ℓ_1 Databases as Decomposable Convex Functions Consider a database $\mathcal{D}$ where the universe is the ℓ-dimensional unit cube $\mathcal{X} = [0,1]^\ell$ endowed with the ℓ_1 metric. Then the function mapping queries $y \in \mathcal{X}$ to their answers takes the form: $F_\mathcal{D}(y) = \frac{1}{n}\sum_{x\in\mathcal{D}} ||x-y||_1$, which is a 1-Lipschitz convex function of y.[5] We wish to proceed by providing an iterative database construction for ℓ_1 distance queries using these properties. Observe that because we are working with the ℓ_1 metric, we can write:

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

Observe that each function $F_\mathcal{D}^{(i)}(y)$ is 1-Lipschitz and convex, and has a 1-dimensional range $[0,1]$. Therefore, to learn an approximation to $F_\mathcal{D}(y)$ up to some error α, it suffices to learn an approximation to each $F_\mathcal{D}^{(i)}(y)$ to error α/ℓ. 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 α_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 α_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 α_1) in the neighborhood $[x_t^* - \alpha_1, x_t^* + \alpha_1]$, and hence the update points are at least α_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 $O(\frac{1}{\sqrt{\alpha_1}})$.

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 $\frac{3}{\sqrt{\alpha_1}}$ updates.*

Proof. Consider any two update points x_t^* and $x_{t'}^*$. Let us assume w.l.o.g. that $t < t'$. 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 α. Therefore, we get that

$$\begin{aligned} \alpha_1 &< G(x_{t'}^*) - (G'(x_t^*)(x_{t'}^* - x_t^*) + G(x_t^*)) \\ &= G(x_{t'}^*) - G(x_t^*) - G'(x_t^*)(x_{t'}^* - x_t^*) \\ &\le G'(x_{t'}^*)(x_{t'}^* - x_t^*) - G'(x_t^*)(x_{t'}^* - x_t^*) \\ &= (G'(x_{t'}^*) - G'(x_t^*))(x_{t'}^* - x_t^*) \ , \end{aligned} \tag{3.1}$$

where the second inequality is by the convexity of f.

Next, consider a maximal set of update points in sorted order: $-1 \le \hat{x}_1 < \cdots < \hat{x}_T \le 1$. Since G is convex and 1-Lipschitz, we have that $-1 \le G'(\hat{x}_1) < \cdots < G'(\hat{x}_T) \le 1$. Therefore, we get that

$$\begin{aligned} 2 \cdot 1 &\ge (G'(\hat{x}_T) - G'(\hat{x}_1))(\hat{x}_T - \hat{x}_1) \\ &= \textstyle\sum_{t=1}^{T-1}(G'(\hat{x}_{t+1}) - G'(\hat{x}_t)) \sum_{i=1}^{T-1}(\hat{x}_{t+1} - \hat{x}_t) \\ &\ge \left(\textstyle\sum_{t=1}^{T-1} \sqrt{(G'(\hat{x}_{t+1}) - G'(\hat{x}_t))(\hat{x}_{t+1} - \hat{x}_t)}\right)^2 \\ &\ge \left(\textstyle\sum_{t=1}^{T-1} \sqrt{\alpha_1}\right)^2 \ . \end{aligned}$$

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 $T \le \frac{\sqrt{2}}{\sqrt{\alpha_1}} + 1 < \frac{3}{\sqrt{\alpha_1}}$.

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.,

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

then a noisy version of the tangent line

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

where

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

[5] It follows from that for each $x \in \mathcal{D}$, $||x-y||_1$ is 1-Lipschitz and convex.

Learning a 1-Lipschitz and Convex Function

Maintain $\hat{G}(x) = \max_k\{a_k \cdot x + b_k\}$ 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 $G'(x_t^*)$.

and

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

will approximate the value at x up to additive error α_1. Hence, the mistake bound of the algorithm in Figure 1 for learning a 1-Lipschitz and convex function up to additive error α_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 ℓ-dimension decomposable convex function $F_{\mathcal{D}} = \sum_{i=1}^{\ell} F_{\mathcal{D}}^{(i)}$ up to additive error α, where each $F_{\mathcal{D}}^{(i)}$ 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 $F_{\mathcal{D}} : [0, 1]^{\ell} \to \mathbb{R}$ such that:*

1. *$F_{\mathcal{D}}(y) = \sum_{i=1}^{\ell} F_{\mathcal{D}}^{(i)}(y_i)$ where each $F_{\mathcal{D}}^{(i)} : [0, 1] \to [0, 1]$ is 1-Lipschitz and convex, and:*
2. *For every $y \in [0, 1]^{\ell}$ and every $i \in [\ell]$: $F_{\mathcal{D}}^{(i)}(y)$ and $(F_{\mathcal{D}}^{(i)}(y))'$ are $1/n$-sensitive in $\mathcal{D}$*

there is an iterative database construction algorithm for $F_{\mathcal{D}}$ using a collection of 2ℓ 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 $\hat{F}_t = \sum_{i=1}^{\ell} \hat{F}_t^{(i)}$ by maintaining ℓ one-dimension piecewise-linear hypothesis functions $\hat{F}_t^{(i)} : [0, 1] \mapsto [0, 1]$ for each $i \in [\ell]$ via the one-dimension learning algorithm with error tolerance α_1, and letting $\hat{F}_t = \sum_{i=1}^{\ell} \hat{F}_t^{(i)}$.
2. If the algorithm makes a mistake on query $y_t \in [0, 1]^{\ell}$, then the algorithm asks query y_t 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: $F_t^{(i)}(y_{ti})$ and $(F_t^{(i)})'(y_{ti})$, tolerating additive error up to $\alpha_1/(4)$, 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.

Note that whenever the above algorithm makes a mistake, at least one of the one-dimensional algorithms must also make a mistake (since otherwise the total error was at most $\ell\alpha_1 = \alpha$), and therefore we can charge this mistake to the mistake bound of at least one of the one-dimensional learning algorithms. By Lemma 3.3, the number of times that the hypothesis function $\hat{F}_t^{(i)}$ in each coordinate admits additive error at least α_1 is at most $O(1/\sqrt{\alpha_1})$. So the above iterative database construction algorithm has mistake bound $O(\ell/\sqrt{\alpha_1}) = O(\ell^{3/2}/\alpha^{1/2})$.

3.3 Proofs of Theorem 3.1 and Theorem 3.2

Proof. [Theorem 3.1] Since the ℓ_1 distance function in each coordinate has range $[0, 1]$ and is 1-Lipschitz, we get that $F_t^{(i)}$ and the derivative $(F_t^{(i)})'$ are $O(1/n)$-sensitive. So by Lemma 3.4, there is an iterative database construction algorithm for releasing answers to ℓ_1 distance queries that uses a set S of 2ℓ $O(1/n)$-sensitive queries with error α_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 (ϵ, δ)-differentially private mechanism in the interactive setting that is (α, β)-accurate for

releasing distance queries with respect to metric space $([0,1]^\ell, \|.\|_1)$, for α satisfying

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

Solving the above we get that

$$\alpha = O\left(\frac{\ell^{9/5}\log^{4/5}(4/\delta)\log^{4/5}(k/\beta)}{n^{4/5}\epsilon^{4/5}}\right) \ .$$

We also get that there is an ϵ-differentially private mechanism in the interactive setting that is (α,β)-accurate, for α satisfying

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

Solving the above we get that

$$\alpha = O\left(\frac{\ell^{7/3}\log^{2/3}(4/\delta)\log^{2/3}(k/\beta)}{n^{2/3}\epsilon^{2/3}}\right) \ .$$

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 ℓ/α' queries that we can make to each of the ℓ one-dimensional learning algorithms maintaining $\hat{F}^{(i)}_{\mathcal{D}}$ that guarantees that for each $y \in [0,1]$,

$$|\hat{F}^{(i)}_{\mathcal{D}}(y) - F^{(i)}_{\mathcal{D}}(y)| \le \alpha/\ell \ .$$

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

$$|\hat{F}_{\mathcal{D}}(y) - F_{\mathcal{D}}(y)| \le \alpha \ .$$

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 $y \in T$, $|\hat{F}^{(i)}_{\mathcal{D}}(y) - F^{(i)}_{\mathcal{D}}(y)| \le \alpha'/\ell$. Moreover, by the fact that $\hat{F}^{(i)}_{\mathcal{D}}$ is 1-Lipschitz, and for every $y \in [0,1]$, $d(y,T) \le \alpha'/\ell$, we have that for every $y \in [0,1]$,

$$|\hat{F}^{(i)}_{\mathcal{D}}(y) - F^{(i)}_{\mathcal{D}}(y)| \le 2\alpha'/\ell = \alpha/\ell \ ,$$

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 ℓ_1 distance queries via metric embeddings. An *embedding* from a metric space $(\mathcal{X}, d)$ to another metric space $(\mathcal{Y}, d')$ is a mapping $\pi : \mathcal{X} \times \mathcal{X}^* \mapsto \mathcal{Y}$, where $\pi(x, X)$ is the embedding of x w.r.t. a subset of points $X \subseteq \mathcal{X}$. 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 the 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 $\mathcal{D}$ and $\mathcal{Q}$ are the set of data points and the set of query points respectively.)

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

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

The contraction *of the embedding is*

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

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

In the rest of this section, we will always choose $X = \mathcal{D}$ and hence omit the second parameter of the embedding (i.e., $\pi(x) = \pi(x, \mathcal{D})$) when appropriate for brevity. We will let the target metric $(\mathcal{Y}, d')$ to be the ℓ_1 metric $([0,1]^\ell, \|.\|_1)$ and scale the embedding such that the expansion is 1.

4.1 1-Sensitive Metric Embeddings

Suppose we have an embedding from $(\mathcal{X}, d)$ to $([0,1]^\ell, \|.\|_1)$ with expansion 1 and contraction C. In some cases, the dimension ℓ of the target ℓ_1 space may depend on the contraction C. We will embed both the data points and the query points into $([0,1]^\ell, \|.\|_1)$ and release distance queries via the the mechanism for ℓ_1. Concretely, consider mechanisms $M_{\epsilon,\delta}$ and M_ϵ 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 ℓ_1 queries privately. So we have the following:

Releasing distance queries via embedding into ℓ_1

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

1. Construct a *proxy database* $\mathcal{D}'$ for releasing ℓ_1 distances by letting $\pi(x) \in \mathcal{D}'$ for all $x \in \mathcal{D}$.
2. For every $y \in \mathcal{Q}$, let the (ϵ, δ)-differentially private mechanism (resp., ϵ-differentially private mechanism) for ℓ_1 answer $\frac{1}{n}\sum_{x\in\mathcal{D}} \|\pi(x) - \pi(y)\|_1$, and release it as the answer to query y.

Figure 2: An (ϵ, δ)-differentially private mechanism $M_{\epsilon,\delta}$ (resp., ϵ-differentially private mechanism M_ϵ) for releasing distance queries via embedding into ℓ_1

THEOREM 4.1. *If π has expansion 1 and contraction C, and we use the (ϵ, δ)-differentially private mechanism for ℓ_1 in Figure 2, then with probability at least $1-\beta$,*

$$\forall y \in \mathcal{Q}: \quad \tfrac{1}{C}\sum_{x\in\mathcal{D}} d(x,y) - \tilde{O}\left(\tfrac{\ell^{9/5}}{n^{4/5}\epsilon^{4/5}}\right) \le M_{\epsilon,\delta}(\mathcal{D}, y) \le \sum_{x\in\mathcal{D}} d(x,y) + \tilde{O}\left(\tfrac{\ell^{9/5}}{n^{4/5}\epsilon^{4/5}}\right) ,$$

If we instead use the ϵ-differentially private mechanism for ℓ_1, then with probability at least $1-\beta$,

$$\forall y \in \mathcal{Q}: \quad \tfrac{1}{C}\sum_{x\in\mathcal{D}} d(x,y) - O\left(\tfrac{\ell^{7/3}}{n^{2/3}\epsilon^{2/3}}\right) \le M_{\epsilon}(\mathcal{D}, y) \le \sum_{x\in\mathcal{D}} d(x,y) + O\left(\tfrac{\ell^{7/3}}{n^{2/3}\epsilon^{2/3}}\right) ,$$

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 $\ell(\alpha)$-dimension ℓ_1 space, then we will choose the optimal additive error bound $\alpha_{\epsilon,\delta}$ and α_ϵ for (ϵ, δ)- and ϵ-differential privacy respectively such that*

$$\alpha_{\epsilon,\delta} = \tilde{O}\left(\tfrac{\ell(\alpha_{\epsilon,\delta})^{9/5}}{n^{4/5}\epsilon^{4/5}}\right) \quad \text{and} \quad \alpha_\epsilon = \tilde{O}\left(\tfrac{\ell(\alpha_\epsilon)^{7/3}}{n^{2/3}\epsilon^{2/3}}\right) .$$

Proof. Let us prove the error bound for ϵ-differential privacy. The proof of the error bound for (ϵ, δ)-differential privacy is similar. We will view the embedding π as from (X, d) to $(\pi(X), \|.\|_1)$. Since the embedding π has expansion 1, the image $\pi(X)$ of X has diameter 1 as well. Let $M_\epsilon^{\ell_1}$ denote the ϵ-differentially private mechanism for releasing answers to the ℓ_1 distance queries. Then we have that

$$\begin{aligned} M_\epsilon(y, D) &= M_\epsilon^{\ell_1}(\pi(y), D') \\ &\le \sum_{x\in D} \|\pi(x) - \pi(y)\|_1 + \tilde{O}\left(\tfrac{\ell^{7/3}}{n^{2/3}\epsilon^{2/3}}\right) \\ &\le \sum_{x\in D} d(x,y) + \tilde{O}\left(\tfrac{\ell^{7/3}}{n^{2/3}\epsilon^{2/3}}\right) . \end{aligned}$$

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 (ϵ, δ)-differentially private mechanism or an ϵ-differentially private mechanism for releasing answers to the ℓ_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 π from $(\mathcal{X}, d)$ to $([0,1]^\ell, \|.\|_1)$ 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 π is 1-sensitive, and if we use the (ϵ, δ)-differentially private mechanism (resp., ϵ-differentially private mechanism) for releasing answers to the ℓ_1 distance queries, then the mechanism $M_{\epsilon,\delta}$ (resp., M_ϵ) is (ϵ, δ)-differentially private (resp., ϵ-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 π is 1-sensitive. Since we are using an ϵ-differentially private mechanism for releasing ℓ_1 distances over the proxy databases, we get that for any set of queries $\mathcal{Q}$ and for any subset S of possible answers,

$$\begin{aligned} \Pr[M_\epsilon(\mathcal{D}_1, \mathcal{Q}) \in S] &= \Pr[M_\epsilon^{\ell_1}(\mathcal{D}_1', \pi(\mathcal{Q})) \in S] \\ &\le \exp(\epsilon)\Pr[M_\epsilon^{\ell_1}(\mathcal{D}_2', \pi(\mathcal{Q})) \in S] \\ &= \exp(\epsilon)\Pr[M_\epsilon(\mathcal{D}_2, \mathcal{Q}) \in S] . \end{aligned}$$

So mechanism M_ϵ is ϵ-differentially private. The proof for (ϵ, δ)-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 π 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 ℓ_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 π 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 ℓ_2 to ℓ_1.

LEMMA 4.1. (E.G., [10, 16]) *There is an embedding π from $([0,1]^{\ell}, \|.\|_2)$ to $([0,1]^{\ell'}, \|.\|_1)$ with expansion 1, contraction $1+\alpha$, and $\ell' = O\left(\frac{\ell \log(1/\alpha)}{\alpha^2}\right)$. Further, this embedding can be probabilistically constructed in polynomial time by defining each coordinate as a random projection.*

Since the above embedding is based on random projections, it is oblivious and hence 1-sensitive. So we can plug this embedding into our framework in Figure 2 and the following theorem follows from Theorem 4.1, Remark 3, Theorem 4.2, and Remark 5.

THEOREM 4.3. *Suppose $(\mathcal{X}, \|.\|_2)$ is a subspace of the ℓ_2 space with diameter at most 1. Then there are poly-time interactive and non-interactive (ϵ,δ)-differentially private mechanisms for answering ℓ_2 distance queries that are $(\alpha_{\epsilon,\delta},\beta)$-accurate for $\alpha_{\epsilon,\delta} = \tilde{O}\big(\frac{\ell^{9/23}}{n^{4/23}\epsilon^{4/23}}\big)$. There are also poly-time interactive and non-interactive ϵ-differentially private mechanisms for answering ℓ_2 distance queries that are (α_ϵ,β)-accurate for $\alpha_\epsilon = \tilde{O}\big(\frac{\ell^{7/17}}{n^{2/17}\epsilon^{2/17}}\big)$.*[6] [7]

4.3 Releasing Distances for General Metric via Bourgain's Theorem In this section, we will consider releasing distance queries with respect to an arbitrary metric $(\mathcal{X}, d)$ by embedding it into an ℓ_1 metric. Bourgain's theorem (e.g., [4, 18]) suggests that for any m points in the metric space, there is an embedding into an $O(\log^2 m)$-dimensional ℓ_1 space with distortion $O(\log m)$. 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 π be the embedding in Figure 3. For any data point $x \in \mathcal{X}$ and any query point $y \in \mathcal{Q}$, with probability at least $1 - \frac{1}{n^2 k}$, $\frac{1}{64 \log k} d(x,y) \le \|\pi(x) - \pi(y)\|_1 \le d(x,y)$.*

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 $x \in \mathcal{D}$ and $y \in \mathcal{Q}$ satisfies $d'(\pi(x),\pi(y)) \ge O(\frac{1}{\log k})(d(x,y) - d(x,\mathcal{Q}))$. We observe that the additive loss of $O(\frac{1}{\log k})d(x,\mathcal{Q})$ 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,

$$\begin{aligned} |\pi_{ij}(x) - \pi_{ij}(y)| &= \frac{1}{K \log k} |d(x, S_{ij}) - d(y, S_{ij})| \\ &\le \frac{1}{K \log k} d(x,y) \ . \end{aligned}$$

Summing over $0 \le i \le \log k$ and $1 \le j \le K$ we have

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

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 i' denote the largest index such that $r^*_{i'} + r^*_{i'-1} \le d(x,y)$. Redefine $r^*_{i'}$ to be $d(x,y) - r^*_{i'-1}$. We have $r^*_{i'} \ge \frac{d(x,y)}{2}$. We will need to following lemmas.

[6] The omitted poly-log factors depends on ℓ, n, and β^{-1} (and δ^{-1} for (ϵ,δ)-differential privacy) in the offline setting. In the interactive setting, this factor also depends on $\log k$. We remark again that in the *offline* setting, the constructed data structure can answer *all* ℓ_2 queries.

[7] We remark that Lemma 4.1 also holds for ℓ_p metrics for $p \in (1,2)$ (e.g., [10]). So the results stated in Theorem 4.3 also apply to ℓ_p metrices for $p \in (1,2)$. Details are omitted.

1-sensitive variant of Bourgain: Embedding an arbitrary metic space (X, d) into ℓ_1

Pre-processing: For $1 \le i \le \log k$ and $1 \le j \le K$, 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 $\{\pi_{ij}(x)\}_{0 \le i \le \log k, 1 \le j \le K}$ in the $O(K \log k)$-dimension ℓ_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 ℓ_1 space with $O(\log k)$ distortion

LEMMA 4.2. *For any $1 < i \le i'$, we have $\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \ge \frac{1}{32 \log k} \left(r_i^* - r_{i-1}^*\right)$ with probability at least $1 - \frac{1}{n^2 k \log k}$.*

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 $1 \le j \le K$, 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 $(1-2^{-(i-1)})^{2^{i-1}} \ge \frac{1}{4}$, while the probability that $S_{ij} \cap B(y, r_{i-1}^*) \ne \emptyset$ is at least $1 - (1 - 2^{-(i-1)})^{2^{i-2}} \ge 1 - e^{-\frac{1}{2}}$. In sum, with probability at least $\frac{1}{4}(1 - e^{-\frac{1}{2}}) > \frac{1}{16}$, we have both $S_{ij} \cap B^o(x, r_i^*) = \emptyset$ and $S_{ij} \cap B(y, r_{i-1}^*) \ne \emptyset$, which indicates that $d(x, S_{ij}) \ge r_i^*$ and $d(y, S_{ij}) \le r_{i-1}^*$ and therefore

$$|\pi_{ij}(x) - \pi_{ij}(y)| \ge \frac{1}{K \log k}(r_i^* - r_{i-1}^*) \ . \tag{4.2}$$

Further, by the additive form of Chernoff-Hoeffding theorem, we get that with probability at least $1 - 2^{-\frac{K}{64}} < 1 - \frac{1}{n^2 k \log k}$, (4.2) holds for at least $\frac{K}{32}$ i's. So we conclude that with probability at least $1 - \frac{1}{n^2 k \log k}$,

$$\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \ge \frac{1}{32 \log k} \left(r_i^* - r_{i-1}^*\right) \ .$$

LEMMA 4.3. $\sum_{j=1}^{K} |\pi_{1j}(x) - \pi_{1j}(y)| = \frac{1}{\log k} r_1^*$.

Proof. It is easy to see that $r_1' = 0$ because y itself is a query point and $r_1^* = r_1 = d(x, Q)$. Note that for every j, S_{1j} equals the set of query points. So we always have $d(x, S_{1j}) = d(x, Q) = r_1^*$ and $d(y, S_{1j}) = 0$. Therefore, $|\pi_{1j}(x) - \pi_{1j}(y)| = \frac{1}{K \log k} r_1^*$ for $1 \le j \le K$, and summing up completes the proof.

By Lemma 4.2 and union bound, with probability at least $1 - \frac{1}{n^2 k}$, we have

$$\sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \ge \frac{1}{32 \log k} \left(r_i^* - r_{i-1}^*\right)$$

for all $1 < i \le i'$. By Lemma 4.3 we have

$$\sum_{j=1}^{K} |\pi_{1j}(x) - \pi_{1j}(y)| \ge \frac{1}{\log k} r_1^* \ .$$

Summing them up we get that

$$\sum_{i=1}^{\log k} \sum_{j=1}^{K} |\pi_{ij}(x) - \pi_{ij}(y)| \ge \frac{1}{32 \log k} r_{i'}^* \ge \frac{1}{64 \log k} d(x, y) \ .$$

As a corollary of Theorem 4.4 and union bound we have

COROLLARY 4.1. *Let π be the embedding in Figure 3. Then with probability at least $1 - \frac{1}{n}$, we have that for any data point $x \in \mathcal{D}$ and any query points $y \in \mathcal{Q}$, $\frac{1}{64 \log k} d(x, y) \le \|\pi(x) - \pi(y)\|_1 \le d(x, y)$.*

Hence, there exists an embedding of an arbitrary metric to an ℓ_1 metric with distortion $O(\log k)$ that is 1-sensitive because it is oblivious to the data points. The dimension of the resulting ℓ_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 (ϵ, δ)-privacy guarantee, but it does not: when computing ℓ_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 an non-interactive (ϵ, δ)-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 $y \in \mathcal{Q}$,*

$$O\left(\frac{1}{\log k}\right) \frac{1}{n} \sum_{x \in \mathcal{D}} d(x, y) - \tilde{O}\left(\frac{1}{n^{4/5} \epsilon^{4/5}}\right) \le M_{\epsilon,\delta}(y)$$

$$\le \frac{1}{n} \sum_{x \in \mathcal{D}} d(x, y) + \tilde{O}\left(\frac{1}{n^{4/5} \epsilon^{4/5}}\right) \ .$$

There is also an non-interactive ϵ-differentially private mechanism M_ϵ, such that with high probability, for any query $y \in \mathcal{Q}$,

$$O\left(\frac{1}{\log k}\right)\sum_{x\in\mathcal{D}} d(x,y) - \tilde{O}\left(\frac{1}{n^{2/3}\epsilon^{2/3}}\right) \leq M_\epsilon(y)$$

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

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

References

[1] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138. ACM, 2005.

[2] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 609–618. ACM, 2008.

[3] Avrim Blum and Aaron Roth. Fast private data release algorithms for sparse queries. *arXiv preprint arXiv:1111.6842*, 2011.

[4] Jean Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, 1985.

[5] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210. ACM, 2003.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, pages 265–284. Springer, 2006.

[7] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 381–390. ACM, 2009.

[8] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[9] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 361–370. ACM, 2009.

[10] Tadeusz Figiel, Joram Lindenstrauss, and Vitali D. Milman. The dimension of almost spherical sections of convex bodies. *Acta Mathematica*, 139(1):53–94, 1977.

[11] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. In *Proceedings of the 43rd annual ACM Symposium on Theory of Computing*, pages 803–812. ACM, 2011.

[12] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *Proceedings of the 9th Conference on Theory of Cryptography*, pages 339–356. Springer, 2012.

[13] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proceedings of the 51st IEEE Annual Symposium on Foundations of Computer Science*, pages 61–70. IEEE, 2010.

[14] Moritz Hardt, Guy N. Rothblum, and Rocco A. Servedio. Private data release via learning thresholds. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 168–187. SIAM, 2012.

[15] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 10–33. IEEE, 2001.

[16] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

[17] Prateek Jain and Abhradeep Thakurta. Mirror descent based database privacy. *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 579–590, 2012.

[18] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[19] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 765–774. ACM, 2010.

[20] Justin Thaler, Jonathan Ullman, and Salil Vadhan. Faster algorithms for privately releasing marginals. *Automata, Languages, and Programming*, pages 810–821, 2012.

[21] Jonathan Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 361–370. ACM, 2013.