A Differentially Private Approximation of the Width Problem

Mor Hale
Bar-Ilan University, Ramat Gan, Israel

Or Sheffet
Bar-Ilan University, Ramat Gan, Israel

Abstract

The width of a point set – the minimum distance between two parallel hyperplanes enclosing the data – is a fundamental geometric measure that captures how "flat" or "fat" a dataset is. As such, it serves as a basic shape descriptor used in visualization, convex hull approximation, and geometric data analysis. Despite its importance, width is highly sensitive to single-point changes, and no differentially private algorithm for approximating it was previously known.

We present the first pure ε-differentially private algorithm that approximates the width of a dataset. Our algorithm is a private adaptation of Chan's approximation scheme [8] and operates by privately approximating the solution to a collection of suitably formulated linear programs. In addition to estimating the width, our method privately identifies a corresponding direction, enabling a private "fattening" transformation of the dataset – a basic structural preprocessing step for many geometric algorithms. This work advances the understanding of how geometric shape descriptors can admit good approximations even under the constraints of differential privacy.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Differential privacy, computational geometry, width approximation, private algorithms

Digital Object Identifier 10.4230/LIPIcs.FORC.2026.18

1 Introduction

This work is motivated by the problem of data-visualization, and is aimed at finding the data's "geometric shape" using differentially private techniques.

A key geometric measure we focus on is the *directional width*, which captures how spread out a point set is along a given direction. Formally, for a unit vector u, the directional width of a point set $P \subset \mathbb{R}^d$ in direction u is

$$w_u(P) = \max_{p,q \in P} \{(p - q) \cdot u\}$$

The collection of directional widths over all directions characterizes the global geometric shape of the data and its convex hull. For any fixed direction u, computing the maximal and minimal projection reduces to a one-dimensional range query, a problem that has been extensively studied in differential privacy [5, 9, 15]. However, extending such guarantees simultaneously to all directions or finding representing u-directions for the data is substantially more challenging.

This issue is best demonstrated by the two extremal directional widths: the *diameter*, which is the maximum directional width over all directions; and the *width* of the set, defined as the *minimum* directional width over all directions. We denote that data's width as $w^*(P)$. The width captures how "flat" or "fat" a dataset is. When the width is comparable to the diameter, the dataset is well-rounded or *fat*. In contrast, when the width is significantly smaller than the diameter, the dataset is nearly degenerate and concentrated around a low-dimensional structure. This distinction has significant algorithmic consequences. Many

geometric approximation schemes – such as those for convex hull simplification, geometric packing, coresets, extent measures, bounding volumes, and LP-type problems – become substantially simpler and more efficient when the input is fat [14, 7, 10]. Conversely, when the width is very small, these problems often require more complex constructions to obtain comparable guarantees.

The main contribution of this work is thus a (pure) ε-DP algorithm that outputs an approximation $\tilde{w}$ to the data's width as well as a direction u on which it is obtained. Once those are found, it is fairly straight-forward to identify an interval in the direction of u of length $\tilde{w}$ that holds most of the data, and then linearly stretch this interval so that the spread of the data along direction u will be roughly as large as the diameter. So by repeating this process at most $d-1$ times we are able to create a "fat" dataset.

Beyond its structural role, the width problem also has direct geometric implications. For example, a ball of radius $\frac{w^*}{2}$ is guaranteed to fit inside the convex hull of the data, a property that is useful in algorithms that seek a point in the convex hull [19, 17]. Thus, width provides both structural and algorithmic insight into the dataset.

Non private width algorithms

In the non-private setting, the width problem on a dataset of size n can be solved exactly in two dimensions in time $O(n \log n)$ [25] and in three dimensions in time $O(n^{3/2+\delta})$ [1]. In higher dimensions, an algorithm running in $O(n^{\lceil d/2 \rceil})$ time is noted in [8]. More importantly, there are known $O(n \cdot \exp(d))$-time approximation schemes for the width problem developed by Duncan, Goodrich, and Ramos [11] and by Chan [8] in his seminal work. The latter forms the basis of our work.

The Setting: Low-dimensional data with an exponentially large grid

It is evident that the algorithms that approximate the width run in $\exp(d)$-time, and so in this work we assume that the data is low-dimensional. Indeed, there exists datasets that hold sensitive information and are intrinsically low-dimensional, such as 2D-locations, trajectories or medical records composed of only a few features. And so we assume the given input P is composed on n points in the d-dimensional cube $[-1,1]^d$. However, as proven by Bun et al. [5], no DP-algorithm is feasible if the data's range is infinite, so we are also provided with a finite grid $\mathcal{G} \subset [-1,1]^d$ that the data resides on. We assume this grid has granularity τ, yet this granularity is *exponentially* small. (As a rule of thumb, think of $\tau = 2^{-32}$, as if each coordinate in the data is given in `C`'s `int`s.) And so our algorithms must avoid brute-search over all points in $\mathcal{G}$.

Our contribution

It is fairly obvious to see that adding or removing a single datum to a data can change its width by an arbitrarily large amount, rendering the DP-techniques that rely on the global sensitivity of the width unapplicable. However, note that for any fixed pair of parallel hyperplanes the number of points between them changes by at most one across neighboring datasets. Thus, while the optimal width is highly unstable, the score function that counts how many points lie within a given slab has sensitivity one.

Building on this observation, we present a differentially private version of Chan's approximation algorithm [8]. Chan reduces the width problem to solving a collection of linear programs (LPs), one for each direction in a (finite) cover of the unit sphere. We give a detailed explanation of Chan's non-private algorithm in Section 2.2.1, where we formalize these LPs.

However, roughly speaking, these LPs seek a vector x (of largest possible magnitude) where by taking the dot-products of all points in P with x, these dot-products fall inside an interval of length 1.

And so in this work we introduce the notion of a *score* of a possible LP solution x, defined as the maximal number of points from P whose dot-product with x reside inside an interval of length 1. (See Section 3 for the formal definition.) Similarly, the score of a *region* (a closed rectangle in $\mathbb{R}^d$) is the maximal score of all xs in the region. These definitions enables us to use simple SVT-based algorithm to conduct our private width approximation, seeing as the sensitivity of the score is at most 1.

This approach yields the first pure ε-differentially private algorithm for approximating width. Given a privacy parameter $\varepsilon > 0$, an approximation parameter $\gamma > 0$ and a failure parameter $\beta > 0$, our algorithm achieves a (γ, Δ)-*bi-criteria* approximation: with probability at least $1 - \beta$ the returned width $\tilde{w}$ satisfies: $\tilde{w} \leq (1 + \gamma)w^*(P)$, where $w^*(P)$ denotes the minimum width of the dataset. Moreover, there exists a pair of parallel hyperplanes at distance $\tilde{w}$ that sandwich all but at most Δ points of P.

Formally, our main result is as follows.

▶ **Theorem 1.** *Fix $d \in \mathbb{N}$, $\varepsilon > 0$, and $\gamma > 0$. Let $\tau > 0$ and let $\mathcal{G}$ be the grid in $[0,1]^d$ of granularity τ. There exists an ε-DP algorithm that, given a set $P \subset \mathcal{G}$ of size n, (γ, Δ)-approximates the width problem, where*

$$\Delta = O\left(\frac{d}{\varepsilon}\log\left(\frac{\log(1/\tau)}{\gamma\beta}\right)\right).$$

The algorithm runs in time

$$O\left(\frac{d\log(1/\tau)}{\gamma}\cdot\gamma^{-\frac{d-1}{2}}\cdot(n^d\omega_{d-1} + n^{d+1}\log n)\right),$$

where ω_d denotes the time required to solve d linear equations in d variables.[1]

In addition, we give an ε-DP algorithm that outputs both an approximate width $\tilde{w}$ and two parallel hyperplanes at distance $\tilde{w}$ that enclose at least $n - \Delta'$ points, where

$$\Delta' = O\left(\frac{d}{\varepsilon}\log(\frac{1}{\gamma\beta}) + \frac{d^4}{\varepsilon}\log^2(\frac{d}{\tau})\log(\frac{\log(d/\tau)}{\beta})\right).$$

Organization. In the remainder of the introduction we survey the few results that do exist in the intersection of DP and computational geometry. We then survey background material in Section 2, detailing also Chan's non-private approximation algorithm. In Section 3 we give our ε-DP algorithms for approximating the width, and detail the main definitions of a *score* of a point and of a region. The main technical hurdle, which is how to find the score of a region without brute-force iterating over all grid-points in this region, is given in Sections 4 and 5. First, in Section 4 we compute it for 2D data (which is arguably the only case where iterating over a cover is practical); and in Section 5 we show how to compute it in d-dimensions. Finally we conclude with open problems in Section 6.

[1] Currently, the best known bound is $\omega_d = d^{2.38\ldots}$.

Related Work. We note that no prior work focuses specifically on the width problem and its DP-approximation. However, the problem of privately approximating geometric structures has been studied from several angles. In [16] the authors aim to learn a shape from the data, assuming there are many points inside and outside the shape. The works of Beimel, Moran, Nissim and Stemmer [2] and of Kaplan, Sharir and Stemmer [19] provide differentially private algorithms for finding a point that lies inside the convex hull of a dataset, using the notion of Tukey-depth [26]. Building on this notion of Tukey-depth, the work of Gao and Sheffet [13] investigates how to privately approximate the convex hull and the structure of Tukey-regions. Among the algorithms proposed by Gao and Sheffet is an algorithm that approximates the width of a Tukey-region, but only under the premise that the width is proportional to the diameter (as they simply approximate the directional width for each direction is a cover of the unit-sphere). Several works [23, 22] look at the 1-cluster problem under differential privacy, which can be applied to approximate the data's diameter (even in a high-dimensional setting).

Chan's approximation method focuses on solving multiple linear programming problems. Our algorithm addresses how to solve these problems privately and efficiently. A related work is [17], which provides a DP-algorithm for solving linear problems, but whose efficiency depends on the "roundness" parameter, defined as the radius of the smallest ball that fits inside the LP polytope. This quantity is related to the width, since the roundness cannot be smaller than the minimal width of the point set. (See more discussion in Section 2.3.)

2 Preliminaries

2.1 Differential Privacy

A dataset $P \subset \mathbb{R}^d$ is a multi-set of points in $\mathbb{R}^d$. Two datasets P and P' are called *neighbors* if they differ on a single point. An algorithm $\mathcal{M}$ is said to be ε-differentially private (ε-DP) if for any two neighboring P and P' and any measurable set S of outputs it holds that

$$\Pr[\mathcal{M}(P) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(P') \in S]$$

One of the basic mechanisms that preserves ε-DP is the Laplace mechanism. First, given a query q from datasets to the reals $\mathbb{R}$, we say that the query is 1-*sensitive* if $\forall P, P'$ neighbors, $|q(P) - q(P')| \leq 1$. Given a 1-sensitive query, it is known that the mechanism that on input P outputs $q(P) + \mathrm{Lap}(\frac{1}{\varepsilon})$ is ε-DP, where $\mathrm{Lap}(\lambda)$ denotes a random variable drawn from the Laplace distribution with $\mathrm{PDF}(x) \propto e^{-|x|/\lambda}$. Another basic property of DP is the *basic composition* of two DP algorithms, where for any $\varepsilon_1, \varepsilon_2 > 0$ the joint output of a ε_1-DP mechanism and of a ε_2-DP mechanism is $(\varepsilon_1 + \varepsilon_2)$-DP.

Sparse Vector Technique (SVT)

One of the tools we use repeatedly in this work is the Sparse Vector Technique (SVT) (also known as the Above-Threshold algorithm in [12]). In this setting we are given a sequence of k queries, $q_1, q_2, ..., q_k$ where all queries are 1-sensitive, and our goal is to identify the *first* query whose value exceeds a publicly known threshold T.

▶ **Fact 2.** *The SVT is a ε-DP algorithm which satisfies the following. Set $\Delta = \frac{6}{\varepsilon}\log(\frac{k+1}{t})$. Then, if SVT halts on query q_{i^*} then w.p. $\geq 1-\beta$ we have that $q_{i^*}(P) \geq T - \Delta$, and for every $i < i^*$ it holds that $q_i(P) \leq T + \Delta$.*

2.2 Width of a Set

The *width* of a set of points $P \subset \mathbb{R}^d$ is defined as the smallest distance between two parallel hyperplanes that enclose all points in P. Formally, if we denote the set of points by P and given a unit vector $u \in \mathbb{R}^d$, the directional width of P along direction u, denoted $w_u(P) = \max_{p,q \in P}\{(p-q)\cdot u\}$. The width of dataset P is thus $w^*(P) = \min_{\|u\|=1} w_u(P)$.

2.2.1 Chan's Approximation Algorithm of the Width

In [8], Chan reformulates an algorithm originally presented by Duncan et al. [11] to approximate the width of a set.

A rather natural "first attempt" at obtaining a $(1+\gamma)$-approximation of the width would work as follows. Discretize the unit sphere into a $O(\gamma)$-cover, and find the minimum width among all directions in the cover. However this approach guarantees only an *additive* $O(\gamma \cdot \mathrm{diam(P)})$ approximation to the width, which may be very large for datasets with diameter $\gg$ width. In other words, in order to obtain a multiplicative guarantee for the width approximation problem, this approach requires a very fine discretization of the unit-sphere (proportional to the data's width). This, in turn, renders the naïve approach of applying the Exponential Mechanism [21] over a cover of the unit-sphere infeasible. Chan's approach avoids this issue by adapting to the geometry of the data via linear programs, one per each direction in the cover.

We now reiterate Chan's algorithm for clarity. Recall that the width problem is: $w^*(P) = \min_{\|x\|=1}\{\max_{q\in P} x\cdot q - \min_{q\in P} x\cdot q\}$. We can write the problem as:

$$w^*(P) = \min_{0\neq x\in\mathbb{R}^d}\left[\frac{z_2}{\|x\|} - \frac{z_1}{\|x\|}\right] \qquad \text{s.t.} \qquad z_2 = \max_{q\in P} x\cdot q, \qquad z_1 = \min_{q\in P} x\cdot q.$$

Equivalently, we can write:

$$w^*(P) = \min_{0\neq x\in\mathbb{R}^d}\left[\frac{z_2 - z_1}{\|x\|}\right]$$
$$\text{s.t.} \quad \forall q\in P: \quad z_1 \le x\cdot q \le z_2.$$

Note that in the latter formulation, for any $x \neq 0$ we have that both x and λx for any scalar λ lead to the same minimal value. So we restrict the solution by requiring that $z_2 - z_1 = 1$. This constraint fixes the norm of x. Thus, we obtain the following problem:

$$w^*(P) = \min_{x\in\mathbb{R}^d}\frac{1}{\|x\|}$$
$$\text{s.t.} \quad \forall q\in P: \quad z_1 \le x\cdot q \le z_1 + 1.$$

which is equivalent to the problem

$$\max_{x\in\mathbb{R}^d}\|x\|$$
$$\text{s.t.} \quad \forall q\in P: \quad z \le x\cdot q \le z+1. \tag{1}$$

In this formulation of the problem all constraints are linear, yet the objective isn't a concave function and so we cannot obtain its maximization easily. Instead, Chan suggestes using a cover V_θ of all directions. Let V_θ be a set of unit-length vectors with the property that for any unit-length u there exists some $a \in V_\theta$ such that $\measuredangle(u,a) \le \theta$. It is known that a cover of

size $O((\frac{1}{\theta})^{d-1})$ exists. It now follows that the optimal solution x to the above-mentioned optimization problem becomes "almost aligned" with some grid direction a – namely, we set $\theta \leq \sqrt{\gamma}$, and have that for some $a \in V_\theta$ it holds that:

$$\|x\| \geq a \cdot x = \|x\| \cdot \cos(\measuredangle(x,a)) \geq \|x\| \cos\theta \geq \|x\|(1 - \frac{\theta^2}{2}) \geq \|x\| \cdot (1 - \frac{\gamma}{2}) \overset{\gamma \leq 1}{\geq} \frac{\|x\|}{1+\gamma}$$

And so, Chan suggestes we solve all $|V_\theta|$ LPs of the form:

$$\begin{aligned} \text{LP}(a): \quad & \max_{x \in \mathbb{R}^d} a \cdot x \\ \text{s.t.} \quad & \forall q \in P: \quad z \leq x \cdot q \leq z + 1. \end{aligned} \tag{2}$$

and take the maximum over all solutions, and this guarantees a solution whose width is $w \leq w^*(P)(1+\gamma)$.

Min. Possible Width. In our setting, all points in P are known to reside on a predefined grid $\mathcal{G}$ of granularity τ. We wonder as to the smallest possible value of $w^*(P)$. clearly, if all points lie on some strict subspace of $\mathbb{R}^d$ then their width is 0. So, we pose the question: what is the smallest non-zero value $w^*(P)$ can take assuming the data isn't degenerate?

We answer this question using Chan's formulation. Based on (2), we know that the solution to LP(a) is attained at a vertex of its feasible polytope. Such a vertex is determined by a system of $(d+1)$ linear equations in $(d+1)$ variables, where each non-zero coefficient in the system lies between τ and 1. By Cramer's rule, each coordinate of the solution is given by a ratio of two determinants. By Hadamard's inequality, the numerator is at most $(\sqrt{d+1})^{d+1}$ (since the norm of each row is bounded by $\sqrt{d+1}$), while the denominator is at least τ^{d+1}. Hence each coordinate of x is bounded by $X = \left(\frac{\sqrt{d+1}}{\tau}\right)^{d+1}$.

It follows that the LP objective value is at most $d\left(\frac{\sqrt{d+1}}{\tau}\right)^{d+1}$. Since this objective only approximates the inverse width, we incur an additional constant factor of 2 when translating the bound to the true objective. Inverting this bound, the smallest possible non-zero width satisfies $w^*(P) \geq \frac{\tau^{d+1}}{2d(d+1)^{(d+1)/2}} \geq \left(\frac{\tau}{d+1}\right)^{d+1}$.

The analysis on the min-possible width is required for our ε-DP algorithm for two reasons. First, it is required as part of the SVT-analysis, whose utility bound depends logarithmically on the number of queries posed – and we pose one for every (approximated) value of the LP value. Second, this analysis plays an even more important role in Section 3.1, where we give the algorithm that finds an approximated LP solution (and not merely the LP-value). The algorithm merely conducts DP-binary search over the *grid of possible solutions* to the LP, and this grid needs to be of granularity $\frac{1}{X}$. As a result, in the processes of conducting said binary search, we may lose additional $\text{poly}\log(\frac{1}{X})/\varepsilon$ many points. Needless to say, if we replace this worst-case grid with a grid of larger granularity, then the cost of the latter part can significantly improve.

2.3 Baseline

An off-the-shelf DP-algorithm for approximating the width would therefore be to run a DP-LP solver for the problems $\{\text{LP}(a)\}_{a \in V_\theta}$, and apply private selection algorithms [20, 24] to choose a good solution. The DP-LP solver of Kaplan et al. [18] runs in exponential time, as it requires convexifying the set of points that satisfy precisely k constraints of the LP for any value of k. The current best DP-LP solver is given in a work of Kaplan et al. [17] where they

bound the number of violated LP constraints at $O(\frac{d^5 \text{polylog}(\frac{1}{\tau\rho})}{\varepsilon})$ where τ is the granularity of the given grid and ρ is a new parameter known as the width of the LP, which we don't know how to apriori control.[2] In contrast, our work yields the bound $\Delta = O(\frac{d}{\varepsilon}\log(\frac{\log(1/\tau)}{\beta}))$, so we enjoy both a better dependency on d and a double-log dependency on $\frac{1}{\tau}$. This unfortunately comes at the expense of the runtime – our algorithms run in time $O(n^{d+1})$ whereas the runtime of the LP solver of [17] runs in time fully polynomial in d. But recall that Chan's approximation algorithm solves $O(|V_\theta|)$ many LPs anyhow, so a runtime with exponential dependency on d is unavoidable (to the best of our knowledge).

3 Private Width Approximation Algorithms

We begin with a high-level description of the full private width-approximation pipeline. At a conceptual level, the algorithm proceeds in two stages. First, it privately estimates the width value by searching over candidate values and checking, for each one, whether there exists a direction and an associated LP solution that places almost all projected points inside an interval of length 1. Second, once such a width value is found, the algorithm privately recovers a corresponding direction and LP solution, from which one can also derive a pair of enclosing hyperplanes. The detailed subroutines used in these stages are given later in this section and in the following sections. The high-level procedure:

- Run Algorithm 1 on P to obtain an approximate width $\tilde{w}$.
- Run Algorithm 3 on input $(P, \tilde{w})$ to obtain a vector $\tilde{x}$ witnessing this approximate width.
- Project the points of P onto $\tilde{x}$ and privately identify an interval of length 1 containing many projected points.
- Convert this interval into a pair of parallel hyperplanes orthogonal to $\tilde{x}$, at distance $\frac{1}{\|\tilde{x}\|}$.

We now describe the first stage of the algorithm, namely the private estimation of the width value. As outlined above, this stage reduces to searching over candidate width values and identifying the largest value for which there exists a direction and an associated LP solution that captures almost all points. At its core, the algorithm is a simple SVT: we are conducting a search for the optimal value of the width w^*, where we start with a guess of 1 and reduce it from there all the way down to the lowest possible value of $(\frac{\tau}{d+1})^{d+1}$. Thus, the number of guesses of w is upper bounded by $W = \lceil \log_{\frac{1}{1+\gamma}}((\frac{d+1}{\tau})^{d+1})\rceil = (d+1)\cdot O(\frac{\log(d/\tau)}{\gamma})$. We denote our guessed values as $\{w_0 = 1, w_1 = \frac{1}{1+\gamma}, ..., w_W\}$.

For each guess w_i, our SVT asks if there exists at least one LP(a) (as in Equation (2)) for which there exists some x that satisfies $x \cdot a = \frac{1}{w_i}$ and when projecting points onto x, then many points fit inside an interval of length 1. More formally, we are looking for some x of the form $x = \frac{1}{w_i}a + y$ for some y perpendicular to a, and ask for the maximal number of points whose projection onto x fits in an interval of length 1.

▶ **Definition 3** (Score at a Point). *Let $P \subset \mathcal{G}$ be a set of points. Fix a scalar $w > 0$ and a direction $a \in V_\theta$. Given a point y perpendicular to a we denote the* score *of y with respect to w and a, denoted $\text{sc}^P_{w,a}(y)$, as the maximum number of points in P whose dot-product with $x = \frac{1}{w}a + y$ lie inside an interval of length 1.*

Formally, given P, w, a and y, we denote $x = \frac{1}{w}a + y$. For any $z \in \mathbb{R}$ consider the interval $[z, z+1]$ and denote $S_z \subset P$ as $S_z = \{p \in P : p \cdot x \in [z, z+1]\}$. Then $\text{sc}^P_{w_i,a}(y) = \max_{z\in\mathbb{R}} |S_z|$. (We omit the superscript P from the score function when the context is clear.)

[2] Kaplan et al. bound it at $O(\frac{1}{d\cdot(d/\tau)^d})$.

In the 2-dimensional setting, we often fix the orthogonal direction to a, $a^\perp$, and identify the point y with the scalar μ such that $y = \mu a^\perp$, namely define $\mathrm{sc}_{w,a}(\mu) = \mathrm{sc}_{w,a}(y)$. We use a similar brevity in d-dimensions as well. Given an orthonormal basis $\{u_1, u_2, ..., u_{d-1}\}$ to the subspace perpendicular to a, we denote the score of $(\lambda_1, \lambda_2, ..., \lambda_{d-1})$ as the score of the point $y = \sum_j \lambda_j u_j$, i.e. $\mathrm{sc}^P_{w,a}(\lambda_1, \lambda_2, ..., \lambda_{d-1}) = \mathrm{sc}^P_{w,a}(\sum_j \lambda_j u_j)$.

Efficiently computing the score of a point

We argue that the score of a given point y, $\mathrm{sc}_{w,a}(y)$, can be efficiently computed in time $O(nd + n\log(n))$ (for any dimension d). Indeed, all we need to do is the following.

1. Compute the dot-product of each $p \in P$ with $x = \frac{1}{w}a + y$.
2. Sort the resulting dot-products to obtain $\{c_1, c_2, ..., c_n\}$.
3. Iterate over the n intervals $[c_i, c_i + 1]$ to check how many points fall inside this interval.
4. Return the maximal number of points found in Step 3.

This takes $O(nd + n\log(n) + n)$-time (Step 3 can be done in $O(1)$-time per interval by simply holding two indices i, j where index i points to the first dot-product $\geq c_i$ and j points to the first dot-product $> (c_i + 1)$).

Score at a region

The main hurdle is that we do not know which point y in the subspace orthogonal to a yields the highest possible score or how to find such a y. And this is the main technical contribution of this paper: we show that in order to find a y of large score, there are $O(n^d)$ points of interest in $\mathbb{R}^{d-1}$ whose score we need to check and y will be among them. But we table the discussion on runtime analysis to Sections 4 and 5, and continue with the formal definitions.

Again, we first consider the 2-dimensional case. Our goal is to find $y = \mu a^\perp$, or μ, of large score. So we thus define the score of any closed interval $I \subset \mathbb{R}$ to be the maximal score of any $\mu \in I$. The same idea is generalized to any d in the following definition.

▶ **Definition 4** (Score at a region). *Let $I_1, I_2, ...I_{d-1} \subset \mathbb{R}$ be $d-1$ closed intervals in $\mathbb{R}$ (including perhaps the case that $I_j = \mathbb{R}$ for some j), and let $R = I_1 \times I_2 \times ... \times I_{d-1}$ be a closed rectangle in $\mathbb{R}^{d-1}$. Let $P \subset \mathcal{G}$ be a set of points. Fix a scalar $w > 0$ and a direction $a \in V_\theta$. Fix $\{u_1, u_2, ..., u_{d-1}\}$ to be some orthonormal basis of the subspace perpendicular to a in $\mathbb{R}^d$. Then the* score of the region R *is defined as*

$$\mathrm{sc}^P_{w,a}(R) = \max_{(\lambda_1,\lambda_2,...,\lambda_{d-1})\in R} \mathrm{sc}^P_{w,a}(\lambda_1, \lambda_2, ..., \lambda_{d-1})$$

The rest of the definitions we give are simply notations for maximal scores. We now define the score of a direction a as $sc^P_w(a) = \mathrm{sc}^P_{w,a}(\mathbb{R}^{d-1})$, and the score of a guess w as

$$\mathrm{sc}^P(w) = \max_{a\in V_\theta} \left\{sc^P_w(a)\right\} = \max_{a\in V_\theta} \left\{\mathrm{sc}^P_{w,a}(\mathbb{R}^{d-1})\right\} \tag{3}$$

(Again, we omit the superscript P from all score-functions when the context is clear.)

It should be clear that for any two neighboring P and P' and any w, a and y, we have

$$\left|\mathrm{sc}^P_{w,a}(y) - sc^{P'}_{w,a}(y)\right| \leq 1$$

or, alternatively, that the sensitivity of the score at a point is at most 1. It follows that for any $(d-1)$-dimensional rectangle R the sensitivity of the query $\mathrm{sc}_{w,a}(R)$ is also 1, as a maximum over queries of sensitivity 1. (See [4] for a proof of this fact.) Similarly, we also have that the query $\mathrm{sc}_w(a)$ is also of sensitivity 1, and so does $\mathrm{sc}(w)$. And so our algorithm

simply uses SVT for the sequence of W queries, where $q_i(P) = \mathrm{sc}(w_i)$, and halts on the first guess w_{i^*} whose score is too low, and returns the next-to-last query. The details appear in Algorithm 1.

Algorithm 1 Differentially Private Width Estimation.

Input: Dataset $P \subset \mathcal{G}$ of size n where $\mathcal{G}$ is a d-dimensional grid of granularity τ, privacy budget $\varepsilon > 0$, failure parameter $\beta > 0$, approximation parameter $\gamma > 0$.
Set $W \leftarrow \lceil \log_{\frac{1}{1+\gamma}}((\frac{d+1}{\tau})^{d+1}) \rceil$ and set $w \leftarrow 1$.
Initialize noisy threshold: $\tilde{T} \leftarrow n - \frac{6}{\varepsilon}\ln\left(\frac{W+2}{\beta}\right) + \mathrm{Lap}\left(\frac{3}{\varepsilon}\right)$
for i from 0 to W **do**
  if $(\mathrm{sc}^P(w) + \mathrm{Lap}\left(\frac{3}{\varepsilon}\right) < \tilde{T})$ **then** ▷ as defined in Equation (3)
    return $(1+\gamma)w$ ▷ the next-to-last guess whose noisy score $\geq \tilde{T}$.
  else
    Update $w \leftarrow w \cdot \frac{1}{1+\gamma}$
  end if
end for
return $w = 0$

▶ **Theorem 5.** *Algorithm 1 is a ε-DP, that returns w.p. $\geq 1-\beta$ an $(3\gamma, \Delta)$-approximation of the width for $\Delta = O(\frac{d}{\varepsilon} \cdot \log(\frac{\log(1/\tau)}{\beta\gamma}))$. Moreover, let $T = T(n, d, \log(1/\tau))$ denote the runtime of computing the score of a region (Definition 4). Then Algorithm 1 runs in time $O(\frac{d\log(1/\tau)}{\gamma} \cdot \frac{1}{\gamma}^{\frac{d-1}{2}} \cdot T)$*

Proof.

Privacy. The fact this algorithm is ε-DP follows directly from Theorem 2. The entire algorithm is composed of $W + 1 = O(\log(1/\tau)/\gamma))$ queries, each is 1-sensitive, for which we run the SVT algorithm, therefore it is ε-DP. Moreover, simple tail bounds on the Laplace distribution give that w.p. $1-\beta$ all $W+2$ random variables drawn from $\mathrm{Lap}(\frac{3}{\varepsilon})$ in the execution of Algorithm 1 are all of magnitude $\leq \frac{3}{\varepsilon}\log(\frac{W+2}{\beta})$.

Utility. We argue that for each guess w_i satisfying $w_i \geq w(P)(1+\gamma)$, it holds that $\mathrm{sc}(w_i) = n$. Let x^* be an optimal solution to Equation (1), before restricting to the discretized directions V_θ, and let a^* be the closest direction to x^* in V_θ. Decompose x^* as $x^* = \lambda^* a^* + y^*$, where $y^* \perp a^*$. By the grid approximation guarantee, we have $\lambda^* = a^* \cdot x^* \geq \frac{1}{w(P)(1+\gamma)}$. Since $w_i \geq w(P)(1+\gamma)$, it follows that $\lambda^* \geq \frac{1}{w_i}$.

In the algorithm, the coefficient of a^* is required to be exactly $\frac{1}{w_i}$. Thus, consider the scaled vector $x^{**} = \eta x^* = \frac{1}{w_i}a^* + \eta y^*$, where $\eta = \frac{1}{w_i\lambda^*}$ and therefore $0 \leq \eta \leq 1$. Since the projections of all n points onto x^* lie within an interval of length 1, scaling by $\eta \leq 1$ implies that the projections onto x^{**} lie within an interval of length at most $\eta \leq 1$. Hence, $\mathrm{sc}_{w_i,a^*}(\eta y^*) = n$.

Therefore, for each guess $w_i \geq w^*(P)(1+\gamma)$ we get that $\mathrm{sc}(w_i) = n$. It follows that the noisy score must be $\geq n - \frac{3}{\varepsilon}\log(\frac{W+2}{\beta})$ whereas the noisy threshold $\tilde{T} \leq n - \frac{3}{\varepsilon}\log(\frac{W+2}{\beta})$ and so the SVT skips on this guess and continues to the next guess. Therefore we can only halt once $w_i < w(P)(1+\gamma)$, which implies we return $w_i(1+\gamma) \leq (1+\gamma)^2 w(P) \leq (1+3\gamma)w(P)$ (assuming $\gamma \leq 1$). Finally, observe that the noisy score at guess $w_{i-1} = (1+\gamma)w_i$ was greater than $\tilde{T}$. It follows we return guess w_{i-1} for which the score must be at least $n - \frac{12}{\varepsilon}\log(\frac{W+2}{\beta})$, hence $\Delta = \frac{12}{\varepsilon}\log(\frac{W+2}{\beta})$ as required.

Runtime. Lastly, the runtime is composed of W iterations in which we do $O(1)$-work plus the time of computing $\mathrm{sc}(w)$. In order to compute $\mathrm{sc}(w)$ as per Equation (3) we need to iterate over all vectors $a \in V_\theta$, of which there are $O(\frac{1}{\sqrt{\gamma}}^{d-1})$-many, and for each one find the max score of any point in the subspace orthogonal to a – which takes T-time. So overall our runtime is $O(W \cdot |V_\theta| \cdot T)$. ◀

3.1 Finding the Direction that Yields the Width

Next we turn our attention to finding an approximated solution to the LP, and not just its value. Assume Algorithm 1 has returned some $\tilde{w}$. The main idea is to do the DP-version of tracing-back our steps in computing the score to find a good solution x for our problem.

In more detail, given $\tilde{w}$, we first run another SVT to find the direction a with a large score $\mathrm{sc}_{\tilde{w}}(a)$ – i.e., where only a few points do not fall into an interval of size 1. This SVT involves $|V_\theta|$ many queries. Once a is found, we conduct a binary search over the region of possible solutions: we equi-partition the region R along some axis to obtain two rectangular regions R_1 and $R_2 = R \setminus R_1$, query to see whether the score of R_1 is large – and if so we continue to R_1, otherwise we continue to R_2. We halt our binary search once our region R holds a single grid point and return that point. Given the solution $\tilde{x}$ to the LP, the last step is to conduct a search along the direction of x for an interval of width $\tilde{w}$ that contains many points, thereby finding the two hyperplanes that yield the width.

There's just one subtlety. While the given n points of P do reside on the given grid $\mathcal{G}$, it doesn't assure as that the solution of the $\mathrm{LP}(a)$ lies on the same grid. Moreover, in our binary search we look for approximated solutions of $\mathrm{LP}(a)$ (one that only places the projection on points of some subset $P' \subset P$ inside an interval of length 1). Luckily, we already established in Section 2.2.1 that each coordinate of any vertex in such a polytope is upper bounded by a ratio of two determinants of matrices whose entries are integer multiplications of τ. Thus, each coordinate of a solution we seek is upper bounded by $X = (\frac{\sqrt{d+1}}{\tau})^{d+1}$, and any non-zero coordinate is lower bounded in magnitude by $\frac{1}{X} = (\frac{\sqrt{d+1}}{\tau})^{-(d+1)}$. So let $\mathcal{G}'$ be the grid in $[-X, X]^d$ of granularity $\frac{1}{X}$. (Thus, the new grid has X^{2d} many points.) Our binary search is conducted on this grid. The full details of the algorithm are given in Algorithm 3, which is deferred to Appendix A along with its proof of correctness.

4 Computing the Score in 2D

We now turn our attention to the question of computing the score of a region $R \subset \mathbb{R}^{d-1}$. In this section, we detail the algorithm that finds a score for a region when the input is two-dimensional. This means that the region in which we search for a point of maximal score is simply a closed interval I.

Recall that our goal is to find $\mu^* \in I$ which maximizes the score $\mathrm{sc}_{w,a}(\mu)$, which is defined as the maximal number of points whose dot-product with the vector $x_\mu = \frac{1}{w}a + \mu a^\perp$ reside in an interval of length 1. So we represent any $p \in P$ as $p = \lambda_p a + \mu_p a^\perp$, and it follows that $x \cdot p = \frac{\lambda_p}{w} + \mu\mu_p$. Algorithm 2, we iterate over all possible choices of two distinct points, p_1 and p_2, which x_μ sends to the bottom and top of the interval of length 1 resp.

▷ Claim 6. Algorithm 2 returns $\mathrm{sc}_{w,a}(I)$ in time $O(n^3 \log(n))$.

Proof. Fix I. Let $\mu^* \in I$ be the point of maximal score. Let P^* be the set of points whose dot-products with $x^* := x(\mu^*) = \frac{1}{w}a + \mu^* a^\perp$ lie inside an interval of length 1. Thus $\mathrm{sc}_{w,a}(\mu^*) = |P^*|$.

Algorithm 2 Compute Score for an Interval (when the Data is 2-Dimensional).

Input: Width w, direction a and its orthogonal direction $a^\perp$, dataset $P \subset \mathbb{R}^2$, interval $I = [I_s, I_e]$.
Split each $p \in P$ to its component over a and $a^\perp$: $\lambda_p = \langle p, a\rangle$, $\mu_p = \langle p, a^\perp\rangle$.
Initialize candidate set $C \leftarrow \{I_s, I_e\}$.
for each point $p_1 \in P$ **do** ▷ lower point
  for each point $p_2 \in P$, $p_2 \neq p_1$ **do** ▷ upper point
    if $(\mu_{p_1} = \mu_{p_2})$ **then**
      continue
    end if
    Solve for μ the equation: $\left(\frac{\lambda_{p_2}}{w} + \mu\mu_{p_2}\right) - \left(\frac{\lambda_{p_1}}{w} + \mu\mu_{p_1}\right) = 1$.
    if $(I_s \leq \mu \leq I_e)$ **then**
      Add μ to C
    end if
  end for
end for
return $\max_{\mu \in C}\{\mathrm{sc}_{w,a}(\mu)\}$ ▷ compute the score for each $\mu \in C$ and return the largest

For each pair p, q of distinct points in P we say it is *degenerate* if $\mu_p = \mu_q$, and otherwise *admissible*. Note that pairs that are degenerate have a fixed distance between their dot-products with x regardless of the value of μ. It is only the distance between admissible pairs that is affected by the value of μ. For a set P and vector x, define the maximal distance within P's dot-products with x as

$$f(P, x) \;=\; \max_{p,q \in P \text{ admissible}} \big(x \cdot p \;-\; x \cdot q\big).$$

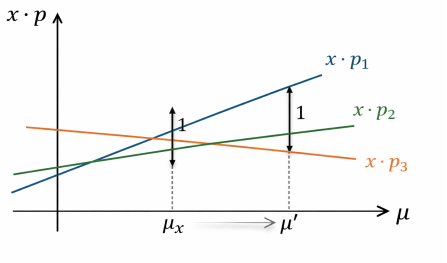
Our focus in this proof is on $f(P^*, x^*)$. In the extreme case there are no admissible pairs in P^*, which makes $f(P^*, x^*)$ ill-defined. But in this case it follows that all points in P^* have the same μ_p value, and so the distance between their dot-products with x isn't affected by μ. So computing the score on either ends of I should allow us to find $|P^*|$.

So assume $f(P^*, x^*)$ is well-defined. By the definition of P^* we know that all dot-products lie inside an interval of length 1, so $f(P^*, x^*) \leq 1$. Now if $f(P^*, x^*) = 1$, then there exists an admissible pair p^*, q^* so that $(q^* - p^*) \cdot x^* = 1$. Clearly, p^*, q^* define μ^*, and so when the for-loop of our algorithm reaches this admissible pair, we add μ^* to the list of candidate values, and the algorithm therefore returns $\mathrm{sc}_{w,a}(\mu^*)$.

Otherwise, $f(P^*, x^*) < 1$. As we increase μ the quantity $x(\mu) \cdot (p - q)$ varies linearly for each pair $p, q \in P^*$. Since $f(P^*, x(\mu))$ is the maximum over finitely many such linear functions, it varies continuously in μ. Define

$$\mu' \;=\; \min\Big\{\mu > \mu^* \;:\; \exists \text{an admissible pair } p, q \in P^* \text{ with } x(\mu) \cdot p - x(\mu) \cdot q = 1\Big\}.$$

Note that this set of larger μ-s isn't empty. Fix some pair of admissible p and q with $\mu_p > \mu_q$, and we can see that the function $f(\mu) = x(\mu) \cdot (p - q) = \frac{\lambda_p - \lambda_q}{w} + \mu(\mu_p - \mu_q)$ is increasing in μ. As at μ^*, by assumption, all admissible pairs satisfy $x^* \cdot (p - q) < 1$, then we must increase μ to obtain the value that sets $x(\mu) \cdot (p - q) = 1$.



Having established that μ' is well defined, denote $x' = x(\mu')$. By construction, we have $f(P^*, x') = 1$, and moreover, for every $\mu \in [\mu^*, \mu')$ we have that $f(P^*, x(\mu)) < 1$. Thus all points in P^* have dot-products with $x(\mu)$ that fall inside an interval of length 1, and so $\mathrm{sc}_{w,a}(\mu) = |P^*|$ for every $\mu \in [\mu^*, \mu']$. Now, in its for-loop, our algorithm reaches the admissible pair that yields μ'. And so if $\mu' \leq I_e$ we add it to C, and we have that $\mathrm{sc}_{w,a}(\mu') = |P^*|$ is returned by the algorithm. Otherwise, $I_e \leq \mu'$, but in this case we test the score on I_e, and so $\mathrm{sc}_{w,a}(I_e)$ is returned by the algorithm.

Finally, observe that our algorithm does $O(n^2)$ iterations to find the μ values, and then does $O(n(2+\log(n)))$ work per score computation, so overall the runtime of our algorithm is $O(n^3 \log(n))$. This completes the proof. ◁

We comment that the algorithm can be "massaged" so that it computes the score of all $O(n^2)$ possible μ-values in advance, and given an interval I finds the max-score by looking at the appropriate intersection. This is especially useful when Algorithm 3 runs its binary search and repeatedly invokes Algorithm 2 over the same choice of w and a.

5 Computing the Score in General Dimension

In this section we generalize the previous 2D-algorithm to a general dimension d. The algorithm's main idea is again to brute-force search over a pairs of points that will determine the end points of the interval of length 1 and then guess additional $d-2$ constraints: either points at the interval extremes or region bounding constraints. For each choice of d tight constraints we solve the linear system to obtain a candidate point. This fixes at most $O(n^d)$ candidate points. Due to brevity, we defer this section in its entirety to Appendix B.

6 Open Problems

In this work we give the first DP algorithm that bi-approximates the width of a dataset P, one of the first measures of a shape of P. Our work leaves many questions unanswered. First, we ponder as to a lower-bound for the problem: can we argue that any ε-DP algorithm that approximates the width must omit $\Omega(\Delta)$ many points? Or does there exists a better bi-criteria approximation of the width?

A second question we raise is regarding the efficiency of the binary search for the point of highest score inside a region. In our work we used the most straight-forward binary search, using pure-DP. However, is it possible to adapt the (ε, δ)-DP technique of Bun et al. [6] for binary search to this setting (over a general function)?[3] This should allow us to conduct a binary search with k queries while omitting only $O(\log(k))$ many points.

[3] The score function isn't quasi-concave, so the recursive technique of [3] is inapplicable in this setting.

Thirdly, we ponder as to the applicability of our approach to general LPs. Can the ideas from this work be successfully applied to any low-dimensional LP? Is there a simpler and more efficient algorithm than the algorithm of [17] for solving LPs in lower dimensions?

Lastly, we would like to extend our work to other notions and properties studied in computational geometry, and devise DP algorithms for estimating those as well. (This includes other notions of responsible computing – like fair algorithms or reproducible algorithms.) Of course, this requires us first to devise robust analogs of these notions and properties, analogs that generalize. Namely, if it is the case that P is drawn i.i.d. from some distribution $\mathcal{P}$, then the estimator over P should be indicative as to the same property over $\mathcal{P}$. Ultimately, we would love to describe P's shape using DP algorithms – afterall, if P's shape is "robust," then adding or removing a single datum shouldn't change its overall shape, and we should be able to say that P's in the shape of say, a rectangle, using DP techniques.

References

1 Pankaj K. Agarwal and Micha Sharir. Efficient randomized algorithms for some geometric optimization problems. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, SCG '95, pages 326–335, New York, NY, USA, 1995. Association for Computing Machinery. doi:10.1145/220279.220314.

2 Amos Beimel, Shay Moran, Kobbi Nissim, and Uri Stemmer. Private center points and learning of halfspaces. In Alina Beygelzimer and Daniel Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 269–282. PMLR, 2019. URL: http://proceedings.mlr.press/v99/beimel19a.html.

3 Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2013. doi:10.1007/978-3-642-40328-6_26.

4 Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 609–618. ACM, 2008. doi:10.1145/1374376.1374464.

5 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 634–649. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.45.

6 Mark Bun, Thomas Steinke, and Jonathan R. Ullman. Make up your mind: The price of online queries in differential privacy. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1306–1325. SIAM, 2017. doi:10.1137/1.9781611974782.85.

7 Vítor Gomes Chagas, Elisa Dell'Arriva, and Flávio Keidi Miyazawa. A framework for efficient approximation schemes on geometric packing problems of d-dimensional fat objects. *arXiv preprint*, 2024. arXiv:2405.00246.

8 Timothy M Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 300–309, 2000. doi:10.1145/336154.336216.

9 Edith Cohen, Xin Lyu, Jelani Nelson, Tamás Sarlós, and Uri Stemmer. Optimal differentially private learning of thresholds and quasi-concave optimization. In *Proceedings of the 55th Annual ACM Symposium aon Theory of Computing*, pages 472–482, 2023. doi:10.1145/3564246.3585148.

10 Minati De, Saksham Jain, Sarat Varma Kallepalli, and Satyam Singh. Online geometric covering and piercing. *Algorithmica*, 86(9):2739–2765, 2024. doi:10.1007/S00453-024-01244-1.

11 Christian A Duncan, Michael T Goodrich, and Edgar A Ramos. E cient approximation and optimization algorithms for computational metrology. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 121–130, 1997. URL: http://dl.acm.org/citation.cfm?id=314161.314196.

12 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. doi:10.1561/0400000042.

13 Yue Gao and Or Sheffet. Differentially private approximations of a convex hull in low dimensions. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, Virtual Conference, July 23-26, 2021*, volume 199 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITC.2021.18.

14 Sariel Har-Peled. *Geometric approximation algorithms*. Number 173 in Mathematical Surveys and Monographs. American Mathematical Soc., 2011.

15 Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *Conference on learning theory*, pages 2263–2285. PMLR, 2020. URL: http://proceedings.mlr.press/v125/kaplan20a.html.

16 Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Differentially private learning of geometric concepts. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3233–3241. PMLR, 2019. URL: http://proceedings.mlr.press/v97/kaplan19a.html.

17 Haim Kaplan, Yishay Mansour, Shay Moran, Uri Stemmer, and Nitzan Tur. On differentially private linear algebra. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 2362–2373, 2025. doi:10.1145/3717823.3718274.

18 Haim Kaplan, Yishay Mansour, Uri Stemmer, and Eliad Tsfadia. Private learning of halfspaces: Simplifying the construction and reducing the sample complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 13976–13985, 2020.

19 Haim Kaplan, Micha Sharir, and Uri Stemmer. How to find a point in the convex hull privately. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry, SoCG 2020, Zürich, Switzerland, June 23-26, 2020*, volume 164 of *LIPIcs*, pages 52:1–52:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.SOCG.2020.52.

20 Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 298–309, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316377.

21 Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007, Providence, RI, USA, October 20-23, 2007, Proceedings*, pages 94–103, 2007. doi:10.1109/FOCS.2007.66.

22 Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *Algorithmic Learning Theory*, pages 619–653. PMLR, 2018. URL: http://proceedings.mlr.press/v83/nissim18a.html.

23 Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Locating a small cluster privately. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 413–427, 2016. doi:10.1145/2902251.2902296.

24 Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with renyi differential privacy. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. URL: https://openreview.net/forum?id=-70L8lpp9DF.

25 F.P. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Monographs in Computer Science. Springer New York, 1993. URL: https://books.google.co.il/books?id=gFtvRdUY09UC.

26 Tukey J. W. Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians, Vancouver, 1975*, 2:523–531, 1975.

A A Differentially Private Algorithm for Finding the Hyperplanes that Yield the Width

Algorithm 3 Differentially Private Estimation of the Direction x on which The Width is Obtained.

1: **Input:** Dataset $P \subset \mathcal{G}$ of size n, estimated width w with a guarantee of enclosing $n' = n - \Delta$ many points. Privacy budget ε, approximation parameter $\gamma > 0$, failure parameter β.
2: Set $\theta \leftarrow \sqrt{\gamma}$ and V_θ as a θ-cover of the unit sphere.
3: Initialize noisy threshold: $\tilde{T} \leftarrow n - \Delta - \frac{6}{\varepsilon}\log\left(\frac{|V_\theta|+1}{\beta}\right) + \mathrm{Lap}\left(\frac{3}{\varepsilon}\right)$
4: Set $b \leftarrow \perp$
5: **for** each direction $a \in V_\theta$ **do**
6: **if** $\mathrm{sc}_w(a) + \mathrm{Lap}\left(\frac{3}{\varepsilon}\right) \geq \tilde{T}$ **then**
7: Set $b \leftarrow a$ and **break**
8: **end if**
9: **end for**
10: **if** $b = \perp$ **then**
11: **return** fail
12: **end if**
13: Fix some orthonormal basis of the subspace perpendicular to b: $\{u_1, u_2, ..., u_{d-1}\}$.
14: Set $n'' \leftarrow n - \Delta - \frac{12}{\varepsilon}\log\left(\frac{|V_\theta|+1}{\beta}\right)$.
15: Set $X \leftarrow (\frac{\sqrt{d+1}}{\tau})^{d+1}$. Let $\mathcal{G}'$ be a grid of $[-X, X]^{d-1}$ of granularity $\frac{1}{X}$.
16: Set $i \leftarrow 0$, $R \leftarrow [-X, X]^{d-1}$, and $t \leftarrow \lceil 2(d-1)\log_2(X)\rceil$.
17: **while** R holds more than one grid point of $\mathcal{G}'$ **do**
18: Equipartition R along some axis to R_1 and $R_2 = R \setminus R_1$.
19: **if** $\mathrm{sc}_{w,b}(R_1) + \mathrm{Lap}(\frac{t}{\varepsilon}) \geq n'' - (2i+1)\cdot\frac{t}{\varepsilon}\log(\frac{t}{\beta})$ **then**
20: $R \leftarrow R_1$ and increment i
21: **else**
22: $R \leftarrow R_2$ and increment i
23: **end if**
24: **end while**
25: $p \leftarrow$ the single grid point in R.
26: **return** $\frac{1}{w}b + \sum_{j=1}^{d-1} p_j u_j$

▶ **Theorem 7.** *Algorithm 3 is a 2ε-DP, that returns w.p. $\geq 1 - 2\beta$ a vector x so that the dot-product of at least $n - \Delta'$ many points with x lies in an interval of length 1, for $\Delta' = O(\frac{d}{\varepsilon}\log(\frac{1}{\gamma\beta}) + \frac{d^4}{\varepsilon}\log^2(\frac{d}{\tau})\log(\frac{\log(\frac{d}{\tau})}{\beta}))$. Moreover, let $T = T(n, d, \log(1/\tau))$ denote the runtime of computing the score of a region (Definition 4). Then Algorithm 3 runs in time $O((\frac{1}{\gamma})^{\frac{d-1}{2}} + d^2\log(\frac{d}{\tau}))\cdot T)$*

Proof.

Privacy. The algorithm is composed of two subalgorithms: one for finding a direction a with a large score (Lines 2-12); and another that finds a grid point of large score using binary search (Lines 13-25). The former is ε-DP as it simply a SVT over $|V_\theta|$-many queries, all of sensitivity 1. The latter is composed of $t = 2(d-1)\log(X) = O(d^2\log(\frac{d}{\tau}))$ many queries, each of sensitivity 1; so we partition the privacy budget to $\frac{\varepsilon}{t}$ and add noise of $\mathrm{Lap}(\frac{t}{\varepsilon})$ to each – which preserves ε-DP overall. Moreover, w.p. $\geq 1 - \beta$, all $|V_\theta| + 1$ r.v.s we draw in

the former part (from $\text{Lap}(\frac{3}{\varepsilon})$) are of magnitude $\leq \frac{3}{\varepsilon}\log(\frac{|V_\theta|}{\beta})$; and w.p. $\geq 1-\beta$, all t r.v.s we draw in the latter part (from $\text{Lap}(\frac{t}{\varepsilon})$) are of magnitude $\leq \frac{t}{\varepsilon}\log(\frac{t}{\beta})$. So w.p. $\geq 1-2\beta$ both event holds. We continue our analysis assuming all r.v.s are bounded in magnitude.

Utility. Based on Theorem 2, it is simple to see that the direction $b \in V_\theta$ we find satisfies that $\text{sc}_w(b) \geq n - \Delta - \frac{12}{\varepsilon}\ln\left(\frac{\log(|V_\theta|+1)}{\beta}\right)$. This implies that some point $y \in R$ satisfies $sc_{w,b}(y) \geq n''$ for the same n'' defined in Line 14.

We now argue inductively, that in each iteration i of the binary search we have a region R of score $\geq n'' - 2i \cdot \frac{t}{\varepsilon}\log(\frac{t}{\beta})$. The base case (iteration $i=0$) was already established. The induction step is also straight forward once we assume all r.v.s in the binary search have bounded magnitude – at most $\frac{t}{\varepsilon}\log(\frac{t}{\beta})$. If the induction hypothesis holds in the beginning of the iteration, then the point of score $n'' - 2i \cdot \frac{t}{\varepsilon}\log(\frac{t}{\beta})$ lies either in R_1 or R_2. If it is in R_1 then we ought to pass the threshold-query in Line 19, and continue to work with R_1 and then in the next iteration $\text{sc}_{w,b}(R_1) = \text{sc}_{w,b}(R)$ as required. Whereas this point of high score lies in R_2 we either fail or pass the threshold-query of Line 19: if we fail it then we continue to R_2 and $\text{sc}_{w,b}(R_2) = \text{sc}_{w,b}(R)$ as required; and if we pass it then we continue to R_1, where because of passing the query we know that $\text{sc}_{b,w}(R_1) \geq n'' - (2i+1)\cdot\frac{t}{\varepsilon}\log(\frac{t}{\beta}) - \frac{t}{\varepsilon}\log(\frac{t}{\beta}) = n'' - (2i+2)\cdot\frac{t}{\varepsilon}\log(\frac{t}{\beta})$.

It follows that end of the binary search, namely, after t iterations, we're left with a single grid point $p \in \mathcal{G}'$ of score $\geq n'' - \frac{2t\cdot t}{\varepsilon}\log(\frac{t}{\beta})$. And so $\Delta' = \Delta + \frac{12}{\varepsilon}\log(\frac{|V_\theta|+1}{\beta}) + \frac{2t^2}{\varepsilon}\log(\frac{t}{\beta}) = O(\frac{d}{\varepsilon}\log(\frac{1}{\gamma\beta}) + \frac{d^4}{\varepsilon}\log^2(\frac{d}{\tau})\log(\frac{\log(\frac{d}{\tau})}{\beta}))$.

Runtime. Lastly, we detail the runtime of Algorithm 3. The former part (SVT on all directions) takes $O(|V_\theta|\cdot T)$ time, and the binary search takes $O(t\cdot T)$. So overall we get a runtime of $O(\left(\frac{1}{\gamma}^{\frac{d-1}{2}} + d^2\log(\frac{d}{\tau})\right)T)$. ◀

We comment that Algorithm 3 finds a vector x so that the dot-product of $n-\Delta'$ many point with x lie in an interval of length 1. (So the projection onto the direction $\frac{x}{\|x\|}$ of at least $n-\Delta'$ many points fits inside an interval of width $\frac{1}{\|x\|}$.) If it is our goal to find a pair of parallel hyperplanes that hold $n-\Delta'$ points between them, then we need to return in addition to x some scalar c so that many points satisfy that their dot-product with x falls inside the interval $[c, c+1]$. This requires us to conduct another binary search for this c using the 1-dimensional dataset of projected points $p\cdot x : p \in P$. This binary search of course may omit a few more points from $n-\Delta'$, but only $\tilde{O}(\frac{\log^2(\frac{d}{\tau})}{\varepsilon})$ many. We thus omit the details of this last binary search as they are subsumed by the $(d-1)$-dimensional binary search for x.

B Computing the Score in General Dimension

In this section we generalize the previous 2D-algorithm to a general dimension. Recall, we are given a scalar w and a direction a and we fix some basis $\{u_1, u_2, ..., u_{d-1}\}$ of the subspace orthogonal to a. In addition, we are given a rectangular region $R = I_1 \times I_2 \times ... \times I_{d-1}$. This means that our goal is to find $d-1$ scalars $(\mu_1, \mu_2, ..., \mu_{d-1})$ so that when taking dot-product of the points in P with

$$x(\mu_1, \mu_2, ..., \mu_{d-1}) = \frac{1}{w}a + \sum_{i=1}^{d-1}\mu_i u_i$$

we maximize the number of points that fit into an interval of length 1. The algorithm for computing the score of the region R (which is the score of the best $(\mu_1, \mu_2, ..., \mu_{d-1}) \in R$) is given in Algorithm 4.

Algorithm 4 Compute Score for a Rectangular Region (when the Data is d-Dimensional).

Input: Width w, direction a with orthogonal basis $\{u_1, u_2, ...u_{d-1}\}$, dataset $P \subset \mathbb{R}^d$, rectangular region $R = [I_{s_1}, I_{e_1}] \times [I_{s_2}, I_{e_2}] \times ... \times [I_{s_{d-1}}, I_{e_{d-1}}]$.
Split each $p \in P$ into its coordinates over $\{a, u_1, u_2, ..., u_{d-1}\}$:

$$\lambda_p = \langle p, a \rangle, \qquad \mu_{i,p} = \langle p, u_i \rangle, \forall 1 \le i \le d-1$$

Initialize set of constraints S with $2(d-1)$ constraints of the form: $\{\mu_i = I_{s_i}\}, \{\mu_i = I_{e_i}\}$ for each $1 \le i \le d-1$.
for each pair $p_1, p_2 \in P$ **do**
    Add to S the constraint $\left\{ \frac{\lambda_{p_2} - \lambda_{p_1}}{w} + \sum_{i=1}^{d-1} \mu_i(\mu_{i,p_2} - \mu_{i,p_1}) = 1 \right\}$
end for
Initialize the candidate set $C \leftarrow \emptyset$.
for each distinct $d-1$ constraints in C **do**
    if the system of $d-1$ linear constraints is invertible **then**
        Set its solution as $(\mu_1, \mu_2, ..., \mu_{d-1})$.
        if for each j, $I_{s_j} \le \mu_j \le I_{e_j}$ **then**
            Add $(\mu_1, \mu_2, ..., \mu_{d-1})$ to C
        end if
    end if
end for
return $\max_{(\mu_1, \mu_2, ..., \mu_{d-1}) \in C} \mathrm{sc}_{w,a}(\mu_1, \mu_2, ..., \mu_{d-1})$ ▷ evaluate the score at each candidate and return the maximum

▶ **Lemma 8.** *Algorithm 4 returns* $\mathrm{sc}_{w,a}(R)$.

Proof. Fix a vector $x^* \in R$ of maximal score, and let P^* be the maximal set of points whose dot-products with x^* lie in an interval of length 1. We show that we can find a vector x' satisfying $d-1$ constraints from S with the property that the dot-products of all points in the same P^* with x' fall inside an interval of length 1.

We proceed by induction on the number $|S'|$ of linearly independent constraints currently satisfied by x^*, and show how to iteratively shift x^* to a new vector that satisfies more and more constraints in S. Starting from the base case: $|S'| = 0$. This implies that all dot-product differences $x^* \cdot (p_i - p_j)$ for $p_i, p_j \in P^*$ are strictly less than 1. Fix a direction v in the $(d-1)$-dimensional parameter space, and move x continuously from x^* along v: in the affine subspace $\{x^* + \lambda v\}_{\lambda \in \mathbb{R}}$. Because all dot-product differences vary continuously and only finitely many pairs exist, there is a smallest $\lambda \ge 0$ such that either (i) some pair $p_i, p_j \in P^*$ satisfies $(x^* + \lambda v) \cdot (p_2 - p_1) = 1$, or (ii) we reach the boundary of the feasible rectangle. In either case a new constraint becomes tight, so set $x^* \leftarrow x^* + \lambda_0 v$, which means now $|S'| \ge 1$.

Now assume $|S'| \ge 1$. The set of vectors satisfying the current constraints in S' is an affine space of dimension $d - 1 - |S'|$. As long as $|S'| < d-1$, this space has positive dimension, so fix some nonzero direction v that preserves all currently tight constraints. We now move x^* to $x^* + \lambda v$, where λ is the smallest value for which either a new pair $p_i, p_j \in P^*$ reaches projection gap 1, or a new boundary of the rectangle is hit. (Here "new pair" means that their

projection gap at x^* was strictly less than 1, i.e., $x^* \cdot (p_j - p_i) < 1$.) Since shifting x^* along v preserves all previously tight equalities and never increases any projection gap beyond 1, all points of P^* remain inside some interval of length 1. Thus one new constraint is added and $|S'|$ increases. We argue that this additional constraint leaves S' linearly independent: it is the only constraint with a component in direction v whereas all other constraints originally in S' were orthogonal to v.

Since at most $d-1$ constraints can become tight, repeating this process finitely many times yields a vector x' satisfying exactly $d-1$ linearly independent constraints, while still preserving the property that all points of P^* project into an interval of length 1. ◀

Now, as written, Algorithm 4 runs in time $O(n^{2(d-1)} \cdot \omega_{d-1})$ where ω_d is the time required to solve a system of d linear equations in d variables. From the proof it follows that we can refine our search – rather than trying all $O(n^2)$ pairs of points for each constraint, our search can work as follows. First we pick one pair of points p_1, p_2 that go to the two endpoints of the interval (namely, so that $x \cdot (p_1 - p_2) = 1$). Once p_1 and p_2 are chosen, iterate over the choice of $d-2$ other $q_1, q_2, ..., q_{d-2}$ points and all 2^{d-2} options of assigning them to the same end as p_1 or the same end as p_2. This gives $d-2$ additional equations of the form $x \cdot (p_1 - q_i) = 0$ (or $x \cdot (p_2 - q_i) = 0$), and now solve the system of $d-1$ equations in $d-1$ variables in time ω_{d-1}. This requires time $O(n^2 \cdot (2n)^{d-2} \cdot \omega_{d-1}) = O(n^d \omega_{d-1})$. Now, each one of those might yield a vertex on which we need to test the score, so this costs overall $O(n^d \omega_{d-1} + n^{d+1} \log(n))$.