Private Center Points and Learning of Halfspaces

Amos Beimel AMOS.BEIMEL@GMAIL.COM
Ben-Gurion University

Shay Moran SHAYM@PRINCETON.EDU
Princeton University

Kobbi Nissim KOBBI.NISSIM@GEORGETOWN.EDU
Georgetown University

Uri Stemmer U@URI.CO.IL
Ben-Gurion University

Editors: Alina Beygelzimer and Daniel Hsu

Abstract

We present a private agnostic learner for halfspaces over an arbitrary finite domain $X \subset \mathbb{R}^d$ with sample complexity $\mathrm{poly}(d, 2^{\log^* |X|})$. The building block for this learner is a differentially private algorithm for locating an approximate center point of $m > \mathrm{poly}(d, 2^{\log^* |X|})$ points – a high dimensional generalization of the median function. Our construction establishes a relationship between these two problems that is reminiscent of the relation between the median and learning one-dimensional thresholds [Bun et al. FOCS '15]. This relationship suggests that the problem of privately locating a center point may have further applications in the design of differentially private algorithms.

We also provide a lower bound on the sample complexity for privately finding a point in the convex hull. For approximate differential privacy, we show a lower bound of $m = \Omega(d + \log^* |X|)$, whereas for pure differential privacy $m = \Omega(d \log |X|)$.

Keywords: Differential privacy, Private PAC learning, Halfspaces, Quasi-concave functions.

1. Introduction

Machine learning models are often trained on sensitive personal information, e.g., when analyzing healthcare records or social media data. There is hence an increasing awareness and demand for privacy preserving machine learning technology. This motivated the line of works on *private learning*, initiated by Kasiviswanathan et al. (2011), which provides strong (mathematically proven) privacy protections for the training data. Specifically, these works aim at achieving *differential privacy*, a strong notion of privacy that is now increasingly being adopted by both academic researchers and industrial companies. Intuitively, a private learner is a PAC learner that guarantees that every single example has almost no effect on the resulting classifier. Formally, a private learner is a PAC learner that satisfies differential privacy w.r.t. its training data. The definition of differential privacy is,

Definition 1 (Dwork et al. (2006)) *Let $\mathcal{A}$ be a randomized algorithm. Algorithm $\mathcal{A}$ is (ε, δ)-*differentially private *if for any two databases S, S' that differ on one row, and any event T, we have $\Pr[\mathcal{A}(S) \in T] \le e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in T] + \delta$. The notion is referred to as* pure *differential privacy when $\delta = 0$, and* approximate *differential privacy when $\delta > 0$.*

The initial work of Kasiviswanathan et al. (2011) showed that any finite concept class C is privately learnable with sample complexity $O(\log |C|)$ (we omit in the introduction the dependencies on accuracy and privacy parameters). Non-privately, $\Theta(VC(C))$ samples are necessary and sufficient to PAC learn C, and much research has been devoted to understanding how large the gap is between the sample complexity of private and non-private PAC learners. For *pure* differential privacy, it is known that a sample complexity of $\Theta(\log |C|)$ is required even for learning some simple concept classes such as one-dimensional thresholds, axis-aligned rectangles, balls, and halfspaces (Beimel et al., 2014, 2013a; Feldman and Xiao, 2015). That is, generally speaking, learning with pure differential privacy requires sample complexity proportional to log the size of the hypothesis class. For example, in order to learn halfspaces in $\mathbb{R}^d$, one must consider some *finite* discretization of the problem, e.g. by assuming that input examples come from a finite set $X \subseteq \mathbb{R}^d$. A halfspace over X is represented using d point from X, and hence, learning halfspaces over X with pure differential privacy requires sample complexity $\Theta(\log \binom{|X|}{d}) = O(d \log |X|)$. In contrast, learning halfspaces non-privately requires sample complexity $O(d)$. In particular, when the dimension d is constant, learning halfspaces non-privately is achieved with *constant* sample complexity, while learning with pure differential privacy requires sample complexity that is proportional to the representation length of domain elements.

For *approximate* differential privacy, the current understanding is more limited. Recent results established that the class of one-dimensional thresholds over a domain $X \subseteq \mathbb{R}$ requires sample complexity between $\Omega(\log^* |X|)$ and $2^{O(\log^* |X|)}$ (Beimel et al. (2013b); Bun et al. (2015); Bun (2016); Alon et al. (2018)). On the one hand, these results establish a separation between what can be learned with or without privacy, as they imply that privately learning one-dimensional thresholds over an infinite domain is impossible. On the other hand, these results show that, unlike with pure differential privacy, the sample complexity of learning one-dimensional thresholds can be much smaller than $\log |C| = \log |X|$. Beimel et al. (2013b) also established an upper bound of $\mathrm{poly}(d \cdot 2^{\log^* |X|})$ for privately learning the class of axis-aligned rectangles over $X \subseteq \mathbb{R}^d$. In a nutshell, this concludes our current understanding of the sample complexity of approximate private learning. In particular, before this work, it was not known whether similar upper bounds (that grow slower than $\log |C|$) can be established for "richer" concept classes, such as halfspaces, balls, and polynomials.

We answer this question positively, focusing on privately learning halfspaces. The class of halfspaces forms an important primitive in machine learning as learning halfspaces implies learning many other concept classes (Ben-David and Litman (1998)). In particular, it is the basis of popular algorithms such as neural nets and kernel machines, as well as various geometric classes (e.g., polynomial threshold functions, polytopes, and d-dimensional balls).

Our Results. Our approach for privately learning halfspaces is based on a reduction to the task of privately finding a point in the convex hull of a given input dataset. That is, towards privately learning halfspaces we first design a sample-efficient differentially private algorithm for identifying a point in the convex hull of the given data, and then we show how to use such an algorithm for privately learning halfspaces.

Privately finding a point in the convex hull. We initiate the study of privately finding a point in the convex hull of a dataset $S \subseteq X \subseteq \mathbb{R}^d$. Even though this is a very natural problem (with important applications, in particular to learning halfspaces), it has not been considered before in the literature of differential privacy. One might try to solve this problem using the *exponential mechanism* of McSherry and Talwar (2007), which, given a dataset and a *quality function*, privately identifies a point with approximately maximum quality. To that end, one must first settle on a suitable quality function such that if a point $\mathbf{x} \in X$ has a high quality then this point is guaranteed to be in the convex hull of S. Note that the indicator function $q(\mathbf{x}) = 1$ if and only if $\mathbf{x}$ is in the convex hull of S is not a good option, as every point $\mathbf{x} \in X$ has quality either 0 or 1, and the exponential mechanism only guarantees a solution with *approximately* maximum quality (with additive error larger than 1).

Our approach is based on the concept of *Tukey depth* (Tukey, 1975). Given a dataset $S \subseteq \mathbb{R}^d$, a point $\mathbf{x} \in \mathbb{R}^d$ has the Tukey depth at most ℓ if there exists a set $A \subseteq S$ of size ℓ such that $\mathbf{x}$ is not in the convex hull of $S \setminus A$. See Section 2 for an equivalent definition that has a geometric flavor. Instantiating the exponential mechanism with the Tukey depth as the quality function results in a private algorithm for identifying a point in the convex hull of a dataset $S \subseteq X \subseteq \mathbb{R}^d$ with sample complexity $\mathrm{poly}(d, \log|X|)$.[1] We show that this upper bound can be improved to $\mathrm{poly}(d, 2^{\log^*|X|})$. Our construction utilizes an algorithm by Beimel et al. (2013b) for approximately maximizing (one-dimensional) quasi-concave functions with differential privacy (see Definition 4 for quasi-concavity). To that end, we show that it is possible to find a point with high Tukey depth in iterations over the axes, and show that the appropriate functions are indeed quasi-concave. This allows us to instantiate the algorithm of Beimel et al. (2013b) to identify a point in the convex hull of the dataset one coordinate at a time. We obtain the following theorem.

Theorem 2 (Informal) *Let $X \subseteq \mathbb{R}^d$. Then, there exists an (ε, δ)-differentially private algorithm that given a dataset $S \in X^m$ identifies (w.h.p.) a point in the convex hull of S, provided that $m = |S| = \mathrm{poly}\left(d, 2^{\log^*|X|}, \frac{1}{\varepsilon}, \log\frac{1}{\delta}\right)$.*

In fact, our algorithm returns a point with a large Tukey-depth (i.e., an approximate center point), which is in particular a point in the convex hull of the dataset.

A privacy preserving reduction from halfspaces to convex hull. We present a reduction from private learning halfspaces to privately finding a point in the convex hull of a set of points, that is, we show how to construct a private algorithm for learning halfsapces given a black-box that privately finds a point in the convex hull. Furthermore, we present a better analysis of our reduction for the case that the black-box returns a point with high Tukey depth (e.g., an approximate center point). Thus, using our private algorithm that finds an approximate center point results in a private learning algorithm with better sample complexity.

Our reduction can be thought of as a generalization of the results by Bun et al. (2015), who showed that the task of privately learning *one-dimensional* thresholds is equivalent to the task of privately solving the *interior point problem*. In this problem, given a set of input numbers, the task is to identify a number between the minimal and the maximal input numbers. Indeed, this is exactly the one dimensional version of the convex hull problem we consider. However, the reduction of Bun et al. (2015) does not apply for halfspaces, and we needed to design a different reduction.

Our reduction is based on the sample and aggregate paradigm: assume a differentially private algorithm $\mathcal{A}$ which gets a (sufficiently large) dataset $D \subset \mathbb{R}^d$ and returns a point in the convex

1. We remark that the domain X does not necessarily contain a point with a high Tukey depth, even when the input points come from $X \subseteq \mathbb{R}^d$. Hence, one must first *extend* the domain X to make sure that a good solution exists. We show how to construct a domain $\tilde{X}$ of size $|\tilde{X}| = |X|^{d^2}$ that contains a point with Tukey depth $\gtrsim m/d$, where m is the number of input points. Applying the exponential mechanism to select a point from $\tilde{X}$ results in a private algorithm with sample complexity $O(d^3 \log|X|)$.

hull of D. This can be used to privately learn halfspaces as follows. Given an input sample S, partition it to sufficiently many subsamples $S_1, \ldots, S_k$, and pick for each S_i an arbitrary halfspace h_i which is consistent with S_i. Next, apply $\mathcal{A}$ to privately find a point in the convex hull of the h_i's (to this end represent each h_i as a point in $\mathbb{R}^{d+1}$ via its normal vector and bias), and output the halfspace h corresponding to the returned point. It can be shown that if each of the h_i's has a sufficiently low generalization error, which is true if the sample is big enough, then the resulting (privately computed) halfspace also has a low generalization error. Instantiating this reduction with our algorithm for the convex hull we get the following theorem.

Theorem 3 (Informal) *Let $X \subseteq \mathbb{R}^d$. Then, there exists an (ε, δ)-differentially private (α, β)-PAC learner for halfspaces over X with sample complexity* $\text{poly}\left(d, 2^{\log^* |X|}, \frac{1}{\alpha\varepsilon}, \log\frac{1}{\beta\delta}\right)$.

In particular, for any constant d, Theorem 3 gives a private learner for halfspaces over $X \subseteq \mathbb{R}^d$ with sample complexity $2^{O(\log^* |X|)}$. Before our work, this was known only for $d = 1$.

Our construction also results in a private *agnostic* learner for halfspaces, where for a distribution on labeled examples (that might not be consistent with a halfspace) the goal is to privately find a halfspace whose error is close to the error of the best halfspace for this distribution. To get our agnostic learner we use a generic transformation of Beimel et al. (2015), who showed that if there exists a private learner for a class of functions, then there exists a private agnostic learner for the class with similar sample complexity. Using this transformation with our learner from Theorem 3, we get a private agnostic learner for halfspaces with sample complexity $m = \text{poly}\left(d, 2^{\log^* |X|}, \frac{1}{\alpha\varepsilon}, \log\frac{1}{\beta\delta}\right)$.

A lower bound for finding a point in the convex hull. Without privacy considerations, finding a point in the convex hull of the data is trivial. Nevertheless, we show that any (ε, δ)-differentially private algorithm for this task (in d dimensions) must have sample complexity $m = \Omega(\frac{d}{\varepsilon}\log\frac{1}{\delta} + \log^* |X|)$. In comparison, our algorithm requires sample of size at least $\tilde{O}(d^{2.5} 2^{O(\log^* |X|)}/\varepsilon)$ (ignoring the dependency on δ and β).

Recall that the sample complexity of privately learning a class C is always at most $O(\log |C|)$. Hence, it might be tempting to guess that a sample complexity of $m = O(\log |X|)$ should suffice for privately finding a point in the convex hull of a dataset $S \subseteq X \subseteq \mathbb{R}^d$, even with pure $(\varepsilon, 0)$-differential privacy. We show that this is not the case, and that any pure $(\varepsilon, 0)$-differentially private algorithm for this task must have sample complexity $m = \Omega(\frac{d}{\varepsilon}\log |X|)$.

Related Work. Most related to our work is the work on private learning and its sample and time complexity by Blum et al. (2005); Kasiviswanathan et al. (2011); Beimel et al. (2014); Chaudhuri and Hsu (2011); Beimel et al. (2013a); Feldman and Xiao (2015); Beimel et al. (2013b); Bun et al. (2015); Bun and Zhandry (2016); Kaplan et al. (2019). As some of these works demonstrate efficiency gaps between private and non-private learning, alternative models have been explored including semi-supervised learning (Beimel et al. (2015)), learning multiple concepts (Bun et al. (2016)), and prediction (Dwork and Feldman (2018), Bassily et al. (2018)).

Dunagan and Vempala (2008) showed an efficient (non-private) learner for halfspaces that works in (a variant of) the *statistical query (SQ)* model of Kearns (1998). It is known that SQ learners can be transformed to preserve differential privacy (Blum et al., 2005), and the algorithm of Dunagan and Vempala (2008) yields a differentially private efficient learner for halfspaces over examples from $X \subseteq \mathbb{R}^d$ with sample complexity $\text{poly}(d, \log |X|)$. Another related work is that of Hsu et al. (2014) who constructed an algorithm for *approximately* solving linear programs with differential privacy. While learning halfspaces *non-privately* easily reduces to solving linear programs, it is not clear whether the results of Hsu et al. (2014) imply a *private* learner for halfspaces (due to the types of errors they incur).

Blum et al. (2005) and Nguyen et al. (2019) studied the related problem of privately learning *large-margin* halfspaces, i.e., learning halfspaces under the assumption that no example lies too close to the separating hyperplane. Specifically, Nguyen et al. (2019) presented an algorithm with sample complexity $O(\frac{1}{\alpha\varepsilon\gamma^2})$, where γ quantifies the margin. In addition, several other works developed tools that implicitly imply private learning of large-margin halfspaces. These include the work of Blum et al. (2008) who presented an algorithm for the task of *private sanitization* of large-margin halfspaces, and the works of Chaudhuri et al. (2011) and Bassily et al. (2014) who studied private ERM. We remark that in this work we do not make any margin assumptions.

2. Preliminaries

In this section we introduce a tool that enables our constructions, describe the geometric object we use throughout the paper, and present some of their properties. See the full version of this paper for additional preliminaries.

Notations. The input of our algorithm is a multiset S whose elements are taken (possibly with repetition) from a set X. We will abuse notation and write that $S \subseteq X$. Databases S_1 and S_2 are called *neighboring* if they differ in exactly one entry. Throughout this paper we use ε and δ for the privacy parameters, α for the error parameter, and β for the confidence parameter, and m for the sample size.

A Private Algorithm for Optimizing Quasi-concave Functions – $\mathcal{A}_{\text{RecConcave}}$. We next describe properties of an algorithm $\mathcal{A}_{\text{RecConcave}}$ of Beimel et al. (2013b). This algorithm is given a quasi-concave function Q (defined below) and privately finds a point x such that $Q(x)$ is close to its maximum provided that the maximum of $Q(x)$ is large enough (see (1)).

Definition 4 *A function $Q(\cdot)$ is quasi-concave if $Q(\ell) \geq \min\{Q(i), Q(j)\}$ for every $i < \ell < j$.*

Definition 5 (Sensitivity) *The sensitivity of a function $f : X^m \to \mathbb{R}$ is the smallest k such that for every neighboring $D, D' \in X^m$, we have $|f(D) - f(D')| \leq k$.*

Proposition 6 (Properties of Algorithm $\mathcal{A}_{\text{RecConcave}}$ (Beimel et al., 2013b)) *Let $Q : X^* \times \tilde{X} \to \mathbb{R}$ be a sensitivity-1 function (that is, for every $x \in \tilde{X}$, the function $Q(\cdot, x)$ has sensitivity 1). Denote $\tilde{T} = |\tilde{X}|$ and let $\alpha \leq \frac{1}{2}$ and $\beta, \varepsilon, \delta, r$ be parameters. There exits an (ε, δ)-differentially private algorithm, called $\mathcal{A}_{\text{RecConcave}}$, such that the following holds. If $\mathcal{A}_{\text{RecConcave}}$ is executed on a database $S \in X^*$ such that $Q(S, \cdot)$ is quasi-concave and in addition*

$$\max_{i \in \tilde{X}}\{Q(S,i)\} \geq r \geq 8^{\log^* \tilde{T}} \cdot \frac{12 \log^* \tilde{T}}{\alpha\varepsilon} \log\left(\frac{192(\log^* \tilde{T})^2}{\beta\delta}\right), \tag{1}$$

then with probability at least $1 - \beta$ the algorithm outputs an index j s.t. $Q(S, j) \geq (1 - \alpha)r$.

Claim 7 *Let $\{f_t\}_{t \in \mathcal{T}}$ be a finite family of quasi-concave functions. Then, $f(x) = \min_{t \in \mathcal{T}} f_t(x)$ is also quasi-concave.*

The proof of Claim 7 appears in the full version of this work.

Private Agnostic-PAC Learning. In *agnostic* PAC learning, there is a distribution over labeled examples and the goal is to find a hypothesis whose error is close to the error of the best concept in class with respect to the distribution. Beimel et al. (2015) proved that any private learner for a class of functions C can be converted to a private agnostic learner for the class with similar sample complexity. Formally,

Theorem 8 (Beimel et al. (2015)) *Let C be a concept class with VC dimension $\mathrm{VC}(C)$. There exists a constant λ such that if there exists an $(1,\delta)$-differentially private (α,β)-PAC learner for C with sample complexity m, then for every $\varepsilon > 0$ there exists a (ε,δ)-differentially private $(\lambda\alpha,\lambda\beta)$-agnostic PAC learner for C with sample complexity $O\left(\frac{m}{\varepsilon} + \frac{1}{\alpha^2\varepsilon}\mathrm{VC}(C)\log(\frac{1}{\alpha\beta})\right)$.*

Halfspaces, Convex Hull, and Tukey Depth. We next define the geometric objects we use in this paper.

Definition 9 (Halfspaces and Hyperplanes) *Let $X \subset \mathbb{R}^d$. For $a_1,\dots,a_d,w \in \mathbb{R}$, let the halfspace $\mathrm{hs}_{a_1,\dots,a_d,w} : X \to \{0,1\}$ be defined as $\mathrm{hs}_{a_1,\dots,a_d,w}(x_1,\dots,x_d) = 1$ if and only if $\sum_{i=1}^d a_i x_i \geq w$. Define the concept class $\mathtt{HALFSPACE}(X) = \{\mathrm{hs}_{a_1,\dots,a_d,w}\}_{a_1,\dots,a_d,w\in\mathbb{R}}$. We say that a halfspace hs contains a point $\mathbf{x} \in \mathbb{R}^d$ if $\mathrm{hs}(\mathbf{x}) = 1$. The hyperplane $\mathrm{hp}_{a_1,\dots,a_d,w}$ defined by $a_1,\dots,a_d,w$ is the set of all points $\mathbf{x} = (x_1,\dots,x_d)$ such that $\sum_{i=1}^d a_i x_i = w$.*

Definition 10 *Let $S \subset \mathbb{R}^d$ be a finite multiset of points. A point $\mathbf{x} \in \mathbb{R}^d$ is in the convex hull of S if $\mathbf{x}$ is a convex combination of the elements of S, that is, there exists non-negative numbers $\{a_{\mathbf{y}}\}_{\mathbf{y}\in S}$ such that $\sum_{\mathbf{y}\in S} a_{\mathbf{y}} = 1$ and $\sum_{\mathbf{y}\in S} a_{\mathbf{y}}\mathbf{y} = \mathbf{x}$.*

We next define the Tukey median of a point, which generalizes the median to $\mathbb{R}^d$.

Definition 11 (Tukey depth (Tukey, 1975)) *Let $S \subset \mathbb{R}^d$ be a finite multiset of points. The Tukey depth of a point $\mathbf{x} \in \mathbb{R}^d$ with respect to S, denoted by $\mathrm{td}^S(\mathbf{x})$, is the minimum number of points in S contained in a halfspace containing the point x, that is,*

$$\mathrm{td}^S(\mathbf{x}) = \min_{\mathrm{hs}\in\mathtt{HALFSPACE}_{d,T},\mathrm{hs}(\mathbf{x})=1} |\{\mathbf{y} \in S : \mathrm{hs}(\mathbf{y}) = 1\}|.$$

The Tukey median of S is a point maximizing the Tukey depth. A centerpoint is a point of depth at least $|S|/(d+1)$.

Observation 12 *The Tukey depth of a point is a sensitivity one function of the multiset S.*

Claim 13 (Tukey depth, alternative definition) *Let $S \subset \mathbb{R}^d$ be a multiset of points. For a given $a_1,\dots,a_d \in \mathbb{R}$ define the function*

$$\mathrm{t}^S_{a_1,\dots,a_d}(w) \triangleq \min\left\{\left|\{(y_1,\dots,y_d) \in S : \sum_{i=1}^d a_i y_i \geq w\}\right|, \left|\{(y_1,\dots,y_d) \in S : \sum_{i=1}^d a_i y_i \leq w\}\right|\right\}. \tag{2}$$

Then,

$$\mathrm{td}^S(x_1,\dots,x_n) = \min_{(a_1,\dots,a_d)\in\mathbb{R}} \mathrm{t}^S_{a_1,\dots,a_d}\left(\sum_{i=1}^d a_i x_i\right). \tag{3}$$

Claim 14 (Tukey depth, another alternative definition) *Let $S \subset \mathbb{R}^d$ be a multiset of points. The Tukey depth of a point $\mathbf{x}$ is the size of the smallest set $A \subseteq S$ such that $\mathbf{x}$ is not in the convex hull of $S \setminus A$.*

Claim 15 (Yaglom and Boltyanskiĭ (1961); Edelsbrunner (1987)) *Let $S \subset \mathbb{R}^d$ be a multiset of points. There exists $\mathbf{x} \in \mathbb{R}^d$ such that $\mathrm{td}^S(\mathbf{x}) \geq |S|/(d+1)$.*

Thus, a centerpoint always exists and a Tukey median must be a centerpoint. However, not every centerpoint is a Tukey median. We will use the following regarding the set of points of whose Tukey depth is at least r.

Fact 16 (see e.g. Liu et al. (2014)) *Let $S \subseteq \mathbb{R}^d$ be a multiset of points and $r > 0$. Define $\mathcal{T}(r) = \{\mathbf{x} \in \mathbb{R}^d : \mathrm{td}^S(\mathbf{x}) \geq r\}$. Then $\mathcal{T}(r)$ is a polytope whose faces are supported by affine subspaces that are spanned by points from S.*

So, for example the set of all Tukey medians is a polytope and if it is d-dimensional then each of its facet is supported by a hyperplane that passes through $d+1$ points from S.

3. Finding a Point in the Convex Hull

Our goal is to privately find a point in the convex hull of a set of input points (i.e., the database). We will actually achieve a stronger task and find a point whose Tukey depth is at least $|S|/2(d+1)$ (provided that $|S|$ is large enough). Observe that $\mathbf{x}$ is in the convex hull of S if and only if $\mathrm{td}^S(\mathbf{x}) > 0$. As we mentioned in the introduction, finding a point whose Tukey depth is high results in a better learning algorithms for halfspaces.

The idea of our algorithm is to find the point $\mathbf{x} = (x_1, \ldots, x_d)$ coordinate after coordinate: we use $\mathcal{A}_{\text{RecConcave}}$ to find a value x_1^* that can be extended by some $x_2, \ldots, x_d$ so that the depth of $(x_1^*, x_2 \ldots, x_d)$ is close to the depth of the Tukey median, then we find a value x_2^* so that there is a point $(x_1^*, x_2^*, x_3 \ldots, x_d)$ whose depth is close to the depth of the Tukey median, and so forth until we find all coordinates. The parameters in $\mathcal{A}_{\text{RecConcave}}$ are set such that in each step we lose depth of at most $n/2(d+1)^2$ compared to the Tukey median, resulting in a point $(x_1^*, \ldots, x_d^*)$ whose depth is at most $n/2(d+1)$ less than the depth of the Tukey median, i.e., its depth is at least $n/2(d+1)$.

Defining a Quasi-Concave Function. To apply the above approach, we need to prove that the functions considered in the algorithm $\mathcal{A}_{\text{RecConcave}}$ are quasi-concave.

Definition 17 *Let $S \subseteq \mathbb{R}^d$ be a finite multiset. For every $1 \leq i \leq d$ and every $x_1^*, \ldots, x_{i-1}^* \in \mathbb{R}$, define $Q^S_{x_1^*,\ldots,x_{i-1}^*}(x_i) \triangleq \max_{x_{i+1},\ldots,x_d \in \mathbb{R}} \mathrm{td}^S(x_1^*, \ldots, x_{i-1}^*, x_i, \ldots, x_d)$.*

Lemma 18 *For every $1 \leq i \leq d$ and every $x_1^*, \ldots, x_{i-1}^* \in \mathbb{R}$, the function $Q^S_{x_1^*,\ldots,x_{i-1}^*}(x_i)$ is a quasi-concave function. Furthermore, $Q^S_{x_1^*,\ldots,x_{i-1}^*}(x_i)$ is a sensitivity 1 function of the multiset S.*

See the full version of this work for the proof.

Constructing Finite Domains. The input to the private algorithm for finding a point in the convex hull is a dataset of points $S \subseteq X$, where X is a finite set whose size is at most T. We note that the dataset S may contain several copies of the same point (i.e. it is a multiset). By the results of Bun et al. (2015) and Alon et al. (2018), the restriction to subsets of a finite set X is essential (even when $d = 1$). The algorithm performs d iterations, where in iteration i it computes the i'th coordinate of the output point.

The point outputted by the algorithm is an (approximate) Tukey Median with respect $S \subseteq X$. Note that a Tukey median might not be a point in X. Furthermore, the proof that the functions $Q^S_{x_1^*,\dots,x_{i-1}^*}$ are quasi-concave is over the reals. We construct finite domains so that the functions $Q^S_{x_1^*,\dots,x_{i-1}^*}$ attain their maximum over these finite domains. In each iteration, the algorithm, given $x_1^*,\dots,x_{i-1}^*$, constructs a new domain. We will not try to minimize the size of the domains as the dependency of the sample complexity of $\mathcal{A}_{\text{RecConcave}}$ on the size of the domain, denoted $\tilde{X}$, is $2^{O(\log^* |\tilde{X}|)}$. The next claim describes the extension in the i'th iteration.

Claim 19 *For every $1 \le i \le d$, and for every $x_1^*,\dots,x_{i-1}^* \in \mathbb{R}$ there exists sets $\tilde{X}_i,\dots,\tilde{X}_d$ such that $|\tilde{X}_j| \le |X|^{2d(d+1)}$ for $i \le j \le d$ and for every multiset $S \subseteq X$:*

- *There exist $x_i^m,\dots,x_d^m \in \tilde{X}_i \times \cdots \times \tilde{X}_d$ such that*

$$\max_{x_i,\dots,x_d\in\mathbb{R}} \operatorname{td}^S(x_1^*,\dots,x_{i-1}^*,x_i,\dots,x_d) = \operatorname{td}^S(x_1^*,\dots,x_{i-1}^*,x_i^m,\dots,x_d^m), \tag{4}$$

- *For every $x_i \in \tilde{X}_i$ there exist $x_{i+1}^m,\dots,x_d^m \in \tilde{X}_{i+1} \times \ldots \times \tilde{X}_d$ such that*

$$\max_{x_{i+1},\dots,x_d\in\mathbb{R}} \operatorname{td}^S(x_1^*,\dots,x_{i-1}^*,x_i,x_{i+1},\dots,x_d) = \operatorname{td}^S(x_1^*,\dots,x_{i-1}^*,x_i,x_{i+1}^m,\dots,x_d^m), \tag{5}$$

Proof The construction heavily exploits Theorem 16. Let L denote the set of all affine subspaces that are spanned by points in X. Since each such subspace is spanned by at most $d+1$ points, it follows that $|L| \le \binom{|X|}{d+1} \le |X|^{d+1}$. By Theorem 16, for every dataset S and every $r > 0$, every vertex of $\mathcal{T}(r)$ (the polytope of points of Tukey depth at least r) can be written as the intersection of at most d subspaces in L. In particular, there exists a Tukey median that is the intersection of at most d subspaces in L.

We construct the sets $\tilde{X}_i,\dots,\tilde{X}_d$ as follows. First, for every $d-i$ subspaces in L such that there exists a unique point $\mathbf{x} = (x_1^* \dots x_{i-1}^*, x_i \dots, x_d)$ in the intersection of these $d-i$ subspaces, we add x_j to $\tilde{X}_j$ for all $j \ge i$. We next argue that Equation (4) is satisfied: indeed, by Theorem 16, this construction contains a vertex of every set of the form $\mathcal{T}(r) \cap \{(x_1,\dots,x_d) \in \mathbb{R}^d : x_1 = x_1^*, \dots x_{i-1} = x_{i-1}^*\}$, for every $r > 0$. In particular, by plugging $r = \max_{x_{i+1},\dots,x_d\in\mathbb{R}} \operatorname{td}^S(x_1^*,\dots,x_{i-1}^*,x_i,\dots,x_d)$, it contains a point satisfying (4). Second, for every $x_i \in \tilde{X}_i$, we repeat the same process with respect to $x_1^* \dots x_{i-1}^*, x_i$ adding values to $\tilde{X}_j$ for all $j \ge i+1$; this ensures that Equation (5) is satisfied.

As explained above, the size of $\tilde{X}_i$ is at most $|X|^{d(d+1)}$ and the size of $\tilde{X}_j$ for every $j > i$ is at most $|\tilde{X}_i||X|^{d(d+1)} \le |X|^{2d(d+1)}$. ■

```
Algorithm $\mathcal{A}_{\mathrm{FindTukey}}$

Algorithm:
(i) Let $\beta, \varepsilon, \delta$ be the utility/privacy parameters, and $S$ be an input database from $X$.
(ii) For $i = 1$ to $d$ do:
    (a) Construct the sets $\tilde{X}_i, \ldots, \tilde{X}_d$ for $x_1^*, \ldots, x_{i-1}^*$ as in Theorem 19. Let $\tilde{T} = \max_{i \le j \le d} |\tilde{X}_j|$.
        (* By Theorem 19, $\log^* \tilde{T} = \log^* d + \log^* |X| + O(1)$. *)
    (b) For every $x_i \in \tilde{X}_i$ define
        $$Q^S_{x_1^*,\ldots,x_{i-1}^*}(x_i) \triangleq \max_{x_{i+1}\in\tilde{X}_{i+1},\ldots,x_d\in\tilde{X}_d} \mathrm{td}^S(x_1^*,\ldots,x_{i-1}^*,x_i,\ldots,x_d).$$
    (c) Execute $\mathcal{A}_{\mathrm{RecConcave}}$ on $S$ with the function $Q^S_{x_1^*,\ldots,x_{i-1}^*}$ and parameters $r = \frac{n}{d+1} - \frac{(i-1)n}{d(d+1)}$, $\alpha_0 = \frac{1}{2d}$, $\beta_0 = \frac{\beta}{d}$, $\varepsilon_0 = \frac{\varepsilon}{2\sqrt{2d\ln(2/\delta)}}$, $\delta_0 = \frac{\delta}{2d}$. Let $x_i^*$ be its output.
(iii) Return $x_1^*, \ldots, x_d^*$.
```

Figure 1: Algorithm $\mathcal{A}_{\mathrm{FindTukey}}$ for finding a point whose Tukey depth is at least $n/2(d+1)$.

The Algorithm. In Figure 1, we present an (ε,δ)-differentially private algorithm $\mathcal{A}_{\mathrm{FindTukey}}$ that with probability at least $1-\beta$ finds a point whose Tukey depth is at least $n/2(d+1)$. The informal description of the algorithm appears in the beginning of Section 3.

Theorem 20 *Let $\varepsilon \le 1$ and $\delta < 1/2$ and $X \subset \mathbb{R}^d$ be a set of size at most T. Assume that the input dataset $S \subseteq X$ satisfies $|S| = O\left(\frac{1}{\varepsilon} \cdot d^{2.5} \cdot 2^{O(\log^* T + \log^* d)} \log^{0.5}\left(\frac{1}{\delta}\right) \log\left(\frac{d}{\beta\delta}\right)\right)$. Then, $\mathcal{A}_{\mathrm{FindTukey}}$ is an (ε,δ)-differentially private algorithm that with probability at least $1-\beta$ returns a point $x_1^*, \ldots, x_d^*$ such that $\mathrm{td}^S(x_1^*,\ldots,x_d^*) \ge \frac{|S|}{2(d+1)}$.*

The correctness (utility) of $\mathcal{A}_{\mathrm{FindTukey}}$ is proved by induction, using the correctness of $\mathcal{A}_{\mathrm{RecConcave}}$. The privacy analysis follows from the privacy of $\mathcal{A}_{\mathrm{RecConcave}}$ and using composition properties of differential privacy. See the full version of this work for the details.

Running Time of the Algorithm. A straightforward implementation of $\mathcal{A}_{\mathrm{FindTukey}}$ results in an algorithm performing at most $|X|^{\mathrm{poly}(d)}$ arithmetic operations on elements of X. To see this, consider the ith iteration, and observe that Step (a) can be implemented in time $|X|^{\mathrm{poly}(d)}$ (as specified in the analysis of Theorem 19). For Step (c), note that in time $|X|^{\mathrm{poly}(d)}$ we can compute $Q^S_{x_1^*,\ldots,x_{i-1}^*}(x_i)$ for every $x_i \in \tilde{X}_i$. Once these are computed, the runtime of $\mathcal{A}_{\mathrm{RecConcave}}$ is at most $\mathrm{poly}(|\tilde{X}_i|)$. Overall, the algorithm an be implemented using at most $|X|^{\mathrm{poly}(d)}$ arithmetic operations.

4. Learning Halfspaces Using Convex Hull

We describe in Figure 2 a reduction from learning halfspaces to finding a point in a convex hull of a multiset of points. Furthermore, we show that if the algorithm we use in the reduction finds a point whose Tukey depth is high (as our algorithm from Section 3 does), then the required sample complexity of the learning algorithm is reduced. As a result, we get an upper bound of $\tilde{O}(d^{4.5}2^{\log^*|X|})$ on the sample complexity of private learning halfspaces (ignoring the privacy and learning parameters). In comparison, using the exponential mechanism of McSherry and Talwar (2007) results in a (ε, δ)-deferentially private algorithm whose sample complexity is $O(d\log|X|)$, e.g., for the interesting case where $X = [T]^d$ for some T, the complexity is $O(d^2 \log T)$. Our upper bound is better than the sample complexity of the exponential mechanism when d is small compared to $\log|T|$, in particular when d is constant.

Algorithm $\mathcal{A}_{\text{LearnHalfSpace}}$

Preprocessing:

- Fix a set $H \subseteq \mathbb{R}^{d+1}$ that contains representations of all halfspaces in $\texttt{HALFSPACE}(X)$.

Algorithm:

1. Let $\varepsilon, \delta, \alpha, \beta$ be the privacy and utility parameters and let S be a realizable input sample of size s, where s is as in Theorem 21.
2. Partition S into m equisized subsamples $S_1, \ldots, S_m$, where $m = m(d+1, 2|X|^{d+1}, \varepsilon, \delta, \beta/2)$ as in Theorem 21.
   ($*$ Note that each S_i has size $\Theta\big(\frac{d\log(\frac{m}{r\alpha})+\log(2m/\beta)}{r\alpha/m}\big)$. $*$)
3. For each S_i pick an arbitrary consistent halfpace $h_i \in H$.
4. Apply an (ε, δ)-differentially private algorithm $\mathcal{A}$ for finding a point in a convex hull with parameters $\varepsilon, \delta, \frac{\beta}{2}$ on $H_0 = (h_1 \ldots h_m)$.
5. Output the halfspace h found by $\mathcal{A}$.

Figure 2: A reduction from learning halfspaces to finding a point in a convex hull.

Theorem 21 *Assume that Algorithm $\mathcal{A}$ used in step 4 of Algorithm $\mathcal{A}_{\text{LearnHalfSpace}}$ is an (ε, δ)-differentialy private algorithm that finds with probability at least $1-\beta$ a point in a convex hull for a multisets $S \subseteq X \subset \mathbb{R}^d$ whose Tukey depth is at least r provided that $|S| \geq m(d, |X|, \varepsilon, \delta, \beta)$ for some function $m(\cdot,\cdot,\cdot,\cdot,\cdot)$.*

Let $\varepsilon \leq 1, \delta \leq \frac{1}{2}$ and $\alpha, \beta \leq 1$ be the privacy and utility parameters. Then, $\mathcal{A}_{\text{LearnHalfSpace}}$ is an (ε, δ)-differentially private (α, β)-PAC learner for $\texttt{HALFSPACE}(X)$ with sample complexity $s = O\Big(\frac{m^2(d\log(\frac{m}{r\alpha})+\log(m/\beta))}{r\alpha}\Big)$ where $m = m(d+1, 2|X|^{d+1}, \varepsilon, \delta, \beta/2)$.

Using $\mathcal{A}_{\text{FindTukey}}$ in step 4 of Algorithm $\mathcal{A}_{\text{LearnHalfSpace}}$, we get the following corollary (which follows from Theorem 20 and Theorem 21).

Corollary 22 *Let $\varepsilon \leq 1$, $\delta < 1/2$, and $X \subseteq \mathbb{R}^d$ be a set. There exists an (ε, δ)-differentially private (α, β)-PAC learner with sample complexity s for $\texttt{HALFSPACE}(X)$ with*

$$s = \tilde{O}\Big(\frac{d^{4.5} 2^{O(\log^* |X| + \log^* d)} \log^{1.5} \frac{1}{\delta} \log^2 \frac{1}{\beta}}{\varepsilon\alpha}\Big).$$

Corollary 22 establishes an upper bound on the sample complexity of privately learning halfspaces whose dependency on the domain size $|X|$ is $2^{O(\log^*(|X|))}$. The crux of the algorithm is a reduction to privately publishing a point with a large Tukey depth with respect to a given input dataset. A drawback of this approach is that the latter task is likely to be computationally difficult (even without privacy constraints), unless the dimension d is constant (see Miller and Sheehy (2010) and references within).

In $\mathcal{A}_{\text{LearnHalfSpace}}$ we can use an algorithm $\mathcal{A}$ that finds a point in the convex hull (i.e., a point whose Tukey depth is at least 1). The resulting learning algorithm require sample complexity of $O(\frac{m^2 \cdot d\log(\frac{m}{\alpha})+\log(m/\beta)}{\alpha})$, where m is the sample complexity of $\mathcal{A}$. This may result in a more efficient private learning algorithm for halfspaces as the task of privately finding a point in the convex hull might be easier than the task of privately finding a point with high Tukey depth. Furthermore, in this case, we can use an algorithm that privately finds a hypothesis that is a linear combination with positive coefficients of the hypotheses in H_0. This follows from the observation that if all hypotheses in H_0 are correct on a point $\mathbf{x}$, then any linear combination with positive coefficients of the hypotheses in H_0 is correct on $\mathbf{x}$.

Private agnostic-PAC learner for halfspaces. Instantiating the transformation of Beimel et al. (2015) for the agnostic case (see Theorem 8) with our learner from Corollary 22, we obtain a private agnostic learner for halfspaces.

Corollary 23 *Let $\varepsilon \leq 1$, $\delta < 1/2$, and $X \subseteq \mathbb{R}^d$ be a set. There exists an (ε, δ)-differentially private (α, β)-*agnostic *PAC learner with sample complexity s for $\texttt{HALFSPACE}(X)$ with*

$$s = \tilde{O}\Big(\frac{d^{4.5} 2^{O(\log^* |X| + \log^* d)} \log^{1.5} \frac{1}{\delta} \log^2 \frac{1}{\beta}}{\varepsilon\alpha} + \frac{d\log(\frac{1}{\alpha\beta})}{\varepsilon\alpha^2}\Big).$$

5. A Lower Bound on the Sample Complexity of Privately Finding a Point in the Convex Hull

In this section we show a lower bound on the sample complexity of privately finding a point in the convex hull of a database $S \subseteq X = [T]^d$. We show that any (ε, δ)-differentially private algorithm for this task must have sample complexity $\Omega(\frac{d}{\varepsilon}\log\frac{1}{\delta})$. Our lower bound actually applies to a possibly simpler task of finding a non-trivial linear combination of the points in the database.

By Bun et al. (2015), finding a point in the convex hull (even for $d = 1$) requires sample complexity $\Omega(\log^* T)$. Thus, together we get a lower bound on the sample complexity of $\Omega(\frac{d}{\varepsilon}\log\frac{1}{\delta} + \log^* T)$.

It may be tempting to guess that, even with pure $(\varepsilon, 0)$-differential privacy, a sample complexity of $O(\log|X|) = O(d \log T)$ should suffice for solving this task, as the size of the output space is T^d, because $S \subseteq [T]^d$, and hence (it seems) that one could privately solve this problem using the exponential mechanism of McSherry and Talwar (2007) with sample complexity that depends logarithmically on the size of the output space. We show that this is not the case, and that any $(\varepsilon, 0)$-differentially private algorithm for this task must have sample complexity $\Omega(\frac{d^2}{\varepsilon}\log T)$.

Theorem 24 *Let $T \geq 2$, and $d \geq 10$. Let $\mathcal{A}$ be an (ε, δ)-differentially private algorithm that takes a database $S \subseteq [T]^d$ of size m and returns, with probability at least $1/2$, a non-trivial linear combination of the points in S. Then, $m = \Omega\left(\min\left\{\frac{d^2}{\varepsilon}\log T,\ \frac{d}{\varepsilon}\log\frac{1}{\delta}\right\}\right)$.*

The proof of Theorem 24 builds on the analysis of Blum et al. (2008) for lower bounding the sample complexity of releasing approximate answers for counting queries. See the full version of this work for more details.

Acknowledgments

A. B. and K. N. were supported by NSF grant no. 1565387. TWC: Large: Collaborative: Computing Over Distributed Sensitive Data. A. B. was supported by ISF grant no. 152/17. Work done while A. B. was visiting Georgetown University. S. M. was supported by the Simons Foundation and the NSF; part of this project was carried while Shay was at the Institute for Advanced Study and was supported by the National Science Foundation under agreement No. CCF-1412958. U. S. was supported by a gift from Google Ltd.

References

Noga Alon, Roi Livni, Maryanthe Malliaris, and Shay Moran. Private PAC learning implies finite littlestone dimension. *CoRR*, abs/1806.00949, 2018. URL http://arxiv.org/abs/1806.00949. To appear at STOC'19.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473, 2014. URL http://dx.doi.org/10.1109/FOCS.2014.56.

Raef Bassily, Abhradeep Guha Thakurta, and Om Dipakbhai Thakkar. Model-agnostic private learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 7102–7112, 2018. URL http://papers.nips.cc/paper/7941-model-agnostic-private-learning.

Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of private learners. In *ITCS*, pages 97–110. ACM, 2013a.

Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *APPROX-RANDOM*, volume 8096 of *Lecture Notes in Computer*

Science, pages 363–378. Springer, 2013b. Journal version:*Theory of Computing*, 12(1):1–61, 2016.

Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine Learning*, 94(3):401–437, 2014.

Amos Beimel, Kobbi Nissim, and Uri Stemmer. Learning privately with labeled and unlabeled examples. In *SODA*, pages 461–477. SIAM, 2015.

Shai Ben-David and Ami Litman. Combinatorial variability of vapnik-chervonenkis classes with applications to sample compression schemes. *Discrete Applied Mathematics*, 86(1):3–25, 1998. URL https://doi.org/10.1016/S0166-218X(98)00000-6.

Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In Chen Li, editor, *PODS*, pages 128–138. ACM, 2005.

Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618. ACM, 2008.

Mark Bun. *New Separations in the Complexity of Differential Privacy*. PhD thesis, Harvard University, 2016. Supervisor-Salil Vadhan.

Mark Bun and Mark Zhandry. Order-revealing encryption and the hardness of private learning. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A*, volume 9562 of *Lecture Notes in Computer Science*, pages 176–206. Springer, 2016. URL https://doi.org/10.1007/978-3-662-49096-9_8.

Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.

Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In *ITCS*, pages 369–380. ACM, 2016.

Kamalika Chaudhuri and Daniel Hsu. Sample complexity bounds for differentially private learning. In Sham M. Kakade and Ulrike von Luxburg, editors, *COLT*, volume 19 of *JMLR Proceedings*, pages 155–186. JMLR.org, 2011.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, Jul 2008. ISSN 1436-4646.

Cynthia Dwork and Vitaly Feldman. Privacy-preserving prediction. In *COLT 2018*, pages 1693–1702, 2018. URL http://proceedings.mlr.press/v75/dwork18a.html.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin, Heidelberg, 1987. ISBN 0-387-13722-X.

Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. *SIAM J. Comput.*, 44(6):1740–1764, 2015. URL http://dx.doi.org/10.1137/140991844.

Justin Hsu, Aaron Roth, Tim Roughgarden, and Jonathan Ullman. Privately solving linear programs. In *ICALP*, pages 612–624, 2014. URL https://doi.org/10.1007/978-3-662-43948-7_51.

Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Differentially private learning of geometric concepts. *CoRR*, abs/1902.05017, 2019. URL http://arxiv.org/abs/1902.05017.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011. URL https://doi.org/10.1137/090756090.

Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.

Xiaohui Liu, Karl Mosler, and Pavlo Mozharovskyi. Fast computation of Tukey trimmed regions and median in dimension $p > 2$. *arXiv e-prints*, arXiv:1412.5122:arXiv:1412.5122, 2014.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

Gary L. Miller and Donald R. Sheehy. Approximate centerpoints with proofs. *Comput. Geom.*, 43 (8):647–654, 2010. URL https://doi.org/10.1016/j.comgeo.2010.04.006.

Huy L. Nguyen, Jonathan Ullman, and Lydia Zakynthinou. Efficient private algorithms for learning halfspaces. *CoRR*, abs/1902.09009, 2019. URL http://arxiv.org/abs/1902.09009.

John W. Tukey. Mathematics and the picturing of data. In *Proc. Int. Congress of Mathematicians*, volume 2, pages 523–532, 1975.

Isaac Moiseevich Yaglom and Vladimir Grigorevich Boltyanskiĭ. *Convex figures*. Holt, Rinehart and Winston, 1961.