Differentially Private Learning of Geometric Concepts

Haim Kaplan [1 2] **Yishay Mansour** [1 2] **Yossi Matias** [2] **Uri Stemmer** [3 4]

Abstract

We present differentially private efficient algorithms for learning union of polygons in the plane (which are not necessarily convex). Our algorithms achieve (α, β)-PAC learning and (ε, δ)-differential privacy using a sample of size $\tilde{O}\left(\frac{1}{\alpha\varepsilon} k \log d\right)$, where the domain is $[d] \times [d]$ and k is the number of edges in the union of polygons.

1. Introduction

Machine learning algorithms have exciting and wide-range potential. However, as the data frequently contain sensitive personal information, there are real privacy concerns associated with the development and the deployment of this technology. Motivated by this observation, the line of work on *differentially private learning* (initiated by Kasiviswanathan et al. (2011)) aims to construct learning algorithms that provide strong (mathematically proven) privacy protections for the training data. Both government agencies and industrial companies have realized the importance of introducing strong privacy protection to statistical and machine learning tasks. A few recent examples include Google (Erlingsson et al., 2014) and Apple (Thakurta et al., 2017) that are already using differentially private estimation algorithms that feed into machine learning algorithms, and the US Census Bureau announcement that they will use differentially private data publication techniques in the next decennial census (Abowd, 2016). Differential privacy is increasingly accepted as a standard for rigorous privacy. We refer the reader to the excellent surveys in (Dwork & Roth, 2014) and (Vadhan, 2016). The definition of differential privacy is,

Definition 1.1 (Dwork et al. (2006)). *Let $\mathcal{A}$ be a randomized algorithm whose input is a sample. Algorithm $\mathcal{A}$ is* (ε, δ)-differentially private *if for every two samples S, S' that differ in one example, and for any event T, we have*

$$\Pr[\mathcal{A}(S) \in T] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in T] + \delta.$$

For now, we can think of a (non-private) learner as an algorithm that operates on a set of classified random examples, and outputs a hypothesis h that misclassifies fresh examples with probability at most (say) $\frac{1}{10}$. A *private learner* must achieve the same goal while guaranteeing that the choice of h preserves differential privacy of the sample points. Intuitively this means that the choice of h should not be significantly affected by any particular sample.

While many learning tasks of interest are compatible with differential privacy, privacy comes with a cost (in terms of computational resources and the amount of data needed), and it is important to understand how efficient can private learning be. Indeed, there has been a significant amount of work aimed at understanding the sample complexity of private learning (Beimel et al., 2010; Chaudhuri & Hsu, 2011; Beimel et al., 2013a;b; Feldman & Xiao, 2015; Bun et al., 2015; Alon et al., 2018; Beimel et al., 2019), the computational complexity of private learning (Bun & Zhandry, 2016), and studying variations of the private learning model (Beimel et al., 2015; Bun et al., 2016; Dwork & Feldman, 2018; Bassily et al., 2018). However, in spite of the significant progress made in recent years, much remains unknown and answers to fundamental questions are still missing. In particular, the literature lacks effective constructions of private learners for specific concept classes of interest, such as halfspaces, polynomials, d-dimensional balls, and more. We remark that, in principle, every (non-private) learner that works in the *statistical queries (SQ)* model of Kearns (1998) can be transformed to preserve differential privacy. However, as the transformation is only tight up to polynomial factors, and as SQ learners are often much less efficient than their PAC learners counterparts, the resulting private learners are typically far from practical.

In this work we make an important step towards bringing differentially private learning closer to practice, and construct an effective algorithm for privately learning simple geometric shapes, focusing on polygons in the plane. To motivate our work, consider the task of analyzing GPS navigation data, or the task of learning the shape of a flood or a fire based on users' location reports. As user location data might be sensitive, the ability to *privately learn* such shapes in the plane is of significant importance.

[1]Tel Aviv University [2]Google research, Israel [3]Ben-Gurion University [4]Supported by a gift from Google Ltd. Correspondence to: Uri Stemmer <u@uri.co.il>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

1.1. A Non-Private Learner for Conjunctions and Existing Techniques

Our learner is obtained by designing a private variant for the classical (non-private) learner for conjunctions using the greedy set-cover algorithm. Before describing our new learner, we first quickly recall this non-private technique (see e.g., (Kearns & Vazirani, 1994) for more details).

Let $\mathrm{CONJ}_{k,d}$ denote the class of all conjunctions (i.e., AND) of at most k literals over d Boolean variables $v_1, \dots, v_d$, e.g., $v_1 \wedge \overline{v}_4 \wedge v_5$. Here, for a labeled example $(\vec{x}, \sigma)$, the vector $\vec{x} \in \{0,1\}^d$ is interpreted as an assignment to the d Boolean variables, and $\sigma = 1$ iff this assignment satisfies the target concept. Given a sample $S = \{(\vec{x}_i, \sigma_i)\}$ of labeled examples, the classical (non-private) learner for this class begins with the hypothesis $h = v_1 \wedge \overline{v}_1 \wedge \cdots \wedge v_d \wedge \overline{v}_d$, and then proceeds by deleting from h any literal that "contradicts" a positively labeled example in S. Observe that at the end of this process, the set of literals appearing in h contains the set of literals appearing in the target concept (because a literal is only deleted when it is contradicted by a positively labeled example).

The next step is to eliminate unnecessary literals from the hypothesis h (in order to guarantee generalization). Note that removing literals from h might cause it to err on negative example in S, and hence, the algorithm must carefully choose which of the literals to eliminate. This can be done using the greedy algorithm for set cover as follows. We have already made sure that each of the literals in h does not err on positive examples in S (since such literals were deleted), and we know that there is a choice of k literals from h that together correctly classify all negative examples in S (since we know that the k literals of the target concept are contained in h). Thus, our task can be restated as identifying a small number of literals from h that together correctly classify all negative examples in S. This can be done using the greedy algorithm for set cover, where every literal in h corresponds to a set, and this set "covers" a negative example if the literal is zero on this example.

To summarize, the algorithm first identifies the collection of all literals that are consistent with the positive data, and then uses the greedy algorithm for set cover in order to identify a small subset of these literals that together correctly classify the negative data. This is a good starting point for designing a private learner for conjunctions, since the greedy algorithm for set cover has a private variant (Gupta et al., 2010). The challenge here is that in the private algorithm of Gupta et al. (2010), the collection of sets from which the cover is chosen is assumed to be fixed and independent of the input data. In contrast, in our case the collection of sets corresponds to the literals that correctly classified the positive data, which is data dependent. One might try to overcome this challenge by first identifying, in a private manner, a collection L of all literals that correctly classify (most of) the positive data, and then to run the private algorithm of Gupta et al. (2010) to choose a small number of literals from L. However, a direct implementation of this idea would require accounting for the privacy loss incurred due to each literal in L. As $|L|$ can be linear in d (the number of possible literals), this would result in an algorithm with sample complexity $\mathrm{poly}(d)$. When we apply this strategy to learn polygons in the plane, d will correspond to the size of an assumed grid on the plane, which we think of as being very big, e.g., $d = 2^{64}$. Hence, $\mathrm{poly}(d)$ sample complexity is unacceptable.

1.2. Our Results

Our first result is an efficient private learner for conjunctions. Our learner is obtained by modifying the strategy outlined above to use the greedy algorithm for set cover in order to choose a small number of literals directly out of the set of all possible $2d$ literals (instead of choosing them out of the set of literals that agree with the positive examples). However, this must be done carefully, as unlike before, we need to ensure that the selected literals will not cause our hypothesis to err on the positive examples. Specifically, in every step of the greedy algorithm we will aim at choosing a literal that eliminates (i.e., evaluates to zero on) a lot of the negative examples without eliminating (essentially) any of the positive examples. In the terminology of the set cover problem, we will have *two* types of elements – positive and negative elements – and our goal will be to identify a small cover of the negative examples that does not cover (essentially) any of the positive examples. We show,

Theorem 1.2. *There exists an efficient (ε, δ)-differentially private (α, β)-PAC learner for $\mathrm{CONJ}_{k,d}$ with sample complexity*[1] $\tilde{O}(\frac{1}{\alpha\varepsilon} k \log d)$.

We remark that our technique extends to disjunctions of literals in a straightforward way, as follows.

Theorem 1.3. *There exists an efficient (ε, δ)-differentially private (α, β)-PAC learner for the class of all disjunctions (i.e., OR) of at most k literals over d Boolean variables with sample complexity* $\tilde{O}(\frac{1}{\alpha\varepsilon} k \log d)$.

We then show that our technique can be used to privately learn (not necessarily convex) polygons in the plane. To see the connection, let us first consider *convex* polygons, and observe that a *convex* polygon with k edges can be represented as the intersection of k halfplanes. Thus, as in our learner for Boolean conjunctions, if we could identify a halfplane that eliminates (i.e., evaluates to zero on) a lot of the negative examples without eliminating (essentially) any of the positive examples in the sample, then we could

[1]For simplicity we used the $\tilde{O}$ notation to hide logarithmic factors in $\alpha, \beta, \varepsilon, \delta, k$. The dependency in these factors wil be made explicit in the following sections.

learn convex polygons in iterations. However, recall that in our learner for conjunctions, the parameter d controlled both the description length of examples (since each example specified an assignment to d variables) and the number of possible literals (which was $2d$). Thus, the running time of our algorithm was allowed to be linear in the number of possible literals. The main challenge when applying this technique to (convex) polygons is that the number of possible halfplanes (which will correspond to the parameter d) is huge, and our algorithm cannot run in time linear in the number of possible halfplanes.

To recover from this difficulty, we consider the dual plane in which sample points correspond to lines, and show that in that dual plane it is possible to (privately) identify a point (that corresponds to a halfplane in the primal plane) with the required properties (that is, eliminating a lot of negative examples while not eliminating positive examples). The idea is that since there are only n input points, in the dual plane there will be only n lines to consider, which partition the dual plane into at most n^2 regions. Now, two different halfplanes in the primal plane correspond to two different points in the dual plane, and if these two points fall in the same region, then the two halfplanes behave identically on the input sample. We could therefore partition the halfplanes (in the primal plane) into at most n^2 equivalence classes w.r.t. the way they behave on the input sample. This fact can be leveraged in order to efficiently implement the algorithm.

Our techniques extend to non-convex polygons, which unlike convex polygons cannot be represented as intersection of halfplanes. It it well known that every (simple[2]) polygon with k edges can be represented as the union of at most k triangles, each of which can be represented as the intersection of at most 3 halfplanes (as a triangle is a convex polygon with 3 edges). In other words, a (simple) polygon with k edges can be represented as a DNF formula (i.e., disjunction of conjunctive clauses) in which each clause has at most 3 literals. As we will see, our techniques can be extended to capture this case efficiently. Our main theorem is the following.

Theorem 1.4 (informal)**.** *There exists an efficient (ε, δ)-differentially private (α, β)-PAC learner for union of (simple) polygons in the plane with sample complexity $\tilde{O}(\frac{1}{\alpha\varepsilon} k \log d)$, where k is the number of edges in the union of polygons and $\log d$ is the description length of examples.*

As the greedy algorithm for set cover has many applications in computational learning theory, we hope that our techniques will continue to find much broader use.

Remark 1.5. *Any informalities made hereafter are removed in the a full version of the paper, available at `https://arxiv.org/abs/1902.05017`.*

[2]A *simple* polygon is one which does not intersect itself.

2. Preliminaries

In the following X is some arbitrary domain. A concept (similarly, hypothesis) over domain X is a predicate defined over X. A concept class (similarly, hypothesis class) is a set of concepts. Two labeled databases $S, S' \in (X \times \{0,1\})^n$ are called *neighboring* if they differ on a single (labeled) entry. A function $f : X^* \to \mathbb{R}$ has *sensitivity* s if for every neighboring S, S', it holds that $|f(S) - f(S')| \leq s$.

Definition 2.1. *Let H be a concept class over a domain X, and let $k \in \mathbb{N}$. We use $H^{\vee k}$ to denote the class of all disjunctions (i.e., OR) of at most k concepts from H, and similarly, we denote $H^{\wedge k}$ for the class of all conjunctions (i.e., AND) of at most k concepts from H.*

Without privacy considerations, the sample complexity of PAC learning a class C is essentially characterized by a combinatorial quantity called the *Vapnik-Chervonenkis* dimension of C, denoted as $\mathrm{VC}(C)$. For conjunctions and disjunctions of a class H we have:

Observation 2.2. *For every concept class H we have $\mathrm{VC}(H^{\vee k}) \leq O(k \log(k) \cdot \mathrm{VC}(H))$ and $\mathrm{VC}(H^{\wedge k}) \leq O(k \log(k) \cdot \mathrm{VC}(H))$.*

Observation 2.2 is standard (see, e.g., (Eisenstat & Angluin, 2007)). Our strategy in the following sections for privately learning a concept class C will be to use a "simpler" concept class H such that for some (hopefully small) k we have $C \subseteq H^{\vee k}$ or $C \subseteq H^{\wedge k}$.

Example 2.3. *Let $\mathrm{DISJ}_{k,d}$ denote the class of all disjunctions (i.e., OR) of at most k literals over d Boolean variables, and similarly, let $\mathrm{CONJ}_{k,d}$ denote the class of all conjunctions (i.e., AND) of at most k literals over d Boolean variables. Trivially, $\mathrm{DISJ}_{k,d} = (\mathrm{DISJ}_{1,d})^{\vee k}$, and $\mathrm{CONJ}_{k,d} = (\mathrm{CONJ}_{1,d})^{\wedge k}$.*

The most basic constructions of differentially private algorithms are via the Laplace mechanism as follows.

Theorem 2.4 (The Laplacian Mechanism (Dwork et al., 2006))**.** *A random variable has probability distribution $\mathrm{Lap}(b)$ if its probability density function is $f(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$, where $x \in \mathbb{R}$. Let $f : X^* \to \mathbb{R}$ be a sensitivity s function. The mechanism $\mathcal{A}$ that on input $S \in X^*$ adds noise with distribution $\mathrm{Lap}(\frac{s}{\varepsilon})$ to the output of $f(S)$ preserves ε-differential privacy.*

We next describe the exponential mechanism of McSherry & Talwar (2007). Let X be a domain and H a set of solutions (in our context, H will be a set of hypotheses). Given a quality function $q : X^* \times H \to \mathbb{N}$ with sensitivity 1, and a database $S \in X^*$, the goal is to chooses a solution $h \in H$ approximately maximizing $q(S, h)$. The mechanism chooses a solution $h \in H$ with probability proportional to $\exp(\varepsilon \cdot q(S, f)/2)$.

Proposition 2.5 (Properties of the Exponential Mechanism)**.** *(i) The exponential mechanism is ε-differentially private. (ii) Let $\hat{e} \triangleq \max_{f\in H}\{q(S,f)\}$ and $\lambda > 0$. The exponential mechanism outputs a solution h such that $q(S,h) \le \hat{e} - \lambda$ with probability at most $|H| \cdot \exp(-\varepsilon\lambda/2)$.*

3. A Generic Construction via Set Cover

In this section we present our generic construction for privately learning a concept class C containing concepts that can be written as the conjunction or the disjunction of functions in a (hopefully simpler) class H. For readability we focus on conjunctions. The extension to disjunction is straightforward.

Claim 3.1. *Fix a target function $c^* \in C$, and consider the execution of `SetCoverLearner` on a sample $S = \{(x_i, c^*(x_i))\}_{i=1}^n$. Assume that every run of the selection procedure $\mathcal{A}$ in Step 1e returns a hypothesis h_j s.t. $q(h_j) \ge \max_{f\in H}\{q(f)\} - \lambda$. Then, with probability at least $1-\beta$ it holds that h_{fin} errs on at most $\max\left\{\frac{\alpha n}{2},\ \frac{8k}{\varepsilon}\log\left(\frac{2}{\alpha}\right)\log\left(\frac{2k}{\beta}\log\frac{2}{\alpha}\right)\right\} + 4k\lambda\log\frac{2}{\alpha}$ example in S.*

Proof. First observe that there are $2k\log\frac{2}{\alpha}$ draws from $\mathrm{Lap}\left(\frac{2k}{\varepsilon}\log\frac{2}{\alpha}\right)$ throughout the execution. By the properties of the Laplace distribution, with probability at least $1-\beta$ it holds that the maximum absolute value of these random variables is at most $\Delta = \frac{2k}{\varepsilon}\log\left(\frac{2}{\alpha}\right)\log\left(\frac{2k}{\beta}\log\frac{2}{\alpha}\right)$. We continue with the analysis assuming that this is the case. In particular, this means that in every iteration j we have $|S^0| - 2\Delta \le b_j \le |S^0|$. Thus, in every iteration there exists a hypothesis $\tilde{h} \in H$ with $q(\tilde{h}) \ge 0$. To see this, recall that the target concept c^* can be written as $c^* = h_1^* \wedge \cdots \wedge h_k^*$ for $h_1^*, \dots, h_k^* \in H$. Hence, in every iteration j there is a hypothesis $\tilde{h} \in H$ that correctly classifies *all* of the (remaining) positive points in S while correctly classifying at least $1/k$ fraction of the (remaining) negative points in S, i.e., at least $|S^0|/k \ge b_j/k$. Such a hypothesis $\tilde{h}$ satisfies $q(\tilde{h}) = 0$. By our assumption on the selection procedure $\mathcal{A}$, we therefore have that in each iteration j, the selection procedure identifies a hypothesis h_j s.t. $q(h_j) \ge -\lambda$.

By the definition of q, in every iteration j we have that the selected h_j misclassifies at most λ of the remaining positive examples in S. Therefore, h_{fin} misclassifies at most $2k\lambda\log\frac{2}{\alpha}$ positive examples in S. Moreover, in every iteration j s.t. $|S^0| \ge 2k\lambda + 4\Delta$ we have that h_j classifies correctly at least $\frac{1}{2k}$ fraction of the negative examples in S. To see this, observe that as $q(h_j) \ge -\lambda$ we have

$$\#_{h_j\to 0}(S^0) \ge b_j/k - \lambda \ge \frac{|S^0| - 2\Delta}{k} - \lambda \ge \frac{|S^0|}{2k}.$$

That is, either there exists an iteration j in which number of negative points in S drops below $2k\lambda + 4\Delta$, or every iteration shrinks the number of negative examples by a factor of $\frac{1}{2k}$, in which case after $2k\log\frac{2}{\alpha}$ iterations there could be at most $\frac{\alpha n}{2}$ negative points in S. Observe that h_{fin} does not err on negative points that were removed from S, and therefore, there could be at most $\max\left\{\frac{\alpha n}{2},\ 2k\lambda + 4\Delta\right\}$ negative points on which h_{fin} errs. Overall, h_{fin} errs on at most $\max\left\{\frac{\alpha n}{2},\ 4\Delta\right\} + 4k\lambda\log\frac{2}{\alpha}$ points in S. $\square$

Claim 3.1 ensures that if at every step $\mathcal{A}$ picks h_j of high quality, then (w.h.p.) algorithm `SetCoverLearner` returns a hypothesis from $H^{\wedge 2k\log\frac{2}{\alpha}}$ with low empirical error. Combining this with standard generalization bounds and with Observation 2.2 (that bounds the VC dimension of $H^{\wedge 2k\log\frac{2}{\alpha}}$) we get the following theorem.

Theorem 3.2. *Let C, H, k be two concept classes and an integer such that $C \subseteq H^{\wedge k}$. Let $\mathcal{A}$ be a selection procedure that takes a database S and a quality function q, and returns a hypothesis $h \in H$ such that $q(h_j) \ge \max_{f\in H}\{q(f)\} - \lambda$ with probability at least $1 - \frac{\beta}{4k\log(2/\alpha)}$. Then, algorithm `SetCoverLearner` with $\mathcal{A}$ as the selection procedure is an (α,β)-PAC learner for C with sample complexity*
$$n = \Theta\left(\frac{k\log\frac{1}{\alpha}}{\alpha}\left(\mathrm{VC}(H)\log(k) + \lambda + \frac{1}{\varepsilon}\log\left(\frac{k}{\beta}\log\frac{1}{\alpha}\right)\right)\right).$$

Tuning the selection procedure. If H is finite, then one could directly implement the selection procedure $\mathcal{A}$ using the exponential mechanism of McSherry & Talwar (2007) to find a hypothesis h_j with large $q(h_j)$ at each iteration. In order to guarantee that all of the $\approx k$ iterations of algorithm `SetCoverLearner` satisfy together (ε,δ)-differential privacy, it suffices that each application of the exponential mechanism satisfies $\hat{\varepsilon} \approx \frac{\varepsilon}{\sqrt{k}}$-differential privacy (see, e.g., (Dwork et al., 2010) for composition theorems for differential privacy). When choosing such an $\hat{\varepsilon}$, the exponential mechanism identifies, in every iteration, an h_j such that $q(j) \gtrsim \max_{f\in H}\{q(f)\} - \frac{1}{\hat{\varepsilon}}\log|H| \approx \max_{f\in H}\{q(f)\} - \frac{\sqrt{k}}{\varepsilon}\log|H|$. This gives a selection procedure $\mathcal{A}$ which selects h_j with $q(h_j) \ge \max_{f\in H}\{q(f)\} - \lambda$, for $\lambda \approx \frac{\sqrt{k}}{\varepsilon}\log|H|$.

Example 3.3. *There exist efficient (ε,δ)-differentially private (α,β)-PAC learners for $\mathrm{CONJ}_{k,d}$ and for $\mathrm{DISJ}_{k,d}$ with sample complexity $n = \tilde{\Theta}\left(\frac{1}{\alpha\varepsilon} \cdot k^{1.5}\log d\right)$.*

The reason for the dependency in $k^{1.5}$ in the above example, is that for the privacy analysis we wanted to ensure that each iteration was differentially private with parameter $\approx \varepsilon/\sqrt{k}$ (because when composing ℓ differentially private mechanisms the privacy budget deteriorates proportionally to $\sqrt{\ell}$). This resulted in $\approx \sqrt{k}/\varepsilon$ misclassified points per iteration (and there are $\approx k$ iterations). As we next explain, in our case, the privacy parameter does not need to deteriorate with the number of iterations, which allows us to improve the sample complexity by a $\sqrt{k}$ factor. Our approach to proving

Algorithm `SetCoverLearner`

Settings: Concept classes C, H and an integer $k \in \mathbb{N}$ such that $C \subseteq H^{\wedge k}$.

Input: Labeled sample $S = \{(x_i, \sigma_i)\}_{i=1}^n \in (X \times \{0,1\})^n$, privacy parameter ε.

Tool used: A selection procedure $\mathcal{A}$ that takes a database S and a quality function q (that assigns a score to each hypothesis in H), and returns a hypothesis $h \in H$.

```
1. For $j = 1$ to $2k \log \frac{2}{\alpha}$
    (a) Let $S^1$ and $S^0$ denote the set of positive and negative examples in $S$, respectively.
    (b) For $h \in H$ let $\#_{h\to 0}(S^1)$ and $\#_{h\to 0}(S^0)$ denote the number of positive and negative examples in $S$, respectively,
        that $h$ labels as 0. That is, $\#_{h\to 0}(S^1) = |\{x_i \in S^1 : h(x_i) = 0\}|$ and $\#_{h\to 0}(S^0) = |\{x_i \in S^0 : h(x_i) = 0\}|$.
    (c) Let $w_j \leftarrow \left\lfloor \mathrm{Lap}\left(\frac{2k}{\varepsilon} \log \frac{2}{\alpha}\right) \right\rfloor$ and set $b_j = |S^0| + w_j - \frac{2k}{\varepsilon} \log\left(\frac{2}{\alpha}\right) \log\left(\frac{2k}{\beta} \log \frac{2}{\alpha}\right)$.
    (d) For every $h \in H$, define $q(h) = \min\left\{ \#_{h\to 0}(S^0) - \frac{b_j}{k} \ , \ -\#_{h\to 0}(S^1) \right\}$.
    (e) Let $h_j \leftarrow \mathcal{A}(S, q)$, and delete from $S$ every $(x_i, \sigma_i)$ such that $h_j(x_i) = 0$.
2. Return the hypothesis $h_{fin} = h_1 \wedge h_2 \wedge \cdots \wedge h_{2k \log \frac{2}{\alpha}}$.
```

this improved bound builds on the analysis of Gupta et al. (2010) for their private set cover algorithm. The main difference is that we have both positive and negative examples, which we need to handle differently. As we next explain, this will be achieved using Step 1c of `SetCoverLearner`.

Claim 3.4. *Let $\varepsilon \in (0,1)$ and $\delta < 1/e$. Instantiating `SetCoverLearner` with the exponential mechanism as the selection procedure $\mathcal{A}$ with privacy parameter $\hat{\varepsilon} = \frac{\varepsilon}{2\ln(e/\delta)}$ (per iteration) satisfies (ε, δ)-differential privacy.*

We next present an intuitive (and oversimplified) overview of the proof. Consider two neighboring databases S and S' such that $S' = S \cup \{(x^*, \sigma^*)\}$, and let us focus here on the case where $\sigma^* = 0$. Fix a possible output $\vec{h} = (h_1, h_2, \ldots, h_{2k \log \frac{2}{\alpha}})$ of `SetCoverLearner`. We will analyze the ratio

$$\frac{\Pr[\texttt{SetCoverLearner}(S) = \vec{h}]}{\Pr[\texttt{SetCoverLearner}(S') = \vec{h}]}. \qquad (1)$$

Let t be such that h_t is the first hypothesis in this output vector satisfying $h_t(x^*) = 0$. Observe that after the tth iteration, the executions on S and on S' continue exactly the same, since (x^*, σ^*) is removed from S' during the tth iteration (because in every iteration we remove all input elements on which the selected hypothesis evaluates to 0). Intuitively, if t is small then we only need to pay (in the privacy analysis) for a small number of iterations. In general, however, t might be as large as $2k \log \frac{2}{\alpha}$, and accounting for that many iterations in the privacy analysis is exactly what we are trying to avoid.

Recall that each iteration j of `SetCoverLearner` draws a random noise w_j from $\left\lfloor \mathrm{Lap}\left(\frac{2k}{\varepsilon} \log \frac{2}{\alpha}\right) \right\rfloor$. Let us denote these noises as they are in the execution on S as $\vec{w} = (w_1, \ldots, w_{2k \log \frac{2}{\alpha}})$ and in the execution on S' as $\vec{w}' = (w'_1, \ldots, w'_{2k \log \frac{2}{\alpha}})$. Furthermore, let us assume that $w'_j = w_j - 1$ for every $j \le t$ and that $w'_j = w_j$ for every $j > t$. By the properties of the Laplace distribution, this assumption distorts our bound on the ratio in expression (1) by at most an e^{ε} factor. (In a sense, for these random noises we *do* account for all $2k \log \frac{2}{\alpha}$ potential iterations by sampling random noises with larger variance. However, this larger variance is mitigated by the fact that in the quality function q we divide noises by k, and hence, we *do not* incurr an increase of $\mathrm{poly}(k)$ in the sample complexity due to this issue.)

We have already established that after the tth iteration, the two executions are identical. In addition, due to our assumption on $\vec{w}$ and $\vec{w}'$, during the first t iterations, the only hypotheses with different qualities (between the execution on S and on S') or those hypotheses that label x^* as 0. This is because if a hypothesis h labels x^* as 1, then $(x^*, 0)$ only effects the quality $q(h)$ via the noisy estimation for the size of S^0 (denoted as b_j in the algorithm), which by our assumption on $\vec{w}$ and $\vec{w}'$ is the same in the two executions (because the difference in the noise cancels out the difference in $|S^0|$). To summarize, after conditioning on $\vec{w}$ and $\vec{w}'$, the additional example $(x^*, 0)$ causes the two executions to differ only in their first t iterations, and within these t iterations it affects only the qualities of the hypotheses that label x^* as zero. This can be formalized to bound the ratio in expression (1) by $\lesssim \prod_{j=1}^{t} \exp(\varepsilon \cdot p_j)$, where p_j is the probability that a hypothesis that labels x^* as 0 is chosen at step j of the algorithm. The proof then concludes by arguing that if these probabilities $\{p_j\}$ are small then they reduce our privacy costs (since they multiply ε), and if these probabilities $\{p_j\}$ are large then the index t should be small (since we are likely to identify a hypothesis that labels x^* as zero quickly, and t is the index of the first such hypothesis), and therefore we must only account for the privacy loss incurred during a small number of iterations.

For example, by combining Claim 3.4 with Claim 3.1, we get improved learners for conjunctions and disjunctions:

Theorem 3.5. *There exist efficient (ε, δ)-differentially private (α, β)-PAC learners for $\mathrm{CONJ}_{k,d}$ and $\mathrm{DISJ}_{k,d}$ with sample complexity $n = \tilde{\Theta}\left(\frac{1}{\alpha\varepsilon} \cdot k \log d\right)$. The runtime of the learners is polynomial in $k, d, \frac{1}{\alpha}, \frac{1}{\varepsilon}, \log\frac{1}{\delta}$, and $\log\frac{1}{\beta}$.*

4. Convex Polygons in the Plane

In this section we show how our generic construction from the previous section applies to *convex* polygons in a (discrete version of the) Euclidean plane. This is an important step towards our construction for (not necessarily convex) polygons. We represent a convex polygon with k edges as the intersection of k halfplanes. A halfplane over $\mathbb{R}^2$ can be represented using 3 parameters $a, b, c \in \mathbb{R}$ with $f_{a,b,c}(x, y) = 1$ iff $cy \geq ax + b$. Denote the set of all such halfplanes over $\mathbb{R}^2$ as $\texttt{HALFPLANE} = \{f_{a,b,c} : a, b, c \in \mathbb{R}\}$. We can now define the class of convex polygons with k edges over $\mathbb{R}^2$ as $\texttt{CONVEX-k-GON} = \texttt{HALFPLANE}^{\wedge k}$.

For a parameter $d \in \mathbb{N}$, let $X_d = \{0, 1, 2, \dots, d\}$, and let $X_d^2 = (X_d)^2$ denote a discretization of the Euclidean plane, in which each axis consists of the points in X_d. We assume that our examples are from X_d^2. Hence, as explained next, we are able to represent a halfplane using only two real parameters $a, b \in \mathbb{R}$ and a bit $z \in \{\pm 1\}$. The parameters a and b define the line $y = ax + b$, and the parameter z determines whether the halfplane is "above" or "below" the line. In other words, $f_{a,b,z}(x, y) = 1$ iff $zy \geq z(ax + b)$. Even though in this representation we do not capture vertical lines, for our purposes, vertical lines will not be needed. The reason is that when the examples come from the discretization X_d^2, a vertical line can always be replaced with a non-vertical line such that the corresponding halfplanes behave exactly the same on all of X_d^2. Moreover, since the discretization X_d^2 is finite, it suffices to consider *bounded* real valued parameters $a, b \in [-2d^2, 2d^2]$. Actually, by letting a reside in a bigger range, we can encode the bit z in a, and represent a halfplane using only two real numbers. We denote the set of all such halfplanes as $\texttt{HALFPLANE}_d = \left\{f_{\hat{a},b} : \hat{a} \in [-2d^2, 6d^2],\ b \in [-2d^2, 2d^2]\right\}$, where $f_{\hat{a},b}(x, y) = 1$ iff $zy \geq z(ax + b)$ for $a = \hat{a} - 4d^2 \cdot \mathbb{1}_{\{\hat{a} > 2d^2\}}$ and $z = 1 - 2 \cdot \mathbb{1}_{\{\hat{a} > 2d^2\}}$.

This discussion can be formalized to get the following observation.

Observation 4.1. *For every $f \in \texttt{HALFPLANE}$ there exists an $\hat{f} \in \texttt{HALFPLANE}_d$ such that for every $(x, y) \in X_d^2$ we have $f(x, y) = \hat{f}(x, y)$.*

Remark 4.2. *We think of the discretization size d as a large number, e.g., $d = 2^{64}$. The runtime and the sample complexity of our algorithms is at most logarithmic in d.*

A consequence of Observation 4.1 is that, in order to learn $\texttt{CONVEX-k-GON}$ over examples in X_d^2, it suffices to describe a learner for the class $\texttt{HALFPLANE}_d^{\wedge k}$ over examples in X_d^2. As we next explain, this can be done using our techniques from Section 3. Concretely, we need to specify the selection procedure used in Step 1e of algorithm `SetCoverLearner`, for privately choosing a hypothesis from $\texttt{HALFPLANE}_d$. Our selection procedure appears in algorithm `SelectHalfplane`.

Privacy analysis of `SelectHalfplane`. Consider running algorithm `SelectHalfplane` with a score function q whose sensitivity is (at most) 1, and observe that, as in the standard analysis of the exponential mechanism (McSherry & Talwar, 2007), algorithm `SelectHalfplane` satisfies 2ε-differential privacy. To see this, fix two neighboring databases S, S', and fix $(a, b) \in D^2$. Let us denote the probability density functions in the execution on S and on S' as $p_S(a, b)$ and $p_{S'}(a, b)$, respectively. Since q is of sensitivity 1, we have that $p_S(a, b) \leq e^{2\varepsilon} p_{S'}(a, b)$, and hence, for any set of possible outcomes U we have $\Pr[\texttt{SelectHalfplane}(S) \in U] \leq e^{2\varepsilon} \cdot \Pr[\texttt{SelectHalfplane}(S') \in U]$, as required. Moreover, a similar analysis to that of Claim 3.4 shows the following.

Claim 4.3. *Consider instantiating `SetCoverLearner` with the selection procedure `SelectHalfplane`. In order for the whole execution to satisfy (ε, δ)-differential privacy, it suffices to execute each instance of `SelectHalfplane` with a privacy parameter $\hat{\varepsilon} = O\left(\varepsilon / \log(1/\delta)\right)$.*

Utility analysis of `SelectHalfplane`. In algorithm `SelectHalfplane` we identify points in X_d^2 with lines in D^2 and vice verse. The following observation states that if two points in D^2 belong to the same region (as defined in Step 3) then these two points correspond to halfplanes in X_d^2 that agree on every point in the input sample S. This allows us to partition the halfplanes (in the primal plane) into a small number of equivalence classes.

Observation 4.4. *Consider the execution of `SelectHalfplane` on a sample S, and let $\hat{R} = \{r_i^j\}$ be the regions defined in Step 3 (for $j \in \{1, 2\}$). For every region $r_i^j \in \hat{R}$, for every two points in this region $(a_1, b_1), (a_2, b_2) \in r_i^j$, and for every example (x, y) in the sample S we have $f_{a_1,b_1}(x, y) = f_{a_2,b_2}(x, y)$.*

Proof. Fix two points $(a_1, b_1), (a_2, b_2)$ that belong to the same region in $\hat{R}$. By the definition of the regions in $\hat{R}$,

Algorithm `SelectHalfplane`

Input: Sample $S = \{((x_i, y_i), \sigma_i)\}_{i=1}^n \in (X_d^2 \times \{\pm 1\})^n$, privacy parameter ε, quality function $q : \texttt{HALFPLANE}_d \to \mathbb{R}$.

```
1. Denote $D = \left[-2d^2, 2d^2\right]$ and $F = \left[-2d^2, 6d^2\right]$. We will refer to the axes of $D^2$ and of $F \times D$ as $a$ and $b$.
2. Identify every example $((x, y), \sigma) \in S$ with the line $\ell_{x,y}$ in $D^2$ defined by the equation $y = xa + b$, where $a, b$ are
   the variables and $x, y$ are the coefficients. Denote $S_{\text{dual}} = \{\ell_{x,y} : ((x, y), \sigma) \in S\}$.
3. Let $R = \{r_1^1, r_2^1, \ldots, r_{|R|}^1\}$ denote the partition of $D^2$ into regions defined by the lines in $S_{\text{dual}}$. Also let $R' =$
   $\{r_1^2, \ldots, r_{|R|}^2\}$ be a partition of $[2d^2, 6d^2] \times D$ identical to $R$ except that it is shifted by $4d^2$ on the $a$ axis. Denote
   $\hat{R} = R \cup R'$.
   % Note that, by induction, $n$ lines can divide the plane into at most $n^2$ different regions. Hence, $|R|$ is small.
4. For every $1 \le i \le |R|$, let $w_i$ denote the area of region $r_i^1$ (which is the same as the area of $r_i^2$), and let $(a_i^1, b_i^1) \in r_i^1$
   and $(a_i^2, b_i^2) \in r_i^2$ be arbitrary points in these regions.
5. Denote $N = \sum_{r_i^j \in \hat{R}} w_i \cdot \exp(\varepsilon \cdot q(f_{a_i^j, b_i^j}))$, where $f_{a_i^j, b_i^j}$ is a halfplane in $\texttt{HALFPLANE}_d$.
6. Choose and return a pair $(\hat{a}, b) \in [-2d^2, 6d^2] \times [-2d^2, 2d^2]$ with probability density function $p(\hat{a}, b) = \frac{1}{N} \cdot \exp(\varepsilon \cdot$
   $q(f_{\hat{a},b}))$.
   % Note that for every $(a, b), (a', b') \in r_i^j$ in the same region we have $q(f_{a,b}) = q(f_{a',b'})$ (see Observation 4.4). Hence, this step
   can be implemented by first selecting a region $r_i^j \in \hat{R}$ with probability proportional to $w_i \cdot \exp(\varepsilon \cdot q(f_{a_i^j, b_i^j}))$, and then selecting a
   random $(a, b) \in r_i^j$ uniformly.
```

for every example (x, y) in the sample S we have that $y \ge a_1 x + b_1$ if and only if $y \ge a_2 x + b_2$, and hence, $f_{a_1,b_1}(x, y) = 1$ iff $f_{a_2,b_2}(x, y) = 1$. □

In particular, Observation 4.4 shows that the function p defined in Step 6 indeed defines a probability density function, as for $F = [-2d^2, 6d^2]$ and $D = [-2d^2, 2d^2]$ we have

$$\int_{F \times D} p(a, b)\, \mathrm{d}^2(a, b) = \sum_{r_i^j \in \hat{R}} \int_{r_i^j} p(a, b)\, \mathrm{d}^2(a, b)$$
$$= \sum_{r_i^j \in \hat{R}} \int_{r_i^j} \frac{\exp(\varepsilon \cdot q(f_{a,b}))}{N}\, \mathrm{d}^2(a, b)$$
$$= \sum_{r_i^j \in \hat{R}} \int_{r_i^j} \frac{\exp(\varepsilon \cdot q(f_{a_i^j, b_i^j}))}{N}\, \mathrm{d}^2(a, b)$$
$$= \sum_{r_i^j \in \hat{R}} \frac{\exp(\varepsilon \cdot q(f_{a_i^j, b_i^j}))}{N} \int_{r_i^j} 1\, \mathrm{d}^2(a, b)$$
$$= \sum_{r_i^j \in \hat{R}} \frac{w_i \cdot \exp(\varepsilon \cdot q(f_{a_i^j, b_i^j}))}{N} = 1.$$

We also need to argue about the *area* of the region in the dual plane that corresponds to hypotheses with high quality (as the probability of a choosing a hypotheses from that region is proportional to its area). This is done in the following claim.

Claim 4.5. *Consider the execution of* `SelectHalfplane` *on a sample S, and let $w_1, \ldots, w_{|R|}$ denote the areas of the regions defined in Step 3. Then for every i we have that $w_i \ge d^{-4}/4$.*

Proof. We will show that every two different vertices of the regions in R are at distance at least $1/d^2$, and hence, the minimal possible area is that of an equilateral triangle with edge length $1/d^2$, which has area $\frac{\sqrt{3}}{4d^4}$.

To show this lower bound on the distance between a pair of vertices, let $\ell_{x_1,y_1}, \ell_{x_2,y_2}, \ell_{x_3,y_3}, \ell_{x_4,y_4}$ be 4 lines in S_{dual}, and assume that ℓ_{x_1,y_1} and ℓ_{x_2,y_2} intersect at $(a_{1,2}, b_{1,2})$, and that ℓ_{x_3,y_3} and ℓ_{x_4,y_4} intersect at $(a_{3,4}, b_{3,4})$. Moreover, assume that these two intersection points are different. We can write the coordinates of these intersection points as

$$a_{1,2} = \frac{y_1 - y_2}{x_1 - x_2}, \qquad b_{1,2} = y_1 - x_1 \cdot \frac{y_1 - y_2}{x_1 - x_1},$$
$$a_{3,4} = \frac{y_3 - y_4}{x_3 - x_4}, \qquad b_{3,4} = y_3 - x_3 \cdot \frac{y_3 - y_4}{x_3 - x_4}.$$

Now if $a_{1,2} \neq a_{3,4}$, then

$$\|(a_{1,2}, b_{1,2}) - (a_{3,4}, b_{3,4})\|_2 \ge |a_{1,2} - a_{3,4}|$$
$$= \left| \frac{y_1 - y_2}{x_1 - x_2} - \frac{y_3 - y_4}{x_3 - x_4} \right|$$
$$= \left| \frac{(y_1 - y_2)(x_3 - x_4) - (y_3 - y_4)(x_1 - x_2)}{(x_1 - x_2)(x_3 - x_4)} \right| \ge \frac{1}{d^2},$$

and if $a_{1,2} = a_{3,4}$, then

$$\|(a_{1,2}, b_{1,2}) - (a_{3,4}, b_{3,4})\|_2 \ge |b_{1,2} - b_{3,4}| =$$
$$= |y_1 - y_3 - a_{1,2}(x_1 - x_3)| \ge \frac{1}{d}.$$ □

The following lemma states the utility guarantees of `SelectHalfplane`.

Lemma 4.6. *Consider executing* `SelectHalfplane` *on a sample S, and assume that there exists a hypothesis $f \in \texttt{HALFPLANE}_d$ with $q(f) \geq \nu$. Then with probability at least $1-\beta$, the output f' is s.t. $q(f') \geq \nu - \frac{8}{\varepsilon}\ln(\frac{2d}{\beta})$.*

Proof. Denote $F = [-2d^2, 6d^2]$ and $D = [-2d^2, 2d^2]$. Let $\hat{R} = \{r_1^1, r_1^2, \dots, r_{|R|}^1, r_{|R|}^2\}$ denote the regions defined in Step 3, and let $B \subseteq \hat{R}$ denote the subset of all regions s.t. the halfplanes that correspond to points in these regions have quality less than $\nu - \frac{8}{\varepsilon}\ln(\frac{2d}{\beta})$. Then the probability that `SelectHalfplane` outputs a hypothesis f' with $q(f') < \nu - \frac{8}{\varepsilon}\ln(\frac{2d}{\beta})$ is at most

$$\begin{aligned}\sum_{r\in B}\int_r p(a,b)\,\mathrm{d}^2(a,b) &\leq \sum_{r\in B}\int_r \frac{e^{\varepsilon\nu - 8\ln(\frac{2d}{\beta})}}{N}\,\mathrm{d}^2(a,b)\\ \leq \frac{e^{\varepsilon\nu-8\ln(\frac{2d}{\beta})}\cdot \mathrm{area}\,(F\times D)}{N} &= \frac{e^{\varepsilon\nu-8\ln(\frac{2d}{\beta})}\cdot 32d^4}{N}\\ \leq \frac{e^{\varepsilon\nu-8\ln(\frac{2d}{\beta})}\cdot 32d^4}{1/(4d^4)\cdot\exp(\varepsilon\nu)} &= 128d^8e^{-8\ln(2d/\beta)} \leq \beta. \qquad \square\end{aligned}$$

Combining Lemma 4.6 with Claim 4.3 and Theorem 3.2 yields our private learners for convex polygons:

Theorem 4.7. *There exists an efficient (ε,δ)-differentially private (α,β)-PAC learner for* `CONVEX-k-GON` *over examples from X_d^2 with sample complexity $O\left(\frac{k}{\alpha\varepsilon}\log\left(\frac{1}{\alpha}\right)\log\left(\frac{1}{\delta}\right)\log\left(\frac{dk}{\beta}\log\frac{1}{\alpha}\right)\right)$.*

5. Union of Non-Convex Polygons

In this section we briefly describe how our techniques from the previous sections can be used to learn the class of (simple) polygons in the plane, as defined next. For a simple and closed curve[3] C, we use $\texttt{interior}(C)$ to denote the union of C and its bounded area. We define the class of all simple polygons in the plane with (at most) k edges as

$$\texttt{k-GON} = \left\{ \texttt{interior}(C) : \begin{array}{l} C \text{ is a simple and closed} \\ \text{curve in } \mathbb{R}^2, \text{ consisting} \\ \text{of at most } k \text{ line segments}\end{array}\right\}.$$

By standard arguments in computational geometry, every such polygon with k edges can be represented as the union of at most k triangles, each of which can be represented as the intersection of at most 3 halfplanes (since a triangle is a convex polygon with 3 edges). Let us denote the class of all triangles in the plane as `TRIANGLE`. Hence,

$$\texttt{k-GON} \subseteq \texttt{TRIANGLE}^{\vee k} \subseteq \left(\texttt{HALFPLANE}^{\wedge 3}\right)^{\vee k}.$$

[3]A curve is simple and closed if it does not cross itself and ends at the same point where it begins.

Thus, in order to learn polygons with k edges, it suffices to construct a learner for the class $\left(\texttt{HALFPLANE}^{\wedge 3}\right)^{\vee k}$. In fact, this class captures *unions* of polygons with a total of at most k edges. In addition, similar arguments to those given in Section 4 show that if input examples come from $X_d^2 = \{0,1,\dots,d\}^2$, then it suffices to construct a learner for $\left(\texttt{HALFPLANE}_d^{\wedge 3}\right)^{\vee k}$, which we can do using our techniques from Sections 3 and 4.

First, as we mentioned, a straightforward modification to algorithm `SetCoverLearner` yields an algorithm for learning classes of the form $C \subseteq H^{\vee k}$ (instead of $C \subseteq H^{\wedge k}$ as stated in Section 3). Now, to get an efficient construction, we need to specify the selection procedure for choosing a hypothesis $h_j \in \texttt{HALFPLANE}_d^{\wedge 3}$ in each step of `SetCoverLearner`. As before, given an input sample S we consider the dual plane D^2 s.t. every input example in S from the primal plane corresponds to a line in the dual plane, and every point from the dual plane corresponds to a halfplane in the primal plane. Recall that in the previous section we identified a hypothesis (which was a halfplane) with a point in the dual plane. The modification is that now a hypothesis is a triangle which we identify with *three points* in the dual plane (these 3 points correspond to 3 halfplanes in the primal plane, whose intersection is a triangle).

We use `k-UNION-GON` to denote the class of all unions of (simple) polygons with a total of at most k edges. That is, every hypothesis $h \in \texttt{k-UNION-GON}$ can be written as $h = h_1 \vee \cdots \vee h_m$ for $(h_1,\dots,h_m) \in (\texttt{k}_1\texttt{-GON} \times \cdots \times \texttt{k}_\texttt{m}\texttt{-GON})$ where $k_1 + \cdots + k_m \leq k$. We obtain the following theorem.

Theorem 5.1. *There exists an efficient (ε,δ)-differentially private (α,β)-PAC learner for* `k-UNION-GON` *over examples from X_d^2 with sample complexity $O\left(\frac{k}{\alpha\varepsilon}\log\left(\frac{1}{\alpha}\right)\log\left(\frac{1}{\delta}\right)\log\left(\frac{dk}{\beta}\log\frac{1}{\alpha}\right)\right)$. The runtime of the learner is polynomial in $k, \frac{1}{\alpha}, \frac{1}{\varepsilon}, \log\frac{1}{\delta}, \log\frac{1}{\beta}$, and $\log d$.*

6. Conclusion and Future Work

In this work we presented a computationally efficient differentially private PAC learner for simple geometric concepts in the plane, which can be described as the union of polygons. Our results extend to higher dimensions by replacing lines with hyperplanes, and triangles with simplices. The running time, however, depends exponentially on the dimension. Our results also extend, via linearization, to other simple geometric concepts whose boundaries are defined by low degree polynomials, such as balls. In general, the dimension of the linearization depends on the degrees of the polynomials. This motivates the open problem of improving the dependency of the running time on the dimension of the problem.

Acknowledgements

The authors would like to thank Kobbi Nissim, and the anonymous reviewers, for their helpful comments.

References

Abowd, J. M. The challenge of scientific reproducibility and privacy protection for statistical agencies. Census Scientific Advisory Committee, 2016.

Alon, N., Livni, R., Malliaris, M., and Moran, S. Private PAC learning implies finite littlestone dimension. *CoRR*, abs/1806.00949, 2018.

Bassily, R., Thakurta, A. G., and Thakkar, O. D. Model-agnostic private learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pp. 7102–7112, 2018.

Beimel, A., Kasiviswanathan, S. P., and Nissim, K. Bounds on the sample complexity for private learning and private data release. In *TCC*, volume 5978 of *LNCS*, pp. 437–454. Springer, 2010.

Beimel, A., Nissim, K., and Stemmer, U. Characterizing the sample complexity of private learners. In *ITCS*, pp. 97–110. ACM, 2013a.

Beimel, A., Nissim, K., and Stemmer, U. Private learning and sanitization: Pure vs. approximate differential privacy. In *APPROX-RANDOM*, volume 8096 of *LNCS*, pp. 363–378. Springer, 2013b.

Beimel, A., Nissim, K., and Stemmer, U. Learning privately with labeled and unlabeled examples. In *SODA*, pp. 461–477. SIAM, 2015.

Beimel, A., Moran, S., Nissim, K., and Stemmer, U. Private center points and learning of halfspaces. *CoRR*, abs/1902.10731, 2019.

Bun, M. and Zhandry, M. Order-revealing encryption and the hardness of private learning. In *TCC*, volume 9562 of *LNCS*, pp. 176–206. Springer, 2016.

Bun, M., Nissim, K., Stemmer, U., and Vadhan, S. P. Differentially private release and learning of threshold functions. In *FOCS*, pp. 634–649, 2015.

Bun, M., Nissim, K., and Stemmer, U. Simultaneous private learning of multiple concepts. In *ITCS*, pp. 369–380. ACM, 2016.

Chaudhuri, K. and Hsu, D. Sample complexity bounds for differentially private learning. In *COLT*, volume 19 of *JMLR Proceedings*, pp. 155–186, 2011.

Dwork, C. and Feldman, V. Privacy-preserving prediction. In *COLT*, pp. 1693–1702, 2018.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *LNCS*, pp. 265–284. Springer, 2006.

Dwork, C., Rothblum, G. N., and Vadhan, S. P. Boosting and differential privacy. In *FOCS*, pp. 51–60. IEEE Computer Society, 2010.

Eisenstat, D. and Angluin, D. The VC dimension of k-fold union. *Inf. Process. Lett.*, 101(5):181–184, 2007. doi: 10.1016/j.ipl.2006.10.004.

Erlingsson, Ú., Pihur, V., and Korolova, A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

Feldman, V. and Xiao, D. Sample complexity bounds on differentially private learning via communication complexity. *SIAM J. Comput.*, 44(6):1740–1764, 2015.

Gupta, A., Ligett, K., McSherry, F., Roth, A., and Talwar, K. Differentially private combinatorial optimization. In *SODA*, pp. 1106–1125, 2010.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

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

Kearns, M. J. and Vazirani, U. V. *An Introduction to Computational Learning Theory*. MIT press, Cambridge, Massachusetts, 1994.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *FOCS*, pp. 94–103. IEEE Computer Society, 2007.

Thakurta, A., Vyrros, A., Vaishampayan, U., Kapoor, G., Freudiger, J., Sridhar, V., and Davidson, D. Learning new words. *US Patent 9594741*, 2017.

Vadhan, S. The complexity of differential privacy, 2016.