Solving Linear Programs with Differential Privacy

Alina Ene
Department of Computer Science, Boston University, MA, USA

Huy Le Nguyen
Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA

Ta Duy Nguyen
Department of Computer Science, Boston University, MA, USA

Adrian Vladu
CNRS & IRIF, Université Paris Cité, France

Abstract

We study the problem of solving linear programs of the form $Ax \leq b$, $x \geq 0$ with differential privacy. For homogeneous LPs $Ax \geq 0$, we give an efficient (ϵ, δ)-differentially private algorithm which with probability at least $1 - \beta$ finds in polynomial time a solution that satisfies all but $O(\frac{d^2}{\epsilon} \log^2 \frac{d}{\delta\beta} \sqrt{\log \frac{1}{\rho_0}})$ constraints, for problems with margin $\rho_0 > 0$. This improves the bound of $O(\frac{d^5}{\epsilon} \log^{1.5} \frac{1}{\rho_0} \mathrm{poly}\log(d, \frac{1}{\delta}, \frac{1}{\beta}))$ by [Kaplan-Mansour-Moran-Stemmer-Tur, STOC '25]. For general LPs $Ax \leq b$, $x \geq 0$ with potentially zero margin, we give an efficient (ϵ, δ)-differentially private algorithm that w.h.p drops $O(\frac{d^4}{\epsilon} \log^{2.5} \frac{d}{\delta} \sqrt{\log dU})$ constraints, where U is an upper bound for the entries of A and b in absolute value. This improves the result by Kaplan et al. by at least a factor of d^5. Our techniques build upon privatizing a rescaling perceptron algorithm by [Hoberg-Rothvoss, IPCO '17] and a more refined iterative procedure for identifying equality constraints by Kaplan et al.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Security and privacy

Keywords and phrases Differential Privacy, Linear Programming

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.65

Category RANDOM

Funding *Alina Ene*: AE was supported in part by an Alfred P. Sloan Research Fellowship.
Huy Le Nguyen: HN was supported in part by NSF CCF 2311649 and a gift from Apple Inc.
Adrian Vladu: AV was supported by the French Agence Nationale de la Recherche (ANR) under grant ANR-21-CE48-0016 (project COMCOPT).

1 Introduction

Linear programming is a fundamental tool for modeling and solving problems in computer science. Consider the standard feasibility problem of finding $x \in \mathbb{R}^d$ subject to $Ax \leq b, x \geq 0$, where the constraints $Ax \leq b$ are input from users. The input can contain sensitive information such as the users' private health data or private transactions, which the users may wish to be protected. For the algorithm designer, this is where differential privacy proves its usefulness. Differential privacy protects sensitive information by requiring that the algorithm must have approximately the same output upon receiving similar input.

The study of solving linear programs with differential privacy was initiated by [9] and has subsequently been studied under different contexts along with other related problems. Notably, privately solving linear programs is closely related to private learning of subspaces and halfspaces, which are fundamental problems in learning theory. In particular, a line of work by [3, 1, 11, 7, 2] showed reductions from learning halfspaces to the problem of finding a point in a convex hull, which in turn can be solved via linear programing. This means that the existence of an efficient private solver for linear programs implies an upper bound for the sample complexity of efficient private learners for halfspaces, and any improvement for the former problem implies an improvement for the latter.

Imposing differential privacy when solving a linear program comes with the impossibility to ensure *all* constraints are satisfied. Indeed, in the extreme case, the addition of one more constraint can change the problem from feasible to infeasible. Therefore, to guarantee differential privacy, it is required that a number of constraints must be dropped. It was known (by folklore) that for a linear program whose feasible region is of positive volume, privatizing the algorithm by [4] results in a solution that violates $\mathrm{poly}(d)$ constraints, where d is the dimension of the problem. The recent work by [10] formalized this claim and generalized it to all linear programs. Their algorithm, however, suffered from a high degree polynomial dependence on d (at least d^9 dependence), whereas the lower bound (from learning theory) is linear in d. Closing this gap remains a challenging open question.

In our work, we make progress in this direction, and propose new algorithms for solving linear programs with differential privacy. Our algorithms are efficient and they achieve significantly improved guarantees on the number of violated constraints.

1.1 Our contribution

Our first contribution is a new algorithm for privately solving linear programs with positive margin with guarantee stated in Theorem 1. Here the margin of the problem is the radius of the largest ball that fits in the intersection of the feasible region and the unit ball, which we define in Section 2.

▶ **Theorem 1.** *Let $\epsilon, \delta, \beta \geq 0$ be given an input. There exists an efficient (ϵ, δ)-differentially private algorithm for finding a feasible solution to the linear program $Ax \geq 0$, $x \neq 0$ such that whenever the margin of the system is at least ρ_0, with probability at least $1 - \beta$, the algorithm outputs a solution x that satisfies all but $O\left(\frac{d^2}{\epsilon}\sqrt{\log\frac{1}{\rho_0}}\log^2\frac{d}{\beta\delta}\right)$ constraints.*

In terms of the dependence on the dimension d, our algorithm significantly improves over the prior work by [10], which drops $O\left(d^5 \log^{1.5}\frac{1}{\rho_0}\mathrm{poly}\log\left(d, \frac{1}{\delta}, \frac{1}{\beta}\right)\right)$ constraints.

For general linear programs with potentially zero margin, we give an iterative private algorithm with the following guarantee.

▶ **Theorem 2.** *Let $\epsilon, \delta, \beta \geq 0$ be given an input. There exists an efficient (ϵ, δ)-differentially private algorithm for finding a feasible solution to the linear program $Ax \leq b$; $x \geq 0$ with integer entries such that with probability at least $1 - (\beta + \delta)$, the algorithm outputs a solution x that satisfies all but $O\left(\frac{d^4}{\epsilon}\sqrt{\log dU}\log^{2.5}\frac{d}{\beta\delta}\right)$ constraints, where U is an upper bound on the absolute values of the entries in A and b.*

The algorithm by [10] requires to drop $\tilde{O}(d^9)$ constraints to guarantee privacy. To improve this, one can use the algorithm from Theorem 1 as a subroutine in the algorithm by [10] to reduce to the number of dropped constraints to $\tilde{O}(d^5)$. Our algorithm with a more refined analysis goes one step further to remove another factor d and achieves $\tilde{O}(d^4)$ dependence. Our bound is a significant improvement towards the lower bound $\Omega(d)$.

1.2 Our techniques

Our technique for showing Theorem 1 is based on privatizing a rescaling perceptron algorithm for solving linear programs of the form $Ax \geq 0, x \neq 0$ with positive margin. Instead of using the algorithm by [4] as in [10], we develop an algorithm based on the work of [8].

A rescaling perceptron algorithm consists of two phases: the Perceptron Phase to find a new solution and the Rescaling Phase to find a direction to rescale the input. To privatize the Perceptron Phase, the algorithm of [10] uses the NoisyAvg mechanism by [13] to construct a noisy average of the constraints that are violated by the current solution, and then updates the solution in the same direction. In the Rescaling Phase, the non-private algorithm by [4] and the privatized version by [10] use another perceptron-style procedure to find a rescaling direction. This approach requires $\tilde{O}(d^2)$ updates and each call to the Rescaling Phase requires to drop $\tilde{O}(d^{2.5})$ constraints.

The main difference between our work and [10] lies in the rescaling phase. We use the following technique by [14] and [8]. During the perceptron phase, the algorithm maintains a weight vector λ for tracking which constraints are violated by the solution in each iteration. Using a random Gaussian vector, the algorithm produces a rescaling direction by a convex combination weighted by λ of the rows satisfied by the random vector. [14] and [8] show that with a constant probability, rescaling the input matrix along that direction increases the volume of the feasible region inside the unit ball by a constant factor.

The benefit of using the weight vector to rescale is that, since we only need $O(\log \frac{1}{\beta})$ steps to boost the success probability, the amount of noise needed to make this phase private is much smaller. Further, we can keep all constraints around until we find a good solution and only discard constraints once at the end of the algorithm. We discard only $O(d^2)$ constraints in total.

For general linear programs with potentially zero margin, we use the standard technique of adding a small perturbation to the constraints that does not change the feasibility of the problem. This perturbation increases the problem margin and allows the application of the private perceptron algorithm. Similar to [10], our algorithm iteratively identifies tight (equality) constraints in the LP, privatizes these equality constraints and uses them to eliminate variables. The approach by [10] needs to account for the blow-up of the input entries after performing the variable elimination steps, which can quickly diminish the margin. The cost of this shows up as an extra factor d in the number of dropped constraints. By contrast, our algorithm always returns to the initial LP after identifying tight constraints. We show that in this way the margin reduces at a slower rate, saving the factor d.

1.3 Related work

Rescaling Perceptron algorithms. To find a solution $x \neq 0$ that satisfies $Ax \geq 0$, one can use the classic perceptron algorithm which convergences after at most $1/\rho^2$ iterations on problems with positive margin ρ, where the margin is the radius of the largest ball that fits in the intersection of the feasible region and the unit ball. [4] show a modification of the classic algorithm with an additional rescaling procedure that runs in time $\tilde{O}(nd^4 \log \frac{1}{\rho})$, for problems with $\rho > 0$. The rescaling procedure by [4] is another variant of the perceptron algorithm which finds a rescaling direction by moving a random unit vector along the direction of a violated constraint. Subsequent works by [14, 8] explore different rescaling operations. In particular, our work relies on the technique by [8] in which the rescaling direction is found by a convex combination of rows whose corresponding constraints are satisfied by a random Gaussian vector.

Solving linear programs with privacy. Solving linear programs with differential privacy has been the focus of several prior works. Under various notions of neighboring inputs (such as input differing by a row or a column), [9] give algorithms that approximately satisfy most constraints. [12] show an algorithm that satisfy most constraints exactly, but only considering neighboring inputs to be differing on the right hand side scalars. [11] provide an algorithm to solve the general feasibility problem $Ax \geq b$, but requires a running time that is exponential in the dimension. Most relevant for our work is the work of [10], which studies the same setting. We provide a detailed comparison of the results and techniques in sections 1.1 and 1.2.

Beyond solving linear programs. Solving linear programs is closely related to private learning subspaces and halfspaces [3]. [1] show a reduction from learning halfspaces to the problem of finding a point in the convex hull, where an efficient algorithm for the latter problem implies an upper bound for the sample complexity of efficient algorithms for the former. Subsequent works by [11, 7, 2] have targeted this question. [10] build techniques for solving this problem via privately solving LPs and achieve the first algorithm that has polynomial dependence on the dimension and polylogarithmic dependence on the domain size. Our algorithm can be used as a subroutine to improve the runtime of the algorithm by [10].

2 Preliminaries

Notation. We let $\|\cdot\|_1$ be the ℓ_1-norm and $\|\cdot\|_2$ be the ℓ_2-norm. When it is clear from context, $\|\cdot\|$ also denotes the $\ell_2$norm. For a matrix A, we denote by a_i the i-th row of A. For a vector a, we let $\overline{a}$ be the normalized vector $\overline{a} = \frac{a}{\|a\|_2}$. We also use $\overline{A}$ to denote the matrix A with normalized rows. We denote by $\mathrm{Lap}(b)$ the Laplace distribution with density $f(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$; $N(0, \sigma^2)$ the Gaussian distribution with mean zero and variance σ^2 and $N(0, \sigma^2 I)$ the multivariate Gaussian distribution with mean zero and covariance $\sigma^2 I$. The dimension will be clear from context.

Linear programs. We consider the problem of finding a feasible solution to a linear program in general standard form $Ax \leq b, x \geq 0$, where A has dimension $n \times d$, b is a vector of dimension n and the entries of A and b are integers. Following [4], we refer to the problem of finding a solution satisfying $Ax \geq 0$, $x \neq 0$ as a homogeneous LP. A homogeneous LP $Ax \geq 0, x \neq 0$ is characterized by a quantity $\rho(A)$, namely, the margin (or roundness) parameter, given as

$$\rho(A) = \max_{\|x\|_2 \leq 1} \min_i \langle \overline{a}_i, x \rangle .$$

Geometrically, $\rho(A)$ is the radius of a ball that fits into the intersection between the feasible region and the unit ball. The classic perceptron algorithm for LPs with $\rho(A) > 0$ converges with $1/\rho(A)^2$ iterations. Rescaling algorithms such as [4, 14, 8] have total runtime $\mathrm{poly}(n, d) \log 1/\rho(A)$.

Homogenization. [4] give a simple reduction (called homogenization) from a general LP $Ax \leq b, x \geq 0$ to a homogeneous LP $A'x \geq 0$, $x \neq 0$ by setting $A' = \begin{bmatrix} -A \mid b \\ I \end{bmatrix}$ and $x = (x \mid x_0)^\top$. We refer to the homogeneous LP constructed via this reduction as the homogenized LP.

Differential privacy. We use the notation (A, b) as shorthand for the LP $Ax \leq b, x \geq 0$. We say that two LPs (A, b) and (A', b') are neighbors is they differ by only one constraint (one LP has an extra constraint). A randomized algorithm $\mathcal{A}$ is said to be (ϵ, δ)-differentially private (DP) if for all neighboring LPs (A, b) and (A', b') and every subset of possible outcomes $\mathcal{O}$,

$$\Pr\left[\mathcal{A}(A, b) \in \mathcal{O}\right] \leq e^{\epsilon} \Pr\left[\mathcal{A}(A', b') \in \mathcal{O}\right] + \delta.$$

In the case $\delta = 0$, we say the algorithm is ϵ-DP. Two commonly used mechanisms for achieving differential privacy are the Laplace mechanism and the Gaussian mechanism. Let f be a function whose output has dimension d. We say f has ℓ_1 sensitivity k if on any two neighboring inputs x and x', $\|f(x) - f(x')\|_1 \le k$, and f has ℓ_2 sensitivity k if on any two neighboring inputs x and x', $\|f(x) - f(x')\|_2 \le k$.

▶ **Theorem 3** (Laplace mechanism [6]). *Let f be a function of ℓ_1 sensitivity k. The mechanism $\mathcal{A}$ that on an input x adds independently generated noise with the Laplace distribution* $\mathrm{Lap}(\frac{k}{\epsilon})$ *to each of the d coordinates of $f(x)$ is ϵ-DP.*

▶ **Theorem 4** (Gaussian mechanism [5]). *Let f be a function of ℓ_2 sensitivity k. The mechanism $\mathcal{A}$ that on an input x adds noise generated with the Gaussian distribution $N(0, \sigma^2)$ where $\sigma \ge \frac{k}{\epsilon}\sqrt{2\ln\frac{2}{\delta}}$ to each of the d coordinates of $f(x)$ is (ϵ, δ)-DP.*

3 Private Perceptron Algorithm for Positive Margin LPs

3.1 Algorithm

Algorithm 1 Private Perceptron.

```
1: Input: Matrix $A \in \mathbb{R}^{n \times d}$, parameters $\rho_0, \epsilon, \delta, \beta$
2: Let $B = I$
3: for $t = 1 \ldots \tau = O\left(d \log \frac{1}{\rho_0}\right)$:
4:     Run $c \leftarrow \mathsf{PrivatePerceptronEpoch}(A, \epsilon, \delta, \beta)$
5:     if $c$ is a solution:
6:         return $Bc$
7:     else if $c$ is a rescaling direction:
8:         $A \leftarrow A\left(I - \frac{1}{2}\overline{c} \cdot \overline{c}^\top\right)$; $B \leftarrow \left(I - \frac{1}{2}\overline{c} \cdot \overline{c}^\top\right) B$
9:     else:
10:        abort; return $\perp$
11: return $\perp$
```

Algorithm 2 PrivatePerceptronEpoch$(A, \epsilon, \delta, \beta)$.

```
1: $x^{(1)} = (0, \ldots, 0) \in \mathbb{R}^d$
2: $\lambda^{(1)} = (0, \ldots, 0) \in \mathbb{R}^n$
3: $\nu = \Theta\left(\frac{\sqrt{d}}{\epsilon} \log^{1.5} \frac{T\tau}{\beta\delta}\right)$, $\sigma^2 = \frac{8 \log \frac{4T}{\delta}}{\nu^2 \epsilon^2}$, $\theta^2 = \frac{8}{d^2 \epsilon^2 \nu^2} \log \frac{4000 \log \frac{\tau}{\beta}}{\delta}$
4: for $t = 1 \ldots T = \Theta(d^2)$:                                    Perceptron Phase
5:     $S^{(t)} = \left\{i \in [n], \left\langle \overline{a}_i, \overline{x}^{(t)} \right\rangle \le 0\right\}$, $m^{(t)} = \left|S^{(t)}\right|$
6:     Let $\widehat{m}^{(t)} = m^{(t)} + \mathrm{Lap}\left(\frac{1}{\epsilon}\right) - \frac{1}{\epsilon} \log \frac{2000T \log \frac{1}{\beta}}{\delta}$. If $\widehat{m}^{(t)} \le \nu$ then return solution $x^{(t)}$.
7:     $u^{(t)} = \frac{1}{m^{(t)}} \sum_{i \in S^{(t)}} \overline{a}_i$, $\widehat{u}^{(t)} = u^{(t)} + \eta^{(t)}$ where $\eta^{(t)} \sim N(0, \sigma^2 I)$
8:     $x^{(t+1)} = x^{(t)} + \widehat{u}^{(t)}$
9:     $\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{1}{m^{(t)}}$ for all $i \in S^{(t)}$          $\lambda^{(t)}$ are kept private
10: Let $\overline{\lambda} = \frac{\lambda^{(T)}}{T}$
11: for $s = 1, \ldots, 1000 \log \frac{\tau}{\beta}$:                              Rescaling Phase
12:     Take a gaussian vector $g^{(s)} \sim N(0, I)$ and compute $P^{(s)} = \left\{i : \left\langle \overline{a}_i, g^{(s)} \right\rangle \ge 0\right\}$
13:     Let $c^{(s)} = \sum_{i \in P^{(s)}} \overline{\lambda}_i \overline{a}_i$ and $\widehat{c}^{(s)} = \sum_{i \in P^{(s)}} \overline{\lambda}_i \overline{a}_i + \gamma^{(s)}$ where $\gamma^{(s)} \sim N(0, \theta^2 I)$
14:     if $\left\|\widehat{c}^{(s)}\right\| \ge \frac{3}{16\sqrt{\pi d}}$, return $\widehat{c}^{(s)}$
15: Output $\perp$
```

We describe our Private Perceptron algorithm in Algorithm 1. Given a matrix $A \in \mathbb{R}^{n\times d}$, margin parameter ρ_0, privacy parameters ϵ, δ, and failure probability β, the algorithm runs at most $\tau = O(d \log \frac{1}{\rho_0})$ and makes calls to PrivatePerceptronEpoch procedure given in Algorithm 2. Algorithm 2 has three possible outcomes. If it outputs a solution, Algorithm 1 terminates and returns this solution with a suitable rescaling. If it outputs a rescaling direction, Algorithm 1 rescales the input matrix A and repeats. Otherwise, Algorithm 1 terminates and returns $\perp$.

Our main novel contribution lies in Algorithm 2. This algorithm consists of two phases: the Perceptron Phase in which the algorithm attempts to find a solution to the LP and the Rescaling Phase in which the algorithm finds a good direction to rescale the input if the solution from the Perceptron Phase is not satisfactory. In each phase, we add maintain the privacy by adding appropriate noise. During the Perceptron Phase, the algorithm maintains a solution and updates it along the direction of a noisy average of the violated constraints. The algorithm also maintains a weight vector $\lambda^{(t)}$ which picks up the constraints violated by the current solution. We keep $\lambda^{(t)}$ private, and use the average value $\overline{\lambda}$ to determine the rescaling direction. To determine the rescaling direction, during the Rescaling Phase, the algorithm takes a random gaussian vector and computes a noisy sum of all rows satisfied by this vector weighted by $\overline{\lambda}$. We will show that with a constant probability, this noisy sum provides a good rescaling direction, and the algorithm repeats the process a number of iterations to boost the success probability.

In the next subsections, we show the privacy and utility guarantees of our algorithm.

3.2 Privacy analysis

Throughout, we let (A, b) and (A', b') be two neighboring LPs. The corresponding computed terms in Algorithm 2 for (A', b') are denoted with the extra prime symbol (for example, $S^{(t)}$ and $S^{(t)\prime}$).

▶ **Proposition 5.** *Algorithm 1 is (ϵ', δ')-DP for $\epsilon' = 2d^3 \log \frac{1}{\rho_0} \epsilon^2 + \sqrt{2d^3\epsilon^2 \log \frac{1}{\rho_0} \log \frac{1}{\delta}}$ and $\delta' = (d+1)\delta$.*

To show this Proposition, we show the following lemmas.

▶ **Lemma 6.** *Each iteration of the Perceptron Phase is $(\epsilon, \frac{\delta}{2T})$-DP.*

The following claim follows similarly to the proof of the NoisyAvg algorithm by [13]. We include the proof in the appendix.

▷ Claim 7. For all $t \in [T]$, $m^{(t)}$ has ℓ_1 sensitivity 1, and $u^{(t)}$ has ℓ_2 sensitivity $\frac{2}{m^{(t)}}$.

Proof of Lemma 6. For the purpose of analysis, in each iteration of the Perceptron Phase, we define the output of the algorithm to be either the solution $x^{(t)}$ or the new update vector $\widehat{u}^{(t)}$. Let O, O' be the outputs of the iteration on (A, b) and (A', b') respectively, and let F be an arbitrary subset of $\mathbb{R}^d$. First, due the the privacy of the Laplace mechanism,

$$\Pr\left[O = x^{(t)}\right] = \Pr\left[\widehat{m}^{(t)} \le \nu\right] \le e^{\epsilon} \Pr\left[\widehat{m}^{(t)\prime} \le \nu\right] = e^{\epsilon} \Pr\left[O' = x^{(t)}\right]$$

Next, consider the case where the output is an update vector $\widehat{u}^{(t)}$. If $m^{(t)} < \nu$:

$$\Pr\left[O = \widehat{u}^{(t)}\right] \le \Pr\left[\widehat{m}^{(t)} > m^{(t)}\right] \le \Pr\left[\mathrm{Lap}\left(\frac{1}{\epsilon}\right) > \frac{1}{\epsilon} \log \frac{2T}{\delta}\right] \le \frac{\delta}{2T}.$$

If $m^{(t)} \geq \nu$, then $\sigma^2 \geq \frac{8 \log \frac{4T}{\delta}}{\left(m^{(t)}\right)^2 \epsilon^2}$, and due to the privacy of the Gaussian mechanism,

$$\Pr\left[O = \widehat{u}^{(t)} \wedge \widehat{u}^{(t)} \in F\right] \leq \frac{\delta}{2T} + e^{\epsilon} \Pr\left[O' = \widehat{u}^{(t)} \wedge \widehat{u}^{(t)} \in F\right]. \blacktriangleleft$$

▶ **Lemma 8.** *Each iteration of the Rescaling Phase is $(d\epsilon, \frac{\delta}{2000 \log \frac{\tau}{\beta}})$-DP.*

To show this lemma, first, we show the sensitivity of $c^{(s)}$ in the following claim, whose proof is deferred to the appendix.

▷ Claim 9. Let $m = \min_{t \in [T]} m^{(t)}$. The ℓ_2 sensitivity of $c^{(s)}$ is $\frac{2}{m}$.

Proof of Lemma 8. If $m < \nu$, the Rescaling Phase only happens when for all $t \in [T]$, $\widehat{m}^{(t)} > \nu$. Hence, for any set of outcomes $F \subseteq \mathbb{R}^d$, by union bound

$$\Pr\left[\text{Rescaling happens } \wedge \widehat{c} \in F\right] \leq \Pr\left[\widehat{m}^{(t)} > \nu, \forall t \in T\right] \leq \Pr\left[\widehat{m}^{(t)} > m, \forall t \in T\right]$$
$$\leq \sum_{t \in [T]} \Pr\left[\widehat{m}^{(t)} > m^{(t)}\right] \leq \sum_{t \in [T]} \Pr\left[\mathrm{Lap}\left(\frac{1}{\epsilon}\right) > \frac{1}{\epsilon} \log \frac{2000T \log \frac{\tau}{\beta}}{\delta}\right] \leq \frac{\delta}{2000 \log \frac{\tau}{\beta}}.$$

Next, consider the case $m \geq \nu$. Since $\theta^2 \geq \frac{8}{d^2 \epsilon^2 \nu^2} \log \frac{4000 \log \frac{\tau}{\beta}}{\delta}$ by the privacy guarantee of the Gaussian mechanism

$$\Pr\left[\widehat{c} \in F\right] \leq \frac{\delta}{2000 \log \frac{\tau}{\beta}} + e^{d\epsilon} \Pr\left[\widehat{c}' \in F\right]. \blacktriangleleft$$

Proof of Proposition 5. The algorithm is (ϵ', δ')-DP, following directly from advanced composition. ◀

3.3 Utility analysis

To analyze the runtime and utility of Algorithm 1 we let $\mathcal{B}$ be the unit ball and $\mathcal{P}$ be the feasible region defined by $Ax \geq 0$. We will show the following proposition about the guarantee on the output of Algorithm 1.

▶ **Proposition 10.** *With probability at least $1 - 5\beta$, Algorithm 1 outputs a solution x that satisfies all but $O\left(\frac{\sqrt{d}}{\epsilon} \log^{1.5} \frac{d}{\beta\delta}\right)$ constraints.*

We outline the proof of Proposition 10. First, if in an iteration, Algorithm 1 finds a solution outputted by Algorithm 2, we show that this solution must be correct with probability $\geq 1 - \beta$.

▶ **Lemma 11.** *If Algorithm 2 terminates in the Perceptron Phase and outputs a solution x, then with probability at least $1 - \beta$, x satisfies all but $O\left(\frac{\sqrt{d}}{\epsilon} \log^{1.5} \frac{d}{\beta\delta}\right)$ constraints.*

If Algorithm 1 finds a rescaling vector c by the output of Algorithm 2, we show that the volume of the feasible region inside the unit ball is increased by a constant factor with high probability. To start with, assuming the initial margin parameter is at least ρ_0, [14] give a lower bound on the volume of the initial feasible region inside the unit ball:

▶ **Lemma 12** (Lemma 3 [14]). *Suppose $\max_{x \in \mathcal{P} \cap \mathcal{B}} \min_i \langle \overline{a_i}, x \rangle \geq \rho_0$ then $\mathrm{vol}\left(\mathcal{P} \cap \mathcal{B}\right) = \Omega\left(\rho_0^d\right) \mathrm{vol}(\mathcal{B})$.*

Note that the rescaling operation $A \leftarrow A\left(I - \frac{1}{2}\overline{c} \cdot \overline{c}^\top\right)$ is equivalent to a linear map $F : \mathbb{R}^d \to \mathbb{R}^d$ such that, $F_c(c) = 2c$ and $F_c(x) = x$ for all $x \perp c$. [8] show the following lemma.

▶ **Lemma 13** (Lemma 4 [14]). *Suppose that c satisfies $\frac{1}{\|c\|} \max_{x \in \mathcal{P} \cap \mathcal{B}} \langle c, x \rangle \leq \frac{2}{3\sqrt{d}}$, then $\mathrm{vol}\left(F(\mathcal{P}) \cap \mathcal{B}\right) \geq 1.02 \cdot \mathrm{vol}\left(\mathcal{P} \cap \mathcal{B}\right)$.*

Next, we show that the rescaling vector c outputted by Algorithm 2 satisfies the condition of Lemma 13 with high probability.

▶ **Lemma 14.** *If Algorithm 2 does not return a solution, then with probability at least $1 - \frac{4\beta}{\tau}$, it outputs a rescaling vector c that satisfies $\frac{1}{\|c\|} \max_{x \in \mathcal{P} \cap \mathcal{B}} \langle c, x \rangle \leq \frac{2}{3\sqrt{d}}$.*

Equipped with Lemma 11 - 14, we are now ready prove Proposition 10.

Proof of Proposition 10. The algorithm fails if either it outputs a solution that does not satisfy more than $\Omega\left(\frac{\sqrt{d}}{\epsilon} \log^{1.5} \frac{d}{\beta\delta}\right)$ constraints, or it fails to output a rescaling vector that satisfies the condition of Lemma 13. This happens with probability at most $\beta + \tau \cdot \frac{4\beta}{\tau} = 5\beta$.

Otherwise, in each iteration, either the algorithm outputs a satisfactory solution, or the volume of the feasible region inside the unit ball is increased by a factor at least 1.02, by Lemma 13. Therefore, by Lemma 12, the algorithm stops after $O\left(\log \frac{1}{\rho_0^d}\right) = O\left(d \log \frac{1}{\rho_0}\right)$. ◀

The remaining work is to prove Lemmas 11 and 14.

Proof of Lemma 11. If Algorithm 2 terminates in iteration t of the Perceptron Phase, we have $\widehat{m}^{(t)} \leq \nu$ where $\nu = O\left(\frac{\sqrt{d}}{\epsilon} \log^{1.5} \frac{d}{\beta\delta}\right)$. That is

$$m^{(t)} \leq \nu - \mathrm{Lap}\left(\frac{1}{\epsilon}\right) + \frac{1}{\epsilon} \log \frac{2000T \log \frac{1}{\beta}}{\delta}$$

Here, $m^{(t)}$ is the number of constraints not satisfied by the solution, and we have

$$\Pr\left[m^{(t)} \geq \nu + \frac{\log \frac{1}{\beta}}{\epsilon} + \frac{1}{\epsilon} \log \frac{2000T \log \frac{1}{\beta}}{\delta}\right] \leq \Pr\left[\mathrm{Lap}\left(\frac{1}{\epsilon}\right) \leq -\frac{\log \frac{1}{\beta}}{\epsilon}\right] \leq \beta. \quad ◀$$

To show Lemma 14 we start with the following claim:

▷ Claim 15. $\Pr\left[\left\|x^{(t)}\right\| \leq 10\sqrt{t},\ \forall t \leq T\right] \geq 1 - \frac{2\beta}{\tau}$.

To show Claim 15, we use the following facts about Gaussian random variables.

▶ **Fact 16.** *If $X \sim N(0, \sigma^2)$ then for $a \geq 0$, $\Pr\left[X \geq \sigma a\right] \leq \exp\left(-a^2/2\right)$. If $X \sim N(0, \sigma^2 I) \in \mathbb{R}^d$ then for $u \in \mathbb{R}^d$, $\langle u, X \rangle$ follows $N(0, \sigma^2 \|u\|^2)$ and $\frac{1}{\sigma^2}\|X\|^2$ follows χ-squared distribution with d degrees of freedom. Furthermore, for $a \geq 0$, $\Pr\left[\frac{1}{\sigma^2}\|X\|^2 \geq d + 2\sqrt{da} + 2a\right] \leq \exp\left(-a\right)$.*

Proof of Claim 15. We give an outline of this proof. First, notice that, by the update of $x^{(t)}$, we have with high probability (proof will follow):

$$\left\|x^{(t+1)}\right\|^2 \leq \left\|x^{(t)}\right\|^2 + 2\left\langle x^{(t)}, \eta^{(t)}\right\rangle + 3.$$

Then, the main task is to bound $\sum_s \left\langle x^{(s)}, \eta^{(s)}\right\rangle$. However, $x^{(s)}$ is not a bounded variable, so we cannot directly apply concentration inequalities. Instead, we will define a bounded sequence $(Y^{(s)})$ that behaves like $(\langle x^{(s)}, \eta^{(s)}\rangle)$ and proceed to bound $\sum_{s=1}^{t} Y^{(s)}$. Finally, we will show that with high probability $(Y^{(s)})$ behaves like $(\langle x^{(s)}, \eta^{(s)}\rangle)$ and from there we can bound $\|x^{(t)}\|$.

For each iteration t of Algorithm 2, consider the following event, called M_t that both of the following happen: 1)$\langle u^{(t)}, \eta^{(t)}\rangle \le \sigma\sqrt{2\log\frac{2T\tau}{\beta}} \le \frac{1}{2}$, and 2)$\|\eta^{(t)}\|_2^2 \le 5\sigma^2 d\log\frac{2T\tau}{\beta} \le \frac{1}{\log\frac{2T}{\beta}} \le 1$. Since $u^{(t)}$ is the average of unit length vectors, $\|u^{(t)}\| \le 1$. Then because $\eta^{(t)} \sim N(0, \sigma^2 I)$, $\langle u^{(t)}, \eta^{(t)}\rangle$ follows a $N(0, \kappa^2)$ with $\kappa^2 \le \sigma^2$ (Fact 16). Additionally, $\frac{1}{\sigma^2}\|\eta^{(t)}\|^2$ follows a χ-squared distribution with d degrees of freedom (Fact 16). Therefore, by Fact 16,

$$\begin{aligned}\Pr[M_t] &\ge 1 - \left(\Pr\left[\left\langle u^{(t)}, \eta^{(t)}\right\rangle > \sigma\sqrt{2\log\frac{2T\tau}{\beta}}\right] + \Pr\left[\left\|\eta^{(t)}\right\|^2 > 5\sigma^2 d\log\frac{2T\tau}{\beta}\right]\right)\\ &\ge 1 - \frac{\beta}{T\tau}.\end{aligned}$$

We define the following random variable:

$$Y^{(t)} = \begin{cases}\left\langle x^{(t)}, \eta^{(t)}\right\rangle & \text{if } \cap_{s=0}^{t} M_t \text{ happens and } \left\|x^{(t)}\right\| \le 10\sqrt{t}\\ 0 & \text{otherwise}\end{cases}$$

Due to the symmetry of $\eta^{(t)}$, $\mathbb{E}\left[Y^{(t)} \mid \mathcal{F}_{t-1}\right] = 0$. If $\cap_{s=1}^{t} M_t$ happens and $\|x^{(t)}\| \le 10\sqrt{t}$ we have

$$\left|Y^{(t)}\right| = \left|\left\langle x^{(t)}, \eta^{(t)}\right\rangle\right| \le \left\|x^{(t)}\right\|\left\|\eta^{(t)}\right\| \le 10\sqrt{\frac{t}{\log\frac{2T\tau}{\beta}}}$$

Then $(Y^{(t)})$ forms a bounded martingale difference sequence. By Azuma's inequality,for all t

$$\begin{aligned}\Pr\left[\left|\sum_{s=1}^{t} Y^{(s)}\right| \ge 40t\right] &\le 2\exp\left(-\frac{(40t)^2}{2\sum_{s=1}^{t}\left(20\sqrt{\frac{s}{\log\frac{2T\tau}{\beta}}}\right)^2}\right)\\ &\le 2\exp\left(-\frac{1600t^2\log\frac{2T\tau}{\beta}}{800t^2}\right) \le \frac{\beta}{T\tau}.\end{aligned}$$

Thus by union bound we have with probability $1 - 2\beta/\tau$, M_t happens and $\left|\sum_{s=1}^{t} Y^{(s)}\right| \le 40t$, for all t, simultaneously. When this happens we show by induction that $\|x^{(t)}\| \le 10\sqrt{t}$. This is true for $t = 1$. Suppose that we have $\|x^{(s)}\| \le 10\sqrt{s}$ for all $s \le t$. We have

$$\begin{aligned}\left\|x^{(t+1)}\right\|^2 &= \left\|x^{(t)} + u^{(t)} + \eta^{(t)}\right\|^2\\ &\le \left\|x^{(t)}\right\|^2 + \left\|u^{(t)}\right\|^2 + \left\|\eta^{(t)}\right\|^2 + 2\left\langle x^{(t)}, u^{(t)}\right\rangle + 2\left\langle u^{(t)}, \eta^{(t)}\right\rangle + 2\left\langle x^{(t)}, \eta^{(t)}\right\rangle\\ &\le \left\|x^{(t)}\right\|^2 + 2\left\langle x^{(t)}, \eta^{(t)}\right\rangle + 3.\end{aligned}$$

where in the last inequality, conditioned on M_t, we have $\|\eta^{(t)}\|^2 \le 1$ and $\langle u^{(t)}, \eta^{(t)}\rangle \le \frac{1}{2}$. Since $u^{(t)} = \frac{1}{m^{(t)}}\sum_{i\in S^{(t)}} \overline{a}_i$, and $S^{(t)} = \left\{i \in [n], \langle \overline{a}_i, \overline{x}^{(t)}\rangle \le 0\right\}$, we have $\|u^{(t)}\|^2 \le 1$ and $\langle x^{(t)}, u^{(t)}\rangle \le 0$. Continuing expanding this recursion, we have

$$\left\|x^{(t+1)}\right\|^2 \le \underbrace{\left\|x^{(1)}\right\|^2}_{=0} + 2\sum_{s=1}^{t}\left\langle x^{(s)}, \eta^{(s)}\right\rangle + 3t.$$

Due to the induction hypothesis $\left\|x^{(s)}\right\| \leq 10\sqrt{s}$ for all $s \leq t$, and M_t happens for all t we have then $Y^{(s)} = \left\langle x^{(s)}, \eta^{(s)} \right\rangle$ for all $s \leq t$. It follows that $2\sum_{s=1}^{t} \left\langle x^{(s)}, \eta^{(s)} \right\rangle \leq 2\sum_{s=1}^{t} Y^{(s)} \leq 40t$. Therefore

$$\left\|x^{(t+1)}\right\|^2 \leq 2 \cdot 40t + 3t \leq 100(t+1). \qquad \triangleleft$$

▷ Claim 17. With probability at least $1 - \frac{3\beta}{\tau}$, $\left\|\overline{\lambda A}\right\| \leq \frac{11}{\sqrt{T}}$, where for convenience we define $\overline{\lambda A} = \text{diag}(\overline{\lambda})\overline{A}$ and $\text{diag}(\overline{\lambda})$ is the diagonal matrix obtained from $\overline{\lambda}$.

Proof. First, with probability at least $1 - \frac{\beta}{\tau}$,we have $\left\|\sum_{s=1}^{T} \eta^{(s)}\right\|_2^2 \leq 5\sigma^2 Td \log \frac{T\tau}{\beta} \leq T$. By Claim 15 and union bound, we have, with probability at least $1 - \frac{3\beta}{\tau}$, $\left\|\sum_{s=1}^{T} \eta^{(s)}\right\| \leq \sqrt{T}$ and $\left\|x^{(T)}\right\| \leq 10\sqrt{T}$. We have that

$$\left(\lambda^{(t+1)} - \lambda^{(t)}\right)\overline{A} = u^{(t)} = \widehat{u}^{(t)} - \eta^{(t)} = x^{(t+1)} - x^{(t)} - \eta^{(t)}$$

Hence$\lambda^{(T)}\overline{A} = x^{(T)} - \sum_{s=1}^{T-1} \eta^{(s)}$. Thus $\left\|\lambda^{(T)}\overline{A}\right\| \leq \left\|x^{(T)}\right\| + \left\|\sum_{s=1}^{T-1} \eta^{(T)}\right\| \leq 11\sqrt{T}$. The claim follows due to $\overline{\lambda} = \frac{\lambda^{(T)}}{T}$. $\triangleleft$

▷ Claim 18. Conditioned on $\left\|\overline{\lambda A}\right\| \leq \frac{11}{\sqrt{T}}$, with probability at least 10^{-3}, we have $\left\|\widehat{c}^{(s)}\right\| \geq \frac{3}{16\sqrt{\pi d}}$, in which case $\frac{1}{\left\|\widehat{c}^{(s)}\right\|} \max_{x \in \mathcal{P} \cap \mathcal{B}} \left\langle \widehat{c}^{(s)}, x \right\rangle \leq \frac{2}{3\sqrt{d}}$.

In order to prove Claim 18, we need Lemma 6 from [8].

▶ **Lemma 19** (Lemma 6 [8]). *Let $\lambda \in \mathbb{R}^n \geq 0$ and $A \in \mathbb{R}^{n \times d}$ be such that $\sum_i \lambda_i = 1$ and $\|A_i\| = 1$. Take a random gaussian vector $g \sim N(0, I) \in \mathbb{R}^d$ and let $J = \{i : \langle A_i, g \rangle \geq 0\}$. With probability at least $5 \cdot 10^{-3}$, $\left\|\sum_{i \in J} \lambda_i A_i\right\| \geq \frac{1}{4\sqrt{\pi d}}$.*

Proof. For simplicity, we drop the superscript s. The proof follows similarly to [8]; however, we need to take into account the noise component γ. Since $\gamma \sim N\left(0, \theta^2 I\right)$ with $\theta^2 = \frac{8}{d^2\epsilon^2\nu^2} \log \frac{4000 \log \frac{\tau}{\beta}}{\delta} \leq O\left(\frac{1}{d^3 \log \frac{Td}{\beta\delta}}\right)$, $\frac{1}{\theta^2}\|\gamma\|^2$ follows a χ-squared distribution of d degrees of freedom (Fact 16). By Fact 16, for sufficiently large constant in ν, $\Pr\left[\|\gamma\| \leq \frac{1}{16\sqrt{\pi d}}\right] \geq 1 - \frac{\beta}{\tau}$. Further, by Lemma 19, with probability $\geq 5 \cdot 10^{-3}$ we have $\left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| \geq \frac{1}{4\sqrt{\pi d}}$.

The two events: $\left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| \geq \frac{1}{4\sqrt{\pi d}}$ and $\|\gamma\| \leq \frac{1}{16\sqrt{\pi d}}$ are independent. With probability at least $(1 - \frac{\beta}{\tau}) \cdot 5 \cdot 10^{-3} \geq 10^{-3}$, both events happen. Then,

$$\|\widehat{c}\| = \left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i + \gamma\right\| \geq \left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| - \|\gamma\| \geq \frac{3}{4}\left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| \geq \frac{3}{16\sqrt{\pi d}}.$$

To complete the proof, we now show that, conditioned on the two events $\left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| \geq \frac{1}{4\sqrt{\pi d}}$ and $\|\gamma\| \leq \frac{1}{16\sqrt{\pi d}}$ happening, we have $\frac{1}{\left\|\widehat{c}^{(s)}\right\|_2} \max_{x \in \mathcal{P} \cap \mathcal{B}} \left\langle \widehat{c}^{(s)}, x \right\rangle \leq \frac{2}{3\sqrt{d}}$. Since $\|\widehat{c}\| \geq \frac{3}{16\sqrt{\pi d}}$ and $\|\gamma\| \leq \frac{1}{16\sqrt{\pi d}} \leq \frac{1}{3}\|\widehat{c}\|$, we have $\left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| \geq \|\widehat{c}\| - \|\gamma\| \geq \frac{1}{8\sqrt{\pi d}}$ and $\left\|\sum_{i \in P} \overline{\lambda}_i \overline{a}_i\right\| \leq \|\widehat{c}\| + \|\gamma\| \leq \frac{4}{3}\|\widehat{c}\|$. This gives us

$$\begin{aligned}
\frac{1}{\|\widehat{c}\|}\max_{x\in\mathcal{P}\cap\mathcal{B}}\langle\widehat{c},x\rangle &= \frac{1}{\|\widehat{c}\|}\max_{x\in\mathcal{P}\cap\mathcal{B}}\left\langle\sum_{i\in P}\overline{\lambda}_i\overline{a}_i+\gamma,x\right\rangle\\
&\le \frac{4}{3\left\|\sum_{i\in P}\overline{\lambda}_i\overline{a}_i\right\|}\max_{x\in\mathcal{P}\cap\mathcal{B}}\left\langle\sum_{i\in P}\overline{\lambda}_i\overline{a}_i,x\right\rangle+\frac{16\sqrt{\pi d}}{3}\|\gamma\|\,\|x\|\\
&\le \frac{4}{3}\frac{\left\|\overline{\lambda}A\right\|}{\left\|\sum_{i\in P}\overline{\lambda}_i\overline{a}_i\right\|}+\frac{16\sqrt{\pi d}}{3}\frac{1}{16\sqrt{\pi d}}\\
&\le \frac{4\cdot 11\cdot 8\sqrt{\pi d}}{3\sqrt{T}}+\frac{1}{3\sqrt{d}}\le\frac{2}{3\sqrt{d}}.
\end{aligned}$$

The third inequality is due to $\|\gamma\|\le\frac{1}{16\sqrt{\pi d}}$ and $\|x\|\le 1$ since $x\in\mathcal{B}$. Besides, since $Ax\ge 0$ for $x\in\mathcal{P}$, we also have

$$\max_{x\in\mathcal{P}\cap\mathcal{B}}\left\langle\sum_{i\in P}\overline{\lambda}_i\overline{a}_i,x\right\rangle\le\max_{x\in\mathcal{P}\cap\mathcal{B}}\left\langle\sum_{i\in[n]}\overline{\lambda}_i\overline{a}_i,x\right\rangle\le\max_{\|x\|\le 1}\left\langle\sum_{i\in[n]}\overline{\lambda}_i\overline{a}_i,x\right\rangle=\left\|\overline{\lambda}A\right\|,$$

The last inequality holds for large enough $T=\Theta(d^2)$. ◀

Proof of Lemma 14. The proof of Lemma 14 follows immediately from Claims 17 and 18 when we repeat $O(\log\frac{\tau}{\beta})$ times in the for-loop (Line 11 of Algorithm 2). ◀

3.4 Putting it all together

Proof of Theorem 1. Theorem 1 follows from Propositions 5 and 10 where we set $\delta\leftarrow\frac{\delta}{d}$ and $\epsilon=\Theta\left(\sqrt{\frac{1}{d^3\log\frac{1}{\rho_0}\log\frac{d}{\delta}}}\right)$. ◀

4 When the LP is Not Fully Dimensional

Algorithm 1 can only guarantee the number of dropped constraints is $\tilde{O}(d^2)$ when the LP has a strictly positive margin. In case the LP has zero margin, i.e, tight (equality) constraints, we can use a standard perturbation technique to create a margin without changing the feasibility of the problem. However, the challenge is to output a solution that satisfies the original constraints. The main idea to handle this case, due to [10], is to iteratively identify tight constraints based on the following observation. An LP with integer entries bounded by U in the absolute value has the same feasibility as that of a relaxed LP with $\eta=\frac{1}{2(d+1)((d+1)U)^{d+1}}$ slackness added to each constraint and, if a solution to the relaxed LP violates a constraint in the initial LP, that constraint must be tight. This suggests that we can solve the relaxed LP which has a positive margin, then identify tight constraints and recurse at most d times (we can stop after identifying d linearly independent tight constraints).

Our improvement over the algorithm of [10] is two-fold. First, we use Algorithm 1 as the solver in each iteration, which improves the number of dropped constraints from $\tilde{O}(\frac{d^5}{\epsilon}\log^{1.5}\frac{1}{\rho_0})$ to $O\left(\frac{d^2}{\epsilon}\sqrt{\log\frac{1}{\rho_0}}\log^2\frac{d}{\beta\delta}\right)$. Second, we avoid the fast decrease of the margin during the course of the algorithm, which saves another factor d.

4.1 Synthetic systems of linear equations

During its course, the algorithm needs to identify the set of tight constraints and privatize them for the subsequent use. To this end, we will use the algorithm from [10] for privately generating (or *sanitizing*) synthetic systems of linear equations, stated in the following result.

▶ **Theorem 20** (Theorem C [10]). *There exists an (ϵ, δ)-DP algorithm such that for any system of linear equations $\mathcal{I}$, with probability at least $1-\delta$, the algorithm outputs a system of linear equations $\mathcal{O}$ which satisfies*

1. *Any solution to $\mathcal{I}$ is a solution to $\mathcal{O}$, and*
2. *Each solution to $\mathcal{O}$ satisfies all but $O\left(\frac{d^2}{\epsilon}\log\frac{d}{\delta}\right)$ equations in $\mathcal{I}$.*

Note that for the analysis we also use the fact that this algorithm first privately selects a set of tight constraints in the original LP, then outputs a canonical basis for the subspace defined by these tight constraints.

4.2 Algorithm

The algorithm is as follows. The algorithm runs at most d iterations. In each iteration, the algorithm solves the LP $Ax \le b + \eta\mathbb{1}$ with a set E of sanitized equality constraints that were identified in previous iterations. First, the algorithm selects $k = |E|$ independent columns in E to eliminate k variables then solves the LP without equality constraints using the Algorithm 1 (after homogenizing the LP via the reduction from Section 2). The algorithm terminates if it has found d tight constraints or if the number of constraints violated by the solution is small. Otherwise, the algorithm sanitizes the set of constraints violated by the solution and adds them to E then recurses. Our algorithm is described in Algorithm 3.

Algorithm 3 Private solver for $Ax \le b$.

```
1: Input $A, b$ whose entries are integers bounded by $U$ in absolute value, $\epsilon, \delta, \beta$
2: Let $\eta = \frac{1}{2(d+1)((d+1)U)^{d+1}}$, $\rho = \eta^3$
3: Let $E = \emptyset$ be the set of tight (equality) constraints
4: for $t = 1, \ldots, d$:
5:     Use $k = |E|$ independent columns in $E$ to eliminate $k$ variables of $Ax \le b + \eta\mathbb{1}$ and
       use PrivatePerceptron to solve with margin parameter $\rho$, obtain solution $x^{(t)}$
6:     Let $J_1$ be the subset of constraints $j$ such that $\left\langle a_j, x^{(t)} \right\rangle \le b_j$ and $J_2$ be the subset
       of constraints $j$ such that $b_j < \left\langle a_j, x^{(t)} \right\rangle \le b_j + \eta$
7:     if $|J_2| + \mathrm{Lap}\left(\frac{1}{\epsilon}\right) \le \frac{d^2}{\epsilon} + \log\frac{1}{\delta}$ return $x^{(t)}$
8:     else:
9:         Let $Cx = g$ be the system obtained by sanitizing $A_{J_2}x = b_{J_2}$ and let $s$ be its
           dimension
10:        if $s = d$ or $|J_1| + \mathrm{Lap}\left(\frac{1}{\epsilon}\right) < \frac{d^2}{\epsilon}\log^2\frac{d}{\beta\delta}\sqrt{\log\frac{1}{\rho}} + \log\frac{1}{\delta}$, return a solution of
           $Cx = g$
11:        else:
12:            $E \leftarrow E \cup \{Cx = g\}$; $A \leftarrow A_{J_1}$; $b \leftarrow b_{J_1}$
13: return $\perp$
```

4.3 Analysis

For the privacy guarantee, we have the following lemma.

▶ **Proposition 21.** *The algorithm is (ϵ', δ')-DP for $\epsilon' = 3d\epsilon^2 + \sqrt{3d\epsilon^2\log\frac{1}{\delta}}$; $\delta' = (3d+1)\delta$.*

Proof. This comes directly from advanced composition over d iterations, each of which uses three (ϵ, δ) private mechanisms. ◀

Next, for the utility guarantee, we first show that during the iterations, the entries of the LP (after variable elimination) does not blow up by a factor more than U^d and thereby, we can lower bound the margin of the relaxed LP.

▶ **Lemma 22.** *For each step t of Algorithm 3, consider the LP $Ax \leq b$, $x \geq 0$ with the entries of A, b bounded by U in the absolute value and let E be the set of k equality constraints $Cx = g$ where C has rank k. We can use k independent columns in C and Gaussian elimination to eliminate k variables to obtain an LP $\tilde{A}x \leq \tilde{b}$, $x \geq 0$ with integer entries and $d - k$ variables such that the entries of $\tilde{A}, \tilde{b}$ are bounded by $k^2(k-1)!U^{k+1}$. Furthermore, if the LP $\{Ax \leq b; Cx = g, x \geq 0\}$ is feasible then the LP resulting from the same elimination from $\{Ax \leq b + \eta\mathbb{1}; Cx = g, x \geq 0\}$ has margin parameter $\rho_t \geq \frac{\eta}{((d+1)U)^{2(d+1)}}$.*

Proof. See Appendix. ◀

Equipped with Lemma 22, we can show the bound for the number of dropped constraints.

▶ **Proposition 23.** *With probability at least $1 - d(\beta + \delta)$, Algorithm 3 outputs a solution that satisfies all but $O(\frac{d^{3.5}}{\epsilon} \log^2 \frac{d}{\beta\delta} \sqrt{\log(dU)})$ constraints.*

Proof. See Appendix. ◀

4.4 Proof of Theorem 2

Proof. The proof of Theorem 2 follows from Propositions 21 and 23 where we select $\delta \leftarrow \frac{\delta}{d}$, $\epsilon \leftarrow \frac{1}{\sqrt{d}\sqrt{\log \frac{1}{\delta}}}$ and $\beta \leftarrow \frac{\beta}{d}$. ◀

References

1 Amos Beimel, Shay Moran, Kobbi Nissim, and Uri Stemmer. Private center points and learning of halfspaces. In *COLT*, volume 99 of *Proceedings of Machine Learning Research*, pages 269–282. PMLR, 2019. URL: `http://proceedings.mlr.press/v99/beimel19a.html`.

2 Omri Ben-Eliezer, Dan Mikulincer, and Ilias Zadik. Archimedes meets privacy: On privately estimating quantiles in high dimensions under minimal assumptions. In *NeurIPS*, 2022.

3 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649. IEEE Computer Society, 2015. `doi:10.1109/FOCS.2015.45`.

4 John Dunagan and Santosh S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Math. Program.*, 114(1):101–114, 2008. `doi:10.1007/S10107-007-0095-7`.

5 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006. `doi:10.1007/11761679_29`.

6 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51, 2016. `doi:10.29012/JPC.V7I3.405`.

7 Yue Gao and Or Sheffet. Differentially private approximations of a convex hull in low dimensions. In *ITC*, volume 199 of *LIPIcs*, pages 18:1–18:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.ITC.2021.18`.

8 Rebecca Hoberg and Thomas Rothvoss. An improved deterministic rescaling for linear programming algorithms. In *IPCO*, volume 10328 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 2017. `doi:10.1007/978-3-319-59250-3_22`.

9 Justin Hsu, Aaron Roth, Tim Roughgarden, and Jonathan R. Ullman. Privately solving linear programs. In *ICALP (1)*, volume 8572 of *Lecture Notes in Computer Science*, pages 612–624. Springer, 2014. doi:10.1007/978-3-662-43948-7_51.

10 Haim Kaplan, Yishay Mansour, Shay Moran, Uri Stemmer, and Nitzan Tur. On differentially private linear algebra. *CoRR*, abs/2411.03087, 2024. doi:10.48550/arXiv.2411.03087.

11 Haim Kaplan, Yishay Mansour, Uri Stemmer, and Eliad Tsfadia. Private learning of halfspaces: Simplifying the construction and reducing the sample complexity. In *NeurIPS*, 2020.

12 Andrés Muñoz Medina, Umar Syed, Sergei Vassilvitskii, and Ellen Vitercik. Private optimization without constraint violations. In *AISTATS*, volume 130 of *Proceedings of Machine Learning Research*, pages 2557–2565. PMLR, 2021. URL: http://proceedings.mlr.press/v130/munoz21a.html.

13 Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Locating a small cluster privately. In *PODS*, pages 413–427. ACM, 2016. doi:10.1145/2902251.2902296.

14 Javier Peña and Negar Soheili. A deterministic rescaled perceptron algorithm. *Math. Program.*, 155(1-2):497–510, 2016. doi:10.1007/S10107-015-0860-Y.

A Missing Proofs from Section 3

A.1 From Section 3.2

As a reminder, we let (A, b) and (A', b') be the neighboring LPs. The corresponding computed terms in Algorithm 2 for V' are denoted with the extra prime symbol (for example, $S^{(t)}$ and $S^{(t)\prime}$).

Proof of Claim 7. Fix two neighboring LPs V and V'. Suppose V' has one extra constraint. $S^{(t)\prime}$ can have at most one more constraint so it is immediate that $m^{(t)}$ has ℓ_2 sensitivity 1. We consider the case $S^{(t)\prime}$ has one extra constraint $\overline{a}$,

$$u^{(t)} - u^{(t)\prime} = \left(\frac{1}{m^{(t)}} \sum_{i \in S^{(t)}} \overline{a}_i\right) - \left(\frac{\overline{a}}{m^{(t)}+1} + \frac{1}{m^{(t)}+1} \sum_{i \in S^{(t)}} \overline{a}_i\right)$$
$$= \frac{\sum_{i \in S^{(t)}} \overline{a}_i}{m^{(t)}(m^{(t)}+1)} - \frac{\overline{a}}{m^{(t)}+1}.$$

Then by triangle inequality

$$\left\|u^{(t)} - u^{(t)\prime}\right\| \le \frac{\sum_{i \in S^{(t)}} \|\overline{a}_i\|}{m^{(t)}(m^{(t)}+1)} + \frac{\|\overline{a}\|}{m^{(t)}+1} = \frac{2}{m^{(t)}+1} \le \frac{2}{m^{(t)}}.$$

When V' has one less constraint, we proceed similarly.

$$u^{(t)} - u^{(t)\prime} = \left(\frac{\overline{a}}{m^{(t)}} + \frac{1}{m^{(t)}} \sum_{i \in S^{(t)\prime}} \overline{a}_i\right) - \left(\frac{1}{m^{(t)}-1} \sum_{i \in S} \overline{a}_i\right) = \frac{\sum_{i \in S^{(t)\prime}} \overline{a}_i}{m^{(t)}(m^{(t)}-1)} + \frac{\overline{a}}{m^{(t)}}.$$

Then by triangle inequality

$$\left\|u^{(t)} - u^{(t)\prime}\right\| \le \frac{\sum_{i \in S^{(t)\prime}} \|\overline{a}_i\|}{m^{(t)}(m^{(t)}-1)} + \frac{\|\overline{a}\|}{m^{(t)}} = \frac{2}{m^{(t)}}.$$ ◁

Claim 9. For simplicity, we drop the superscript s for iteration s. Consider the case where V' and P' have one extra constraint a_k. For all $t \in [T]$, for $i \neq k$ we have

$$0 \le \lambda_i^{(t)} - \lambda_i^{\prime(t)} \le \frac{1}{m^{(t)}} - \frac{1}{m^{(t)}+1} = \frac{1}{m^{(t)}\left(m^{(t)}+1\right)} \le \frac{\lambda_i^{\prime(t)}}{m^{(t)}} \le \frac{\lambda_i^{(t)}}{m}.$$

Hence $\left|\overline{\lambda}_i - \overline{\lambda}'_i\right| \le \frac{\overline{\lambda}_i}{m}$. Besides, we also have $\overline{\lambda}'_k \le \frac{1}{m}$. For coordinate j,

$$c_j - c'_j = \sum_{i\in P} \overline{\lambda}_i \overline{a}_{i,j} - \sum_{i\in P} \overline{\lambda}'_i \overline{a}_{i,j} - \overline{\lambda}'_k \overline{a}_{k,j} = \sum_{i\in P} \left(\overline{\lambda}_i - \overline{\lambda}'_i\right) \overline{a}_{i,j} - \overline{\lambda}'_k \overline{a}_{k,j}.$$

Then, to compute the ℓ_2 sensitivity of c, we take $\sum_j \left(c_j - c'_j\right)^2$.

$$\begin{aligned}
\sum_j \left(c_j - c'_j\right)^2 &= \sum_j \left(\sum_{i\in P} \underbrace{\left(\overline{\lambda}_i - \overline{\lambda}'_i\right)}_{\ge 0} \overline{a}_{i,j} - \overline{\lambda}'_k \overline{a}_{k,j}\right)^2 \\
&\le \sum_j \left(\sum_{i\in P} \left(\overline{\lambda}_i - \overline{\lambda}'_i\right) \left|\overline{a}_{i,j}\right| + \overline{\lambda}'_k \left|\overline{a}_{k,j}\right|\right)^2 \\
&\le \sum_j \left(\frac{1}{m} \sum_{i\in P} \overline{\lambda}_i \left|\overline{a}_{i,j}\right| + \frac{1}{m} \left|\overline{a}_{k,j}\right|\right)^2 \\
&\le \frac{2}{m^2} \sum_j \underbrace{\left(\sum_{i=1}^n \overline{\lambda}_i \left|\overline{a}_{i,j}\right|\right)^2}_{\text{Jensen inequality: } \sum_i \overline{\lambda}_i = 1} + \frac{2}{m^2} \underbrace{\sum_j \left|\overline{a}_{k,j}\right|^2}_{=1} \\
&\le \frac{2}{m^2} \sum_j \sum_i \overline{\lambda}_i \left|\overline{a}_{i,j}\right|^2 + \frac{2}{m^2} = \frac{2}{m^2} \sum_i \overline{\lambda}_i \underbrace{\sum_j \left|\overline{a}_{i,j}\right|^2}_{=1} + \frac{2}{m^2} = \frac{4}{m^2}.
\end{aligned}$$

Similarly, consider the case where V and P have one extra constraint a_k compared with the neighbor V'. For all $t \in [T_1]$, for $i \ne k$ we have

$$0 \le \lambda'^{(t)}_i - \lambda^{(t)}_i \le \frac{1}{m^{(t)} - 1} - \frac{1}{m^{(t)}} = \frac{1}{m^{(t)}\left(m^{(t)} - 1\right)} \le \frac{\lambda'^{(t)}_i}{m^{(t)}}.$$

Hence $\left|\overline{\lambda}_i - \overline{\lambda}'_i\right| \le \frac{\overline{\lambda}'_i}{m}$. Besides, we also have $\overline{\lambda}_k \le \frac{1}{m}$. For coordinate j

$$c_j - c'_j = \sum_{i\in P'} \overline{\lambda}_i \overline{a}_{i,j} - \sum_{i\in P'} \overline{\lambda}'_i \overline{a}_{i,j} + \overline{\lambda}_k \overline{a}_{k,j} = \sum_{i\in P'} \left(\overline{\lambda}_i - \overline{\lambda}'_i\right) \overline{a}_{i,j} + \overline{\lambda}_k \overline{a}_{k,j}.$$

Then

$$\begin{aligned}
\sum_j \left(c_j - c'_j\right)^2 &= \sum_j \left(\sum_{i\in P'} \underbrace{\left(\overline{\lambda}'_i - \overline{\lambda}_i\right)}_{\ge 0} \overline{a}_{i,j} + \overline{\lambda}_k \overline{a}_{k,j}\right)^2 \\
&\le \sum_j \left(\sum_{i\in P'} \left(\overline{\lambda}'_i - \overline{\lambda}_i\right) \left|\overline{a}_{i,j}\right| + \overline{\lambda}_k \left|\overline{a}_{k,j}\right|\right)^2 \\
&\le \sum_j \left(\frac{1}{m} \sum_{i\in P'} \overline{\lambda}'_i \left|\overline{a}_{i,j}\right| + \frac{1}{m} \left|\overline{a}_{k,j}\right|\right)^2 \\
&\le \frac{2}{m^2} \sum_j \underbrace{\left(\sum_{i=1}^n \overline{\lambda}'_i \left|\overline{a}_{i,j}\right|\right)^2}_{\text{Jensen inequality: } \sum_i \overline{\lambda}'_i = 1} + \frac{2}{m^2} \underbrace{\sum_j \left|\overline{a}_{k,j}\right|^2}_{=1}
\end{aligned}$$

$$\le \frac{2}{m^2}\sum_j\sum_i \overline{\lambda}'_i\left|\overline{a}_{i,j}\right|^2 + \frac{2}{m^2} = \frac{2}{m^2}\sum_i \overline{\lambda}'_i \underbrace{\sum_j \left|\overline{a}_{i,j}\right|^2}_{=1} + \frac{2}{m^2} = \frac{4}{m^2}. \qquad \triangleleft$$

B Missing Proofs from Section 4

Proof of Lemma 22. Recall that the equality constraint $Cx = g$ is obtained from sanitizing a set of equality constraints with the entries bounded by U in the absolute value. In particular, there is a set of tight constraints $A'x = b'$ in the original LP with entries bounded by U in the absolute value that spans the same subspace as $Cx = g$ – that is, there is an invertible matrix $M \in \mathbb{R}^{k\times k}$ such that $[C \mid g] = M[A' \mid b']$.

Let C_K be k independent columns of C, where we let K be the set of indices of the columns we selected; the set K corresponds to the set of variables that we will eliminate. Eliminating the variables in K in $Ax \le b$ using $Cx = g$ means switching to the new linear constraints $\left(A - A_K C_K^{-1} C\right) x \le b - A_K C_K^{-1} g$. Eliminating the same variables using $A'x = b'$ would get the result $\left(A - A_K {A'}_K^{-1} A'\right) x \le b - A_K {A'}_K^{-1} b'$. Notice that $C_K^{-1} C = \left(MA'_K\right)^{-1} MA' = {A'}_K^{-1} A'$ and $C_K^{-1} g = \left(MA'_K\right)^{-1} g = {A'}_K^{-1} b'$ so the two results are identical.

Therefore, we can think of using $A'x = b'$ for variable elimination instead of using $Cx = g$. Note that the entries of A, b, A', b' are bounded by U and the entries of $\left(A'_K\right)^{-1}$ are fractions with denominator $\det\left(A'_K\right)$, so to maintain the integrality during the elimination, we can multiply each row of A with $\det(A'_K)$. Therefore, the resulting LP has entries bounded by $k^2(k-1)!U^{k+1}$.

We now show the second claim. First, let us show that the LP $\{Ax \le b; Cx = g, x \ge 0\} = \{Ax \le b; A'x = b', x \ge 0\}$ has a solution x such that $x_i \le ((d+1)U)^{d+1}$ for all i. Let x be any vertex solution. By Cramer's rule, x_i is the ratio of two determinants with integer entries $\le U$. We can bound the numerator of this ratio by $((d+1)U)^{d+1}$ and lower bound the denominator by 1.

Next, consider the LP with added slack $\{Ax \le b + \eta\mathbb{1}, A'x = b', x \ge 0\}$. After performing the variable elimination as described above, we obtain an LP $\left\{\tilde{A}x \le \tilde{b} + \eta\mathbb{1}, x \ge 0\right\}$. Finally, we bound the margin of the vertex solution x (restricted to the variables that were not eliminated) for the homogenized version of the latter LP . Since $\tilde{A}, \tilde{b}$ have entries bounded by $k^2(k-1)!U^{k+1}$ and x has entries bounded by $((d+1)U)^{d+1}$, the margin of the homogenized LP is at least

$$\frac{\eta}{\|(x \mid 1)\| \cdot \max_i \left\|[-\tilde{A}_i \mid \tilde{b}_i]\right\|} \ge \frac{\eta}{(d+1)\left(k^2(k-1)!U^{k+1}\right) \cdot ((d+1)U)^{d+1}}$$
$$\ge \frac{\eta}{((d+1)U)^{2(d+1)}}. \qquad \blacktriangleleft$$

To show Lemma 23, we restate the following lemma from [10] shows that tightness in the relaxed system implies tightness in the original system.

▶ **Lemma 24** (Lemma 36 [10]). *For matrices A_1, A_2 with dimension $m_1 \times d$, $m_2 \times d$ and b_1, b_2 being vectors of length m_1, m_2. The entries are integers with upper bound U in the absolute value. For $\eta_2 \ge 0$ being a vector of length m_2 such that the entries of η_2 are bounded by $\frac{1}{2(d+1)((d+1)U)^{d+1}}$. Then if the system*

$$A_1 x \le b_1$$
$$A_2 x = b_2 + \eta_2$$
$$x \ge 0$$

is feasible then the system

$$\begin{aligned} A_1x &\le b_1 \\ A_2x &= b_2 \\ x &\ge 0 \end{aligned}$$

is feasible.

Proof of Lemma 23. In each iteration, the algorithm solves the LP $\tilde{A}x \le \tilde{b} + \eta\mathbb{1}$ with a set of equality constraints $Cx = g$, where $\tilde{A}x \le \tilde{b}$ is a subset of the constraints of the original LP $Ax \le b$. Note that, by construction, $Cx = g$ is equivalent to a set of tight constraints $A'x = b'$ in the original LP. Solving this LP gives us a solution that satisfies

$$\begin{aligned} A_1x &\le b_1 \\ A_2x &= b_2 + \eta_2 \\ A'x &= b' \end{aligned}$$

for a vector $\eta_2 \le \eta\mathbb{1}$, where $A_1x \le b_1$ and $A_2x \le b_2$ are constraints in the original LP. Since all entries of the above LP are bounded by U and $\eta = \frac{1}{2(d+1)((d+1)U)^{d+1}}$, Lemma 24 implies that the LP

$$\begin{aligned} A_1x &\le b_1 \\ A_2x &\le b_2 \\ A'x &= b' \qquad \text{equivalently, } Cx = g \end{aligned}$$

is feasible. The PrivatePerceptron algorithm succeeds with probability $1-\beta$ and the sanitization step succeeds with probability $1-\delta$. Therefore, after each iteration, with probability at least $1-(\beta+\delta)$ the set of tight (equality) constraints is increased by at least 1. Hence, with probability at least $1-d(\beta+\delta)$ the algorithm must terminate after at most d iterations.

By Lemma 22, in the homogenized LP after variable elimination, the margin is at least $\frac{\eta}{((d+1)U)^{2(d+1)}} \ge \eta^3 = \rho$. Therefore, with probability $\ge 1-\beta$, the number of dropped constraints when using PrivatePerceptron in Line 5 is at most

$$O\left(\frac{d^2}{\epsilon}\log^2\frac{d}{\beta\delta}\sqrt{\log\frac{1}{\rho}}\right) = O\left(\frac{d^{2.5}}{\epsilon}\log^2\frac{d}{\beta\delta}\sqrt{\log(dU)}\right).$$

If in iteration t, the algorithm returns at Line 7, the number of dropped constraints in J_2 is at most $\frac{d^2}{\epsilon}$ with probability at least

$$1 - \Pr\left[\mathrm{Lap}\left(\frac{1}{\epsilon}\right) < -\log\frac{1}{\delta}\right] \ge 1-\delta.$$

If the algorithm returns at Line 10, again, the number of dropped constraints is at most $O\left(\frac{d^2}{\epsilon}\log^2\frac{d}{\beta\delta}\sqrt{\log\frac{1}{\rho}}\right)$ with probability $\ge 1-\delta$. By Theorem 20, the number of dropped constraints by sanitization in each iteration is at most $O\left(\frac{d^2}{\epsilon}\log\frac{d}{\delta}\right)$ with probability $\ge 1-\delta$.

Combining these, over all iterations, with probability at least $1-d(\beta+\delta)$, the number of dropped constraints is at most $O\left(\frac{d^{3.5}}{\epsilon}\log^2\frac{d}{\beta\delta}\sqrt{\log(dU)}\right)$. ◀