Oblivious Sketching-based Central Path Method for Linear Programming

Zhao Song [1] **Zheng Yu** [2]

Abstract

In this work, we propose a sketching-based central path method for solving linear programmings, whose running time matches the state of the art results (Cohen et al., 2019b; Lee et al., 2019). Our method opens up the iterations of the central path method and deploys an "iterate and sketch" approach towards the problem by introducing a new coordinate-wise embedding technique, which may be of independent interest. Compare to previous methods, the work (Cohen et al., 2019b) enjoys feasibility while being non-oblivious, and (Lee et al., 2019) is oblivious but infeasible, and relies on *dense* sketching matrices such as subsampled randomized Hadamard/Fourier transform matrices. Our method enjoys the benefits of being both oblivious and feasible, and can use *sparse* sketching matrix (Nelson & Nguyên, 2013) to speed up the online matrix-vector multiplication. Our framework for solving LP naturally generalizes to a broader class of convex optimization problems including empirical risk minimization.

1. Introduction

Linear programming is one of the fundamental models widely used in both practice and theory. It has been extensively applied in many fields such as economics (Tintner, 1955; Dorfman et al., 1987), operations research (Delson & Shahidehpour, 1992), compressed sensing (Donoho, 2006; Candes et al., 2006), medical studies (Mangasarian et al., 1990; 1995), adversarial deep learning (Wong & Kolter, 2018; Weng et al., 2018), etc., due to its simple and intuitive structure. The problem of solving linear programs has been studied since the 19-th century (Sierksma & Zwols, 2015).

[1]School of Mathematics, Institute for Advanced Study, United States [2]Department of Operations Research and Financial Engineering, Princeton University, United States. Correspondence to: Zhao Song <magic.linuxkde@gmail.com>, Zheng Yu <zhengy@princeton.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

Consider solving a general linear program in standard form $\min_{Ax=b,x\geq 0} c^\top x$ of size $A \in \mathbb{R}^{d\times n}$ without redundant constraints. For the case $d = \Omega(n)$ we considered in this paper, the state of the art results take a total running time of $O^*(n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})$[1] to obtain a solution of δ accuracy in current matrix multiplication time (Cohen et al., 2019b; Lee et al., 2019), where ω is the exponent of matrix multiplication whose current value is roughly 2.373 (Williams, 2012; Le Gall, 2014), and α is the dual exponent of matrix multiplication whose current value is 0.31 (Le Gall & Urrutia, 2018). The breakthrough work due to Cohen, Lee, and Song (Cohen et al., 2019b) improves the long standing running time of $O^*(n^{2.5})$ since 1989 (Vaidya, 1989). For the current ω and α, (Cohen et al., 2019b) algorithm takes $O^*(n^{2.373})$ time.

For the current state of the art results, the work (Cohen et al., 2019b) involves a non-oblivious sampling technique, whose sampling set and size changes along the iterations. It avoids the possibilities of implementing expensive calculations in the preprocessing stage and also makes it harder to extend to other classical optimization problems. On the other hand, the work (Lee et al., 2019) only maintains an infeasible update in each iteration and requires the usage of dense sketching matrices, which will ruin the potential sparsity structure of the original linear programs. Thus, a natural question to ask is:

Is there an oblivious and feasible algorithm for solving linear programs in fast running time (i.e. current matrix multiplication time) ?

In this work, we propose a both oblivious and feasible (per iteration)[2] method that solves linear programs in the same running time as the state of the art.

The algorithm we propose is a *sketching-based* short step central path method. The classical short step method follows the central path in the interior of the feasible region. It decreases the complementarity gap uniformly by roughly a $(1 - \frac{1}{\sqrt{n}})$ factor in each iteration and takes $O^*(\sqrt{n})$ itera-

[1]We use notation O^* to hide $n^{o(1)}$ and $\log^{O(1)}(1/\delta)$ factors.

[2]In each iteration, we approximate the central path by solving a linear system. Our approach constructs a randomized oblivious system equation which can be solved exactly. While previous work (Cohen et al., 2019b) constructs a non-oblivious one, and (Lee et al., 2019) doesn't solve the system exactly (in each iteration).

tions to converge. This results in $O^*(\sqrt{n}) \times n = O^*(n^{1.5})$ coordinate updates throughout the algorithm (Vaidya, 1989). Compared to (Cohen et al., 2019b; Lee et al., 2019), our randomized algorithm improves this amount (in (Vaidya, 1989)) of updates via a different approach. We only update a $O^*(\sqrt{n})$-dimensional subspace in each iteration while keeping the same number of iterations $\sqrt{n}$ through an **O**blivious **C**oordinate-wise **E**mbedding (OCE) technique. Thus, our method updates $O^*(n)$ dimensions in total, which is nearly optimal.

The coordinate-wise embedding we introduce in this work is a distribution of matrices $R \in \mathbb{R}^{b_{\text{sketch}} \times n}$ with $b_{\text{sketch}} \ll n$ such that, for any inner product $g^\top h$ between two n-dimensional vector $g, h \in \mathbb{R}^n$, with "high" probability $g^\top R^\top R h$ approximates $g^\top h$ well. In the case of solving linear programs, we approximate the calculation of matrix-vector multiplication Ph in each iteration by $PR^\top Rh$ through OCE, such that the resulting random vector is close to previous one in each coordinate, i.e., $(PR^\top Rh)_i \approx (Ph)_i$ for all $i \in [n]$. Combining with lazy update and low-rank update techniques to maintain the query structure $PR^\top Rh$ for any input vector $h \in \mathbb{R}^n$, we can ensure the new random path is still close to the central path throughout the iterations. Therefore, our method decreases the average running time per iteration while keeping the same number of iterations. Furthermore, the sketching matrix R in our approach can be chosen in an oblivious way since it does not depend on the algorithm updates. Compare to previous work (Lee et al., 2019), our approximation form $PR^\top Rh$ also helps admit a closed form solution in each iteration for solving LP. Thus, our approach takes the advantages of being oblivious and feasible, compared to other state of the art results (Cohen et al., 2019b; Lee et al., 2019).

We state our main result as follows:

Theorem 1.1 (Main result)**.** *Given a linear program $\min_{Ax=b, x\geq 0} c^\top x$ with no redundant constraints. Assume that for any $x \geq 0$ with $Ax = b$, we have $\|x\|_1 \leq R$. Then for any accuracy $\delta \in (0, 1]$, our LP algorithm outputs a vector $x \geq 0$ such that*

$$c^\top x \leq \min_{Ax=b, x\geq 0} c^\top x + \delta \cdot \|c\|_\infty R \quad \textit{and}$$
$$\|Ax - b\|_1 \leq \delta \cdot (R\|A\|_1 + \|b\|_1)$$

in expected time

$$\left(n^{\omega+o(1)} + n^{2.5-\alpha/2+o(1)} + n^{2+1/6+o(1)}\right) \cdot \log(n/\delta).$$

Note that ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication.

Remark 1.2. *For the current value of $\omega \sim 2.38$ and $\alpha \sim 0.31$, our running time becomes $n^{\omega+o(1)} \log(n/\delta)$.*

Our main contributions are summarized as below:

- We propose a new randomized algorithm for solving linear programs that matches the state of the art running time (Cohen et al., 2019b; Lee et al., 2019). Compare to the state of the art works, our approach takes the advantage of being both oblivious and feasible. Our approach also applies to a broader range of sparse random matrices.
- We provide the intuition and rigorous analysis of our proposed approach by studying the key bottlenecks in running time of classical central path method.
- We propose a new sketching technique, named as coordinate-wise embedding, Which is different and more powerful than Johnson-Lindenstrauss (JL) embedding and subspace embedding when applied to iterative optimization method.

1.1. Related works

Linear programming. Linear programmings have been studied for nearly a century. One of the first and most popular LP algorithm is the simplex algorithm (Dantzig, 1947). Despite it works well in practical small size problems, the simplex algorithm is known to be an exponential time algorithm in the worst case of Klee-Minty cube (Klee & Minty, 1972). The first polynomial time algorithm for solving LP is the ellipsoid method (Khachiyan, 1980) proposed by Khachiyan. Although this algorithm runs in polynomial time in theory, but in practice this algorithm runs much slower than the simplex algorithm. The interior point type of methods (Karmarkar, 1984) have both polynomial running time in theory and fast and stable performance in practice. In the case of $d = \Omega(n)$ considered in this work, Karmarkar's algorithm (Karmarkar, 1984) takes $O^*(n^{3.5})$ running time. Then it was improved to $O^*(n^3)$ in the work (Renegar, 1988; Vaidya, 1987). In 1989, Vaidya further proposed an algorithm that takes a running of $O^*(n^{2.5})$. This result hasn't been improved until recent work due to Cohen, Lee and Song (Cohen et al., 2019b). Apart from the square LP case $d = \Omega(n)$, there are also a series of work devoted to study the flat LP case $d \ll n$, for example (Clarkson, 1995; Brand et al., 2020; Chowdhury et al., 2020).

Sketching. Classical sketching methodology proposed by (Clarkson & Woodruff, 2013) is the so-called "sketch and solve". The most standard and well-known applications are linear regression (Clarkson & Woodruff, 2013; Nelson & Nguyên, 2013; Andoni et al., 2018; Clarkson et al., 2019; Song et al., 2019a) and low-rank approximation (Clarkson & Woodruff, 2013; Nelson & Nguyên, 2013; Boutsidis & Woodruff, 2014; Clarkson & Woodruff, 2015b;a; Razenshteyn et al., 2016; Song et al., 2017; 2019b;c). It further generalizes to subspace embeddings (Wang & Woodruff, 2019; Li et al., 2020a), positive semi-definite matrices

(Clarkson & Woodruff, 2017), distance matrices (Indyk et al., 2019), total least regression (Diao et al., 2019b), quantile regression (Li et al., 2020b), tensor regression (Li et al., 2017; Diao et al., 2018; 2019a; Song et al., 2021), tensor decomposition (Song et al., 2019d).

The sketching method we deploy in this work is called "'iterate and sketch" (Song, 2019). The major difference between classical "sketch and solve", and "iterating and sketch" is: the first one only applied the sketch once at very beginning to reduce the dimension of problem, while does not modify the solver; the second one opens up and modifies the solver by applying sketching techniques iteratively in each iteration. The idea of "iterate and sketch" has been applied to a number of problems, e.g. computing John Ellipsoid (Cohen et al., 2019a), Newton method (Pilanci & Wainwright, 2016; 2017), tensor decomposition (Wang et al., 2015; Song et al., 2016), semidefinite programming (Jiang et al., 2020; Huang et al., 2021), training deep neural network (Brand et al., 2021).

Empirical risk minimization Empirical risk minimization (ERM) problem is a fundamental question in statistical machine learning. Extensive literature has been devoted to study this topic (Nesterov, 1983; Vapnik, 1992; Nesterov, 1998; Polyak & Juditsky, 1992; Nemirovski et al., 2009; Nesterov, 2013; Vapnik, 2013). First-order methods including accelerated gradient descent algorithms for ERM are well-developed and studied (Jin et al., 2018; Johnson & Zhang, 2013; Nesterov & Stich, 2017; Xiao & Zhang, 2014; Allen-Zhu, 2018). These rates has a polynomial dependence on the smoothness/strong convexity parameters of the objective in order to achieve a logarithmic $\log(1/\epsilon)$ dependence on the error parameter ϵ.

Notations For a positive integer n, we use $[n]$ to denote set $\{1, 2, \cdots, n\}$. For vectors $x, z \in \mathbb{R}^n$ and parameter $\epsilon \in (0, 1)$, we use $x \approx_\epsilon z$ to denote $(1-\epsilon)z_i \leq x_i \leq (1+\epsilon)z_i,\ \forall i \in [n]$. For any scalar t, we use $a \approx_\epsilon t$ to denote $(1-\epsilon)t \leq a_i \leq (1+\epsilon)t,\ \forall i \in [n]$. Given diagonal matrices $X = \mathrm{diag}(x) \in \mathbb{R}^{n\times n},\ S = \mathrm{diag}(s) \in \mathbb{R}^{n\times n}$, we use $\frac{X}{S}$ to denote the diagonal matrix with $(\frac{X}{S})_{i,i} = x_i/s_i, \forall i \in [n]$.

2. Technique overview

In this section, we discuss the key ideas of our approach based on the classical central path method.

2.1. Short Step Central Path Method

Consider the following standard primal and dual problems of linear programmings:

$$\min_{Ax=b, x\geq 0} c^\top x \text{ (primal)} \quad \text{and} \quad \max_{A^\top y + s = c,\ x,s\geq 0} b^\top y \text{ (dual)}$$

where $A \in \mathbb{R}^{d\times n}$ is full rank with $d = O(n)$. Then (x, y, s) is an optimal solution if and only if it satisfies the following optimality conditions (Vanderbei et al., 2015):

$$\begin{aligned} Ax = b,\ x \geq 0 \quad & \text{(primal feasibility)} \\ A^\top y + s = c,\ s \geq 0 \quad & \text{(dual feasibility)} \\ x_i s_i = 0 \text{ for all } i \quad & \text{(complementary slackness)} \end{aligned}$$

The classical interior point method finds an optimal solution by following the central path in the interior of the feasible region, which is defined as the tuple (x, y, s, t) that satisfies:

$$\begin{aligned} & Ax = b,\ x > 0 \\ & A^\top y + s = c,\ s > 0 \\ & x_i s_i = t \text{ for all } i \end{aligned} \tag{1}$$

where $t > 0$ is called the complementarity gap. It has been shown we can obtain an initialization point on the central path with $t = 1$ according to (Ye et al., 1994). Then in each iteration, the classical algorithm deceases the complementarity gap uniformly from t to ηt with $\eta < 1$, and solves Eq. (1). As t approaches 0, the central path will converge to an optimal solution. The short step central path method approximately solves Eq. (1) by the following linear system:

$$\begin{aligned} X\delta_s + S\delta_x &= \delta_\mu, \\ A\delta_x &= 0, \\ A^\top \delta_y + \delta_s &= 0, \end{aligned} \tag{2}$$

where $X = \mathrm{diag}(x),\ S = \mathrm{diag}(s)$ and we update the solution by $x = x + \delta_x,\ s = s + \delta_s$ and $y = y + \delta_y$. Denote the actual complementarity gap $\mu \in \mathbb{R}^n$ defined under Eq. (2) as $\mu_i = x_i s_i$ for $i \in [n]$. Then Eq. (2) maintains the feasibility conditions while approximately moving the gap from μ to $\mu + \delta_\mu$. As long as the actual complementarity gap μ is always close to the aiming complementarity gap t during the algorithm, we can assure the actual complementarity gap μ will converge to 0 as t goes to 0, which leads us to an optimal solution.

To solve the linear system (2), note when A is full-rank, it has an unique solution explicitly given by:

$$\delta_x = \frac{X}{\sqrt{XS}}(I - P)\frac{1}{\sqrt{XS}}\delta_\mu \text{ and } \delta_s = \frac{S}{\sqrt{XS}}P\frac{1}{\sqrt{XS}}\delta_\mu, \tag{3}$$

where $P = \sqrt{\frac{X}{S}}A^\top \left(A\frac{X}{S}A^\top\right)^{-1} A\sqrt{\frac{X}{S}}$ is an orthogonal projection matrix.

The work of (Vaidya, 1989) shows that we can choose η to be roughly $1 - \frac{1}{\sqrt{n}}$, and the algorithm converges in $O^*(\sqrt{n})$ iterations. Therefore, the total running time needed of solving LP by explicit solution Eq. (3) is $O^*(n^{\omega+1/2})$.

2.2. Sketching-based Central Path Method

In the following subsections, we discuss our approach of sketching-based central path method. In Subsection 2.3, we introduce the coordinate-wise embedding (CE) technique. We discuss the difference between CE and classical sketching techniques, such as Johnson-Lindenstrauss (JL) Lemma and subspace embedding (SE). We also discussed the results of applying common sketching matrices in CE. In Subsection 2.4, we explain why our sketching-based central path method can speed up the computation. In Subsection 2.5, we explain the reason why our sketch-based central path method is feasible and oblivious. In Subsection 2.6, we discuss the projection maintenance needed for the algorithm updates.

Algorithm 1 Main algorithm (simplified)

procedure MAIN($A, b, c, \delta_{\mathrm{lp}}$) ▷ Theorem D.1
  Modify the linear program and obtain an initial x and s according to (Ye et al., 1994)
  ▷ Ensure the initial complementarity gap start with $x_i s_i = 1$
  Initialize: sketching size $b_{\mathrm{sketch}} = O^*(\sqrt{n})$, parameters $\epsilon = O^*(1)$, projection maintenance datastructure mp
  $t \leftarrow 1$ ▷ Initialize the aiming gap t
  while $t > \delta_{\mathrm{lp}}^2/(32n^3)$ **do** ▷ Stop once the precision is good
    $t^{\mathrm{new}} \leftarrow (1 - \frac{\epsilon}{3\sqrt{n}})t$ ▷ Decrease the aiming gap by $1 - 1/\sqrt{n}$ in each iteration
    $\mu \leftarrow xs$ ▷ Actual gap
    $\delta_\mu \leftarrow (\frac{t^{\mathrm{new}}}{t} - 1)xs - \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\nabla \Phi_\lambda(\mu/t-1)}{\|\nabla \Phi_\lambda(\mu/t-1)\|_2}$ ▷ Here $\Phi_\lambda(r) := \sum_{i=1}^n \cosh(\lambda r_i)$ is the potential function characterizing the ℓ_∞ closeness between actual path $\widetilde{\mu}$ and actual path $\widetilde{t}$. We have Φ_λ in the update to help ensure $\widetilde{u} \approx_{0.1} t$.
    $(x^{\mathrm{new}}, s^{\mathrm{new}}) \leftarrow$ STOCHASTICSTEP($\mathrm{mp}, x, s, \delta_\mu, b_{\mathrm{sketch}}, \epsilon$) ▷ Algorithm 2
    $(x, s) \leftarrow (x^{\mathrm{new}}, s^{\mathrm{new}}), t \leftarrow t^{\mathrm{new}}$
  end while
  return an approximate solution of the original linear program according to (Ye et al., 1994) .
end procedure

2.3. Coordinate-wise Embedding

To speed up the classical central path method, we introduce the coordinate-wise embedding (CE) as follows:

Definition 2.1 ((α, β, δ)-coordinate wise embedding (CE))**.** *Given parameters $\alpha, \beta \in \mathbb{R}$ and $\delta \in (0, 1)$, we say a randomized matrix $R \in \mathbb{R}^{b_{\mathrm{sketch}} \times n}$ with distribution Π satisfies (α, β, δ)-coordinate-wise embedding property if for any fixed vector $g, h \in \mathbb{R}^n$, we have*

1. $\mathbf{E}_{R \sim \Pi}[g^\top R^\top Rh] = g^\top h$,
2. $\mathbf{E}_{R \sim \Pi}[(g^\top R^\top Rh)^2] \leq (g^\top h)^2 + \frac{\alpha}{b_{\mathrm{sketch}}} \|g\|_2^2 \|h\|_2^2$,

Algorithm 2 Sketching-based central path step

procedure STOCHASTICSTEP($\mathrm{mp}, x, s, \delta_\mu, b, \epsilon$)
  $w \leftarrow \frac{x}{s}, \widetilde{v} \leftarrow \mathrm{mp}.\textsc{Update}(w)$ ▷ Projection maintenance
  $\overline{x} \leftarrow x\sqrt{\frac{\widetilde{v}}{w}}, \overline{s} \leftarrow s\sqrt{\frac{w}{\widetilde{v}}}$ ▷ It guarantees that $\frac{\overline{x}}{\overline{s}} = \widetilde{v}$ and $\overline{xs} = xs$
  repeat
    $p_x, p_s \leftarrow \mathrm{mp}.\textsc{Query}(\frac{1}{\sqrt{\overline{XS}}}\delta_\mu)$ ▷ Projection maintenance
    $\widetilde{\delta}_s \leftarrow \frac{\overline{S}}{\sqrt{\overline{XS}}}p_s$ ▷ According to (4)
    $\widetilde{\delta}_x \leftarrow \frac{\overline{X}}{\sqrt{\overline{XS}}}p_x$ ▷ According to (4)
  until $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{1}{100 \log n}$ and $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{1}{100 \log n}$
  return $(x + \widetilde{\delta}_x, s + \widetilde{\delta}_s)$
end procedure

3. $\Pr_{R \sim \Pi}\left[|g^\top R^\top Rh - g^\top h| \geq \frac{\beta}{\sqrt{b_{\mathrm{sketch}}}} \|g\|_2 \|h\|_2\right] \leq \delta.$

We remark that the (α, β, δ)-coordinate wise embedding we proposed here is different from the conventional Johnson-Lindenstrauss Lemma (Johnson & Lindenstrauss, 1984) or subspace embedding (Sarlós, 2006) in classical literature as discussed in Section 3.

Several well-known sketching matrices To further concretize our sketching approach, we discuss the following commonly used sketching matrices and their corresponding properties when acting as coordinate-wise embedding and solving LP.

Random Gaussian matrix All entries are sampled from $\mathcal{N}(0, 1/b_{\mathrm{sketch}})$ independently.

SRHT matrix (Lu et al., 2013) Let $R = \sqrt{n/b_{\mathrm{sketch}}} SHD$, where $S \in \mathbb{R}^{b_{\mathrm{sketch}} \times n}$ is a random matrix whose rows are b_{sketch} uniform samples (without replacement) from the standard basis of $\mathbb{R}^n$, $H \in \mathbb{R}^{n \times n}$ is a normalized Walsh-Hadamard matrix, and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose diagonal elements are i.i.d. Rademacher random variables[3].

AMS sketch matrix (Alon et al., 1999) Let $R_{i,j} = h_i(j)$, where $h_1, h_2, \cdots, h_{b_{\mathrm{sketch}}}$ are b_{sketch} random hash functions picking from a random hash family $\mathcal{H} = \{h : [n] \to \{-\frac{1}{\sqrt{b_{\mathrm{sketch}}}}, +\frac{1}{\sqrt{b_{\mathrm{sketch}}}}\}\}$.

Count-sketch matrix (Charikar et al., 2002) Let $R_{h(i),i} = \sigma(i)$ for all $i \in [n]$ and other entries to zero, where $h : [n] \to [b_{\mathrm{sketch}}]$ and $\sigma : [n] \to \{-1, +1\}$ are random hash functions.

Sparse embedding matrix (Nelson & Nguyên, 2013) Let $R_{(j-1)b_{\mathrm{sketch}}/s + h(i,j), i} = \sigma(i,j)/\sqrt{s}$ for all

[3] In this case, we require $\log n$ to be an integer.

$(i,j) \in [n] \times [s]$ and all other entries to zero, where $h : [n] \times [s] \to [b_{\text{sketch}}/s]$ and $\sigma : [n] \times [s] \to \{-1, 1\}$ are random hash functions.

Uniform sampling matrix Let $R = \sqrt{n/b_{\text{sketch}}} SD$, where $S \in \mathbb{R}^{b_{\text{sketch}} \times n}$ is a random matrix whose rows are b_{sketch} uniform samples (without replacement) from the standard basis of $\mathbb{R}^n$, and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose diagonal elements are i.i.d. Rademacher random variables.

Considering an oblivious regime, where the size of sketching b_{sketch} is fixed, we have

Lemma 2.2 (Oblivious coordinate-wise embedding properties)**.** *For above defined sketching matrices, they are of (α, β, δ)-coordinate wise embedding as in Table 1.*

Remark 2.3. *The approach in (Cohen et al., 2019b) behaves similarly as applying uniform sampling matrix in our sketching, which doesn't work in an oblivious setting. Therefore, (Cohen et al., 2019b) needs to modify the sketching size in each iteration. In general, to apply the sketching in an oblivious way, we observe that the sketching matrix should be relatively dense to better concentrate around its expectation, so that we can control the extra perturbation introduced by random sketching in solving linear programming problems.*

2.4. Speeding up central path method through OCE

To speed up the classical central path method, we randomize the calculation of Eq. (3) by

$$\begin{aligned} \widetilde{\delta}_x &= \frac{X}{\sqrt{XS}}(I - P)R^\top R \frac{1}{\sqrt{XS}}\delta_\mu \\ \widetilde{\delta}_s &= \frac{S}{\sqrt{XS}} P R^\top R \frac{1}{\sqrt{XS}}\delta_\mu, \end{aligned} \tag{4}$$

where the random sketching matrix $R \in \mathbb{R}^{b_{\text{sketch}} \times n}$ satisfies the (α, β, δ)-coordinate wise embedding property with $\alpha = O(1)$, $\beta = O(\sqrt{b_{\text{sketch}}})$ and any failure probability $\delta \in (0, 1)$.

The coordinate-wise embedding properties 2.1 ensures the randomized Eq. (4) are well concentrated around original Eq. (3), which implies the new randomized path $\widetilde{\mu}$, will still be near the aiming path t during the algorithm updates. Therefore, given the same decreasing rate of aiming path t as before, we are able to prove $\widetilde{\mu} \approx_{0.1} t$ throughout the iterations. As they converge to zero, we are still able to obtain an optimal solution in $O^*(\sqrt{n})$ iterations.

In terms of running time, note Eq. (4) actually reduces the dimension of previous matrix-vector multiplication from n to b_{sketch}. Assume we can maintain the sketched projection matrix $PR^\top$ in an efficient manner, the calculation in Eq. (4) reduces to $(PR^\top) \cdot u$ for some vector $u \in \mathbb{R}^{b_{\text{sketch}}}$, which is a multiplication between a matrix of size $n \times b_{\text{sketch}}$ and a vector of size b_{sketch} and costs $O(n b_{\text{sketch}})$ running time. Choosing b_{sketch} to be $O^*(\sqrt{n})$, we speed up the updates. We summarize our approach in Algorithm 1, 2. We discuss how to maintain the projection in Section 2.6.

2.5. Feasible central path equation via sketching

To explain the strength of our approach, we discuss the feasible and oblivious advantages of our method over past state of the art results.

The new update rule Eq. (4) can be viewed as an **exact** solution of the following linear system:

$$\begin{aligned} X\widetilde{\delta}_s + S\widetilde{\delta}_x &= \widetilde{\delta}_\mu, \\ A\widetilde{\delta}_x &= 0, \\ A^\top \widetilde{\delta}_y + \widetilde{\delta}_s &= 0, \end{aligned} \tag{5}$$

where

$$\widetilde{\delta}_\mu = \sqrt{XS} R^\top R \frac{1}{\sqrt{XS}} \delta_\mu. \tag{6}$$

Therefore, our approach can also be viewed as an update of a subspace of the complementarity gaps in each iteration, instead of decreasing the complementarity gaps uniformly.

Note in each iteration, our update solves the new linear system (5) **exactly**. Compared to the state of the art approach (Lee et al., 2019) which constructs a solution that solves the linear system **approximately**, the feasibility of our approach helps us to have simpler analysis and also be able to use sparse embedding matrix to prevent ruining the potential sparsity structure of the original linear programs, compared to the usage of dense sketching matrices in the work (Lee et al., 2019).

On the other hand, our method is oblivious since the choice of sketching matrix $R \in \mathbb{R}^{b_{\text{sketch}} \times n}$ **does not depend** on the algorithm updates, which implies we can pick the sketching matrices at the preprocessing stage. While for the state of the art approach (Cohen et al., 2019b), its sampling probability **depends** on the algorithm updates and needs to be calculated on-the-fly.

2.6. Projection maintenance

In this section, we discuss our approach to deal with the second computational bottleneck, i.e., how to maintain the projection after sketching $PR^\top \in \mathbb{R}^{n \times b_{\text{sketch}}}$ in an efficient way, where $P \in \mathbb{R}^{n \times n}$ is the orthogonal projection matrix defined in Eq. (3) and $R \in \mathbb{R}^{b_{\text{sketch}} \times n}$ is a random sketching matrix with appropriate (α, β, δ)-coordinate-wise embedding property.

Sketching matrix	α	β	LP? (Left)	LP? (Right)
Random Gaussian	$O(1)$	$O(\log^{1.5}(n/\delta))$	Yes	Yes
SRHT	$O(1)$	$O(\log^{1.5}(n/\delta))$	Yes	Yes
AMS	$O(1)$	$O(\log^{1.5}(n/\delta))$	Yes	Yes
Count-sketch	$O(1)$	$O(\sqrt{b_{\mathrm{sketch}}}\log(1/\delta)$ or $O(1/\sqrt{\delta})$	No	No
Sparse embedding	$O(1)$	$O(\sqrt{b_{\mathrm{sketch}}/s}\log^{1.5}(n/\delta)$	No†	Yes*
Uniform sampling	$O(n)$	$O(n/\sqrt{b_{\mathrm{sketch}}})$	No	No

Table 1: Summary for different sketching matrices. * A sparse embedding sketching matrix can be used in LP algorithm when it is added on the right and $s = \Omega(\log^3(n/\delta))$. † However when sketching on the left (in (Lee et al., 2019)), additional algorithmic designs are needed to make the algorithm feasible (see Section F.1 for more discussion), and the error of the feasibility part cannot be bounded unless $s = \Omega(b_{\mathrm{sketch}})$.

Let $W := \mathrm{diag}(w) \in \mathbb{R}^n$ denotes the diagonal matrix with $w_i = x_i/s_i$. Then we have $P := \sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W} \in \mathbb{R}^{n\times n}$. Therefore, our final goal of implementing of Eq. (4) reduces to the task of maintaining the query structure which outputs:

$$PR^\top Rh = (PR^\top)\cdot(R\cdot h) = (PR^\top)\cdot u \qquad (7)$$

where $u \in \mathbb{R}^{b_{\mathrm{sketch}}}$.

To achieve this, we have the similar observation as in (Cohen et al., 2019b): W doesn't vary much between two iterations under the sketching approach, which is shown in the following lemma:

Lemma 2.4 (Change of W)**.** *Let w_i and w_i^{new} denote the value x_i/s_i in two consecutive iterations, then we have*

$$\sum_{i=1}^n (\mathbf{E}[\ln w_i^{\mathrm{new}}] - \ln w_i)^2 \le 64\epsilon^2,$$
$$\sum_{i=1}^n (\mathbf{Var}[\ln w_i^{\mathrm{new}}])^2 \le 1000\epsilon^2.$$

for some $0 < \epsilon < 1/(40000\log n)$.

Above observation motivates us to take the benefit of lazy update if w_i only has little changes since we only need to maintain the projection approximately. We discuss two extreme scenarios to illustrate the core ideas: 1) w changes uniformly across all coordinates and 2) w only changes in few coordinates.

In the first case, we use the idea of lazy update. Lemma 2.4 implies the changes of w between two iterations are roughly $w_i^{\mathrm{new}} \approx (1 \pm \frac{1}{\sqrt{n}})w_i$. Therefore, w_i's will only vary by more than a constant and possibly ruin our previous ℓ_∞ closeness after $O^*(\sqrt{n})$ number of iterations. In this case, we only need to update the matrix $PR^\top$ once every $O^*(\sqrt{n})$ iterations, while being "lazy" in any other time. Since the algorithm finishes in $O^*(\sqrt{n})$ iterations, it means we only need to update matrix $PR^\top$ $O^*(1)$ many times, whose total running time is $O^*(n^\omega)$.

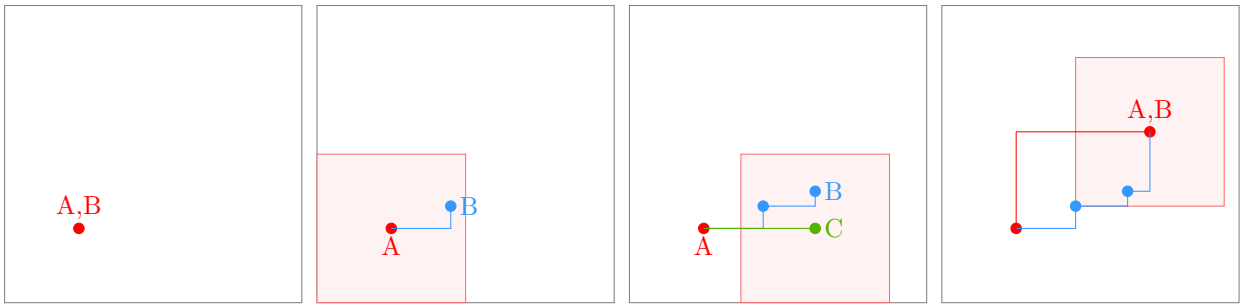
In the second case, we use the idea of low-rank update. Instead of updating $PR^\top$ in each iteration, we directly compute $PR^\top u$ using the Woodbury matrix identity. Since w only changes in few coordinates, we only need to focus on computing the inverse of small matrix instead of the original $n \times n$ matrix. And computing $PR^\top u$ instead of $PR^\top$ speeds up the computation because we only need to do matrix-vector multiplication. As a result, we can output $PR^\top u$ in $O^*(nb_{\mathrm{sketch}})$ time. Recall we choose $b_{\mathrm{sketch}} = O^*(n^{1/2})$. Therefore, the running time for this case is $O^*(n^2)$, which is also within our budget.

For general cases, we combine above techniques together. Specifically, we save the projection matrices $PR^\top$ in our data structure. And in each iteration, three possible cases might happen:

1. Case 1: The current W is close to the one we save W^{save} in every coordinate. Then we stay lazy and output $PR^\top u$ directly;
2. Case 2: The current W differs from W^{save} only in few coordinates, while being close in other coordinates. Then we stay lazy in maintaining the projection $PR^\top$ and output $PR^\top u$ using low-rank updates;
3. Case 3: The current W differs a lot from W^{save} in many coordinates. Then we update the projection $PR^\top$ and calculate $PR^\top u$ exactly;

Note in Case 2, the direct low-rank updates doesn't output $Q_w u$ exactly since the change of W is usually nearly sparse instead of exact sparse. To deal with this issue, we construct a version of W, denoted as $\widetilde{V} = \mathrm{diag}\{\widetilde{v}_1, \cdots, \widetilde{v}_n\}$. We can ensure $\widetilde{v}_i = \Theta(w_i)$ and the update of $\widetilde{V}$ will always be exact sparse. We visualize the above discussion using a chasing example.

Using such approach, we are able to maintain the query structure $PR^\top u$ at point $\widetilde{V}$ in an efficient way. We summarize our datastructure in Algorithm 8, and the update and query scheme in Algorithm 9 and 10. We state the



(a) round 0 (b) round 1 (c) round 2 (d) round 3

Figure 1: Visualization of projection maintenance: we model the task as a game of person A chasing person B, where person A needs to report the approximate location of person B in each round while moving as little as possible. Person A brings a drone C which can only fly in one direction. The location of Person B represents the exact projection matrix, the location of Person A represents the projection matrix we store in the datastructure, the location reported in each round represents the output of our algorithm. In the beginning, they start off at the same location. At round 1, person B moves but is still close to person A. In this case, person A stays idol and reports its location. This case corresponds to the situation that the projection changes little in all coordinates, so we use the idea of lazy updates. At round 2, person B moves far away from A only in one direction. In this case, person A keeps its location and releases drone C to chase person B in the direction where person B moves a lot. And we report the location of the drone C. This case corresponds to the situation that the projection only changes a lot in few coordinates, so we use the idea of low-rank updates while keeping lazy on updating the stored projection matrix. In round 3, person B moves far away from A in all directions. In this case, person A moves to the location of person B and reports its location. This case corresponds to the situation that the projection changes a lot in many coordinates, so we update the stored projection matrix.

Algorithm 3 Projection Maintenance Data Structure

members
  $w \in \mathbb{R}^n$ ▷ Target vector
  $v, \widetilde{v} \in \mathbb{R}^n$ ▷ Approximate vector
  $A \in \mathbb{R}^{d \times n}$
  $M \in \mathbb{R}^{n \times n}$ ▷ Approximate projection matrix
  $Q \in \mathbb{R}^{n \times n^b L}$ ▷ Sketched version approximate projection matrix
  $R_{1,*}, R_{2,*}, \cdots R_{L,*} \in \mathbb{R}^{n^b \times n}$ ▷ Sketching matrices
  $l \in \mathbb{N}_+, L \in \mathbb{N}_+$
  $\epsilon_{\mathrm{mp}} \in (0, 1/4)$ ▷ Tolerance
  $a \in (0, \alpha]$ ▷ Batch Size for Update (n^a)
end members

procedure INITIALIZE($A, w, \epsilon_{\mathrm{mp}}, a$)
  $w \leftarrow w$, $v \leftarrow w$, $\epsilon_{\mathrm{mp}} \leftarrow \epsilon_{\mathrm{mp}}$, $A \leftarrow A$, $a \leftarrow a$
  $M \leftarrow A^\top (AVA^\top)^{-1} A$
  Choosing $R_{1,*}, R_{2,*} \cdots, R_{L,*} \in \mathbb{R}^{n^b \times n}$ to be sketching matrices
  $R \leftarrow [R_{*,1}, R_{*,2} \cdots, R_{*,L}]$
  $Q \leftarrow M\sqrt{V}R^\top$
  $l \leftarrow 1$
end procedure

following main result for projection maintenance and refer to Appendix E for a detailed explanantion:

Theorem 2.5 (Projection maintenance)**.** *Given a number $a \in (0, \alpha)$[4] and sketching matrices $R \in \mathbb{R}^{n^b \times n}$ with $b \in [0, 1]$. We can approximately maintain the projection by*

1. UPDATE(w)*: Output a vector $\widetilde{v}$ such that for all i,*
$$(1 - \epsilon_{\mathrm{mp}})\widetilde{v}_i \leq w_i \leq (1 + \epsilon_{\mathrm{mp}})\widetilde{v}_i.$$
2. QUERY(h)*: Output $\sqrt{\widetilde{V}}A^\top (A\widetilde{V}A^\top)^{-1} A\sqrt{\widetilde{V}}(R^\top)_{*,l}R_{l,*}h$ for the $\widetilde{v}$ outputted by the last call to* UPDATE.

The data structure takes $n^2 d^{\omega-2}$ time to initialize, each call of QUERY(h) *takes time $O^*(n^{1+b} + n^{1+a})$, and the amortized expected time per call of* UPDATE(w) *is $O^*(n^{\omega-1/2} + n^{2-a/2})$.*

Note our approach maintains the projection at point $\widetilde{V} = \overline{X}/\overline{S}$ instead of $W = X/S$, where $\overline{x}_i = x_i\sqrt{\widetilde{v}_i/w_i} \approx_{0.1} x_i$ and $\overline{s}_i = s_i\sqrt{w_i/\widetilde{v}_i} \approx_{0.1} s_i$. Equivalently speaking, instead of solving Eq. (5), we are solving

$$\overline{X}\delta_s + \overline{S}\widetilde{\delta}_x = \widetilde{\delta}_\mu,$$

[4] α is the dual exponent of matrix multiplication, whose current value is roughly 0.31 (Le Gall & Urrutia, 2018).

Algorithm 4 UPDATE

procedure UPDATE(w)
  $y_i \leftarrow \ln w_i - \ln v_i, \forall i \in [n]$
  $r \leftarrow$ the number of indices i such that $|y_i| \geq \epsilon_{\mathrm{mp}}/2$.
  if $r < n^a$ **then**
    $v^{\mathrm{new}} \leftarrow v$
    $M^{\mathrm{new}} \leftarrow M$
    $l \leftarrow l + 1$
  else
    Let $\pi : [n] \to [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
    while $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r \rceil)}| \geq (1 - 1/\log n)|y_{\pi(r)}|$ **do**
      $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
    end while
    $v^{\mathrm{new}}_{\pi(i)} \leftarrow \begin{cases} w_{\pi(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\pi(i)} & i \in \{r+1, \cdots, n\} \end{cases}$
    $\Delta \leftarrow \mathrm{diag}(v^{\mathrm{new}} - v)$ ▷ $\Delta \in \mathbb{R}^{n \times n}$ and $\|\Delta\|_0 = r$
    $\Gamma \leftarrow \mathrm{diag}(\sqrt{v^{\mathrm{new}}} - \sqrt{v})$
    Let $S \leftarrow \pi([r])$ be the first r indices in the permutation.
    Let $M_S \in \mathbb{R}^{n \times r}$ be the r columns from S of M.
    Let $M_{S,S}, \Delta_{S,S} \in \mathbb{R}^{r \times r}$ be the r rows and columns from S of M and Δ.
    $M^{\mathrm{new}} \leftarrow M - M_{*,S} \cdot (\Delta^{-1}_{S,S} + M_{S,S})^{-1} \cdot (M_{*,S})^\top$
    Re-generate R
    $Q^{\mathrm{new}} \leftarrow Q + (M^{\mathrm{new}} \cdot \Gamma) \cdot R^\top + (M^{\mathrm{new}} - M) \cdot \sqrt{V} \cdot R^\top$
    $l \leftarrow 1$
  end if
  $v \leftarrow v^{\mathrm{new}}$
  $M \leftarrow M^{\mathrm{new}}$
  $Q \leftarrow Q^{\mathrm{new}}$
  $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } |\ln w_i - \ln v_i| < \epsilon_{\mathrm{mp}}/2 \\ w_i & \text{otherwise} \end{cases}$
  return $\widetilde{v}$
end procedure

$$A\widetilde{\delta}_x = 0, \qquad (8)$$
$$A^\top \widetilde{\delta}_y + \widetilde{\delta}_s = 0,$$

where $\widetilde{\delta}_\mu = \sqrt{\overline{XS}} R^\top R \frac{1}{\sqrt{\overline{XS}}} \delta_\mu$ as shown in Algorithm 2. And the final running time of our algorithm can be bounded bt $O^*(n^\omega)$.

Remark 2.6. *Our approach for solving LP naturally generalizes to other convex optimization problems of the following form (Lee et al., 2019), including empirical risk minimization:*

$$\min_x \sum_i f_i(A_i x + b_i),$$

where f_i is convex function on $\mathbb{R}^{n_i}$ with $n = \sum_i n_i$. Our algorithm output the solution in time $O^*(n^{2.373} \log(n/\delta))$, where δ is the precision parameter.

Algorithm 5 QUERY

procedure QUERY(h)
  Let $\widetilde{S}$ be the indices i such that $|\ln w_i - \ln v_i| \geq \epsilon_{\mathrm{mp}}/2$.
  $\widetilde{\Delta} \leftarrow \widetilde{V} - V$
  $\widetilde{\Gamma} \leftarrow \sqrt{\widetilde{V}} - \sqrt{V}$
  $p_m \leftarrow \sqrt{\widetilde{V}} \cdot (M_{*,\widetilde{S}}) \cdot (\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot (Q_{\widetilde{S},l} + M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot (R^\top)_{*,l}) \cdot R_{l,*} \cdot h$
  $p_s \leftarrow \sqrt{\widetilde{V}} \cdot (Q_{*,l} + M \cdot \widetilde{\Gamma} \cdot (R^\top)_{*,l}) \cdot R_{l,*} \cdot h - p_m$
  $p_x \leftarrow (R^\top)_{*,l} \cdot R_{l,*} \cdot h - p_s$
  return (p_x, p_s)
end procedure

3. Comparison between Coordinate-wise embedding and classical sketching

In this section, we discuss our proposed coordinate-wise embedding (CE) technique. Despite the seemingly similarity, we show that our technique is fundamentally different from the classical sketching methods including Johnson-Lindenstrauss (JL) embedding and subspace embedding (SE).

3.1. Sketching guarantee comparison

We compare the coordinate-wise embedding proposed in Definition 2.1 with the most well-known sketching guarantees: Johnson-Lindenstrauss embedding (JL) and sparse embedding (SE) as an example. We first list out the definition for JL embedding and SE.

Definition 3.1 (Johnson-Lindenstrauss embedding (JL) (Johnson & Lindenstrauss, 1984))**.** *Given $0 < \epsilon < 1$, a finite point set X in $\mathbb{R}^n$ with $|X| = m$, we say a randomized matrix $R \in \mathbb{R}^{b \times n}$ satisfies Johnson-Lindenstrauss property if $(1-\epsilon) \cdot \|g\|_2^2 \leq \|Rg\|_2^2 \leq (1+\epsilon) \cdot \|g\|_2^2$ for all $g \in X$.*

Definition 3.2 (Subspace embedding (SE) (Sarlós, 2006))**.** *Given $0 < \epsilon < 1$, a matrix $A \in \mathbb{R}^{n \times d}$, we say a randomized matrix $R \in \mathbb{R}^{b \times n}$ satisfies $(1 \pm \epsilon)$ ℓ_2-subspace embedding for the column space of A if $\|RAx\|_2^2 = (1 \pm \epsilon)\|Ax\|_2^2$, for all $x \in \mathbb{R}^d$.*

We note coordinate-wise embedding only works for a fixed vector pair $g, h \in \mathbb{R}^n$, while the Johnson-Lindenstrauss

Statement	Reference	Emb.	ℓ_2 guarantee	Unb.	Var.
Definition 2.1	This paper	CE	for a fixed one	Yes	Yes
Definition 3.1	(Johnson & Lindenstrauss, 1984)	JL	for a fixed set	No	No
Definition 3.2	(Sarlós, 2006)	SE	for a subspace	No	No

Table 2: Summary of the guarantees of different embeddings. The three embeddings give the ℓ_2-norm guarantee for different number of vectors. Coordinate-wise embedding also guarantees the embedding is unbiased, and the variance is bounded. See Section G for more details. "Emb." denotes Embedding. "Unb." denotes Unbiased. "Var." denotes Variance.

embedding stated below works for a finite set of points. Further, the subspace embedding property works for all vectors from the subspace $\mathbb{R}^d$.

The key difference lies on the dimension after sketching. While classical JL embedding and sparse embedding all map high-dimensional vectors $g \in \mathbb{R}^n$ to lower-dimensional vectors

$$g \in \mathbb{R}^n \to Rg \in \mathbb{R}^b$$

the coordinate-wise embedding does not. Instead, coordinate-wise embedding de-sketches the sketched vector and outputs a vector of the original dimension:

$$g \in \mathbb{R}^n \to R^\top \cdot Rg \in \mathbb{R}^n$$

Here, we can view $R : \mathbb{R}^n \to \mathbb{R}^b$ as the sketching step, and $R^\top : \mathbb{R}^b \to \mathbb{R}^n$ as the de-sketching step. Coordinate-wise embedding combines these two steps together and still output a high-dimensional vector.

Correspondingly, we have different guarantees of JL/sparse embedding and coordinate-wise embedding for well-known sketching matrices. As shown in Table 1, coordinate-wise embedding gives a much looser guarantee for the variance bound due to the de-sketching step. Given sketching dimension b, coordinate-wise embedding only guarantees a $O(\frac{n}{b})$-factor variance bound while JL/sparse embedding usually guarantees an $(1+\epsilon)$-factor bound with $\epsilon < 1$.

3.2. Sketch-and-Solve vs. Iterate-and-Sketch

Given the guarantee comparison, we further discuss how coordinate-wise embedding differ from classical sketching approaches from a methodology perspective.

As we discussed, classical sketching maps vectors to a lower dimension while preserve the inner-product structure. Therefore, classical approaches first sketch the problem to a lower dimension and use black-box methods to solve the lower-dimensional problem. The sketching guarantees the sketched problem will produce an approximate solution to the original problem. We call such approach "sketch-and-solve".

Specifically, we consider the least squares problem as an example. Given $A \in \mathbb{R}^{n \times d}$, and $b \in \mathbb{R}^n$, we try to solve

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|_2,$$

whose solution is $x^* = A^\dagger b = (A^\top A)^{-1} A^\top b$ and it takes $O(nd^{\omega-1} + d^\omega)$ running time to compute.

To speed it up, in a over-constrained case where n is much larger than d, the "sketch and solve" approach chooses a $b \times n$ random matrix R from a certain distribution Π on matrices, where $b \ll n$. Consider the following algorithm for least squares regression:

1. Sample a random matrix $R \sim \Pi$.
2. (Sketch) Compute $R \cdot A$ and $R \cdot b$.
3. (Solve) Output the exact solution x' to the regression problem $\min_{x \in \mathbb{R}^d} \|(RA)x - (Rb)\|_2$.

Above approach gives the following guarantee when R to satisfy SE guarantee.

Theorem 3.3 (SE gives approximate regression, Theorem 21 of (Woodruff, 2014))**.** *When $R \in \mathbb{R}^{b \times n}$ used in the "sketch and solve" algorithm satisfies the subspace embedding guarantee with parameters $\epsilon/2$ and δ, then with probability $1 - \delta$, the output x' satisfies*

$$\|Ax' - b\|_2 \leq (1+\epsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2.$$

We remark that classical "sketch-and-solve" approach requires the problem to have certain inner-produce structure which limits its power.

On the contrary, coordinate-wise embedding preserves the vector dimension. It aims to accelerate the matrix-vector or matrix-matrix multiplication instead of reducing the dimension of the problem. Therefore, we go inside each iteration of the algorithm and apply sketching in each step. We call such approach "iterate-and-sketch". In the case of solving linear programs, we note in each step of short central path method, we only need to obtain an approximate solution with loose ℓ_∞ guarantees. Hence, applying sketching in each step will not deteriorate the convergence rate (i.e., the number of iterations) while accelerate the calculation in each iteration, which benefits the overall running time complexity. Such approach does not require any specific property of the problem itself but only requires enough approximation tolerance of each iteration of the iterative algorithm, which could apply to a much broader class of problems.

References

Achlioptas, D. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

Ailon, N. and Chazelle, B. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (STOC)*, pp. 557–563. ACM, 2006.

Allen-Zhu, Z. Natasha 2: Faster non-convex optimization than sgd. In *Advances in neural information processing systems*, pp. 2675–2686, 2018.

Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

Andoni, A., Lin, C., Sheng, Y., Zhong, P., and Zhong, R. Subspace embedding and linear regression with orlicz norm. In *ICML*. https://arXiv.org/pdf/1806.06430, 2018.

Bernstein, S. On a modification of chebyshev's inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924.

Boutsidis, C. and Woodruff, D. P. Optimal cur matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 353–362. ACM, https://arxiv.org/pdf/1405.7910, 2014.

Boutsidis, C., Woodruff, D. P., and Zhong, P. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 236–249. ACM, https://arxiv.org/pdf/1504.06729, 2016.

Brand, J. v. d. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 259–278. SIAM, 2020.

Brand, J. v. d., Lee, Y. T., Sidford, A., and Song, Z. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 775–788, 2020.

Brand, J. v. d., Peng, B., Song, Z., and Weinstein, O. Training (overparametrized) neural networks in near-linear time. In *ITCS*, 2021.

Candes, E. J., Romberg, J. K., and Tao, T. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59 (8):1207–1223, 2006.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. In *Automata, Languages and Programming*, pp. 693–703. Springer, 2002.

Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pp. 493–507, 1952.

Chowdhury, A., London, P., Avron, H., and Drineas, P. Speeding up linear programming using randomized linear algebra. *arXiv preprint arXiv:2003.08072*, 2020.

Clarkson, K. L. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995.

Clarkson, K. L. and Woodruff, D. P. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pp. 81–90. https://arxiv.org/pdf/1207.6365, 2013.

Clarkson, K. L. and Woodruff, D. P. Input sparsity and hardness for robust subspace approximation. In *IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 310–329. https://arxiv.org/pdf/1510.06073, 2015a.

Clarkson, K. L. and Woodruff, D. P. Sketching for m-estimators: A unified approach to robust regression. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 921–939. SIAM, 2015b.

Clarkson, K. L. and Woodruff, D. P. Low-rank psd approximation in input-sparsity time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2061–2072. SIAM, 2017.

Clarkson, K. L., Wang, R., and Woodruff, D. P. Dimensionality reduction for tukey regression. In *ICML*. https://arxiv.org/pdf/1905.05376.pdf, 2019.

Cohen, M. B., Cousins, B., Lee, Y. T., and Yang, X. A near-optimal algorithm for approximating the john ellipsoid. In *COLT*. https://arxiv.org/pdf/1905.11580, 2019a.

Cohen, M. B., Lee, Y. T., and Song, Z. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. https://arxiv.org/pdf/1810.07896.pdf, 2019b.

Dantzig, G. B. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1947.

Delson, J. K. and Shahidehpour, S. Linear programming applications to power system economics, planning and operations. *IEEE Transactions on Power Systems*, 7(3): 1155–1163, 1992.

Diao, H., Song, Z., Sun, W., and Woodruff, D. P. Sketching for kronecker product regression and p-splines. In *AISTATS*. https://arxiv.org/pdf/1712.09473, 2018.

Diao, H., Jayaram, R., Song, Z., Sun, W., and Woodruff, D. P. Optimal sketching for kronecker product regression and low rank approximation. In *NeurIPS*, 2019a.

Diao, H., Song, Z., Woodruff, D., and Yang, X. Total least squares regression in input sparsity time. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2478–2489. https://arxiv.org/pdf/1909.12441.pdf, 2019b.

Donoho, D. L. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006.

Dorfman, R., Samuelson, P. A., and Solow, R. M. *Linear programming and economic analysis*. Courier Corporation, 1987.

Foss, S., Korshunov, D., and Zachary, S. *An introduction to heavy-tailed and subexponential distributions*, volume 6. Springer, 2011.

Haagerup, U. The best constants in the khintchine inequality. *Studia Mathematica*, 70(3):231–283, 1981. URL http://eudml.org/doc/218383.

Hanson, D. L. and Wright, F. T. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42(3):1079–1083, 1971.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Huang, B., Jiang, S., Song, Z., and Tao, R. Solving tall dense sdps in the current matrix multiplication time. *arXiv preprint arXiv:2101.08208*, 2021.

Indyk, P., Vakilian, A., Wagner, T., and Woodruff, D. Sample-optimal low-rank approximation of distance matrices. In *COLT*, 2019.

Jiang, H., Kathuria, T., Lee, Y. T., Padmanabhan, S., and Song, Z. A faster interior point method for semidefinite programming. *arXiv preprint arXiv:2009.10217*, 2020.

Jiang, S., Song, Z., Weinstein, O., and Zhang, H. Faster dynamic matrix inverse for faster lps. In *STOC*, 2021.

Jin, C., Liu, L. T., Ge, R., and Jordan, M. I. On the local minima of the empirical risk. In *Advances in neural information processing systems*, pp. 4896–4905, 2018.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Johnson, W. B. and Lindenstrauss, J. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Kane, D. M. and Nelson, J. Sparser johnson-lindenstrauss transforms. In *SODA*, pp. 1195–1206, 2012.

Karmarkar, N. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing(STOC)*, pp. 302–311. ACM, 1984.

Khachiyan, L. G. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

Khintchine, A. Über dyadische brüche. *Mathematische Zeitschrift*, 18(1):109–116, 1923.

Klee, V. and Minty, G. J. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

Laurent, B. and Massart, P. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pp. 1302–1338, 2000.

Le Gall, F. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation(ISSAC)*, pp. 296–303. ACM, 2014.

Le Gall, F. and Urrutia, F. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pp. 1029–1046. SIAM, 2018.

Lee, Y. T., Song, Z., and Zhang, Q. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*. https://arxiv.org/pdf/1905.04447.pdf, 2019.

Li, X., Haupt, J., and Woodruff, D. Near optimal sketching of low-rank tensor regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3466–3476. https://arxiv.org/pdf/1709.07093.pdf, 2017.

Li, Y., Wang, R., and Woodruff, D. P. Tight bounds for the subspace sketch problem with applications. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1655–1674. SIAM, https://arxiv.org/pdf/1904.05543.pdf, 2020a.

Li, Y., Wang, R., Yang, L., and Zhang, H. Nearly linear row sampling algorithm for quantile regression. In *ICML*, 2020b.

Lu, Y., Dhillon, P., Foster, D. P., and Ungar, L. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems*, pp. 369–377, 2013.

Mangasarian, O. L., Setiono, R., and Wolberg, W. H. Pattern recognition via linear programming: Theory and application to medical diagnosis. 1990.

Mangasarian, O. L., Street, W. N., and Wolberg, W. H. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

Nelson, J. and Nguyên, H. L. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 117–126. IEEE, https://arxiv.org/pdf/1211.1002, 2013.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Nesterov, Y. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 3(4):5, 1998.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Nesterov, Y. and Stich, S. U. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.

Nesterov, Y. E. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Pilanci, M. and Wainwright, M. J. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016.

Pilanci, M. and Wainwright, M. J. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Razenshteyn, I., Song, Z., and Woodruff, D. P. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th Annual Symposium on the Theory of Computing (STOC)*, 2016.

Renegar, J. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

Rudelson, M. and Vershynin, R. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18, 2013.

Sarlós, T. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS) , 21-24 October 2006, Berkeley, California, USA, Proceedings*, pp. 143–152, 2006.

Sierksma, G. and Zwols, Y. *Linear and integer optimization: theory and practice*. CRC Press, 2015.

Song, Z. *Matrix theory: optimization, concentration, and algorithms*. PhD thesis, The University of Texas at Austin, 2019.

Song, Z., Woodruff, D. P., and Zhang, H. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS) 2016, December 5-10, 2016, Barcelona, Spain*, pp. 793–801, 2016.

Song, Z., Woodruff, D. P., and Zhong, P. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1611.00898, 2017.

Song, Z., Wang, R., Yang, L., Zhang, H., and Zhong, P. Efficient symmetric norm regression via linear sketching. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 828–838. https://arxiv.org/pdf/1910.01788.pdf, 2019a.

Song, Z., Woodruff, D., and Zhong, P. Average case column subset selection for entrywise ℓ_1-norm loss. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10111–10121. https://arxiv.org/pdf/2004.07986.pdf, 2019b.

Song, Z., Woodruff, D., and Zhong, P. Towards a zero-one law for column subset selection. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6120–6131. https://arxiv.org/pdf/1811.01442.pdf, 2019c.

Song, Z., Woodruff, D. P., and Zhong, P. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2772–2789. SIAM, https://arxiv.org/pdf/1704.08246.pdf, 2019d.

Song, Z., Woodruff, D. P., Yu, Z., and Zhang, L. Fast sketching of polynomial kernels of polynomial degree. In *ICML*, 2021.

Tintner, G. Stochastic linear programming with applications to agricultural economics. In *Proceedings of the Second Symposium in Linear Programming*, volume 1, pp. 197–228. National Bureau of Standards Washington, DC, 1955.

Tropp, J. A. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011.

Vaidya, P. M. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$arithmetic operations. In *FOCS*. IEEE, 1987.

Vaidya, P. M. Speeding-up linear programming using fast matrix multiplication. In *FOCS*. IEEE, 1989.

Vanderbei, R. J. et al. *Linear programming*. Springer, 2015.

Vapnik, V. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pp. 831–838, 1992.

Vapnik, V. *The nature of statistical learning theory*. Springer science & business media, 2013.

Wang, R. and Woodruff, D. P. Tight bounds for ℓ_p oblivious subspace embeddings. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1825–1843. SIAM, 2019.

Wang, Y., Tung, H.-Y., Smola, A. J., and Anandkumar, A. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 991–999. https://arxiv.org/pdf/1506.04448, 2015.

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. Towards fast computation of certified robustness for relu networks. In *ICML*. https://arxiv.org/pdf/1804.09699.pdf, 2018.

Williams, V. V. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pp. 887–898. ACM, 2012.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

Woodruff, D. P. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Ye, Y., Todd, M. J., and Mizuno, S. An $O(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994.