Fast Algorithms for Interior Point Methods

by

Qiuyi Zhang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Applied Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Spring 2019

Fast Algorithms for Interior Point Methods

Abstract

Fast Algorithms for Interior Point Methods

by

Qiuyi Zhang

Doctor of Philosophy in Applied Mathematics

University of California, Berkeley

Professor Nikhil Srivastava, Chair

Interior point methods (IPM) are first introduced as an efficient polynomial time algorithm to solve linear programs. Since then, they have enjoyed success in general convex optimization with the introduction of self-concordant barriers, replacing the ellipsoid method as the optimizer of choice in many settings. As compared to the ellipsoid method, interior point methods boast a better runtime complexity due to its $O(\sqrt{n})$ iteration complexity, where each iteration requires a linear system solve for the Newton step computation. This implies a naive $O(n^{0.5+\omega})$ total runtime for IPMs, where ω is the exponent of matrix multiplication.

In a recent breakthrough work, [Cohen, Lee, Song'18] showed that we can solve linear programs in the IPM framework in current matrix multiplication time $\widetilde{O}(n^{\omega})$, implying that linear programs are computationally not much harder than matrix inversion. In this thesis, we extend this result to general Empirical Risk Minimization (ERM), showing that many convex optimization problems can be solved as efficiently as matrix inversion.

Specifically, many convex problems in machine learning and computer science share the same form:

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

where f_i are convex functions on $\mathbb{R}^{n_i}$ with constant n_i, $A_i \in \mathbb{R}^{n_i \times d}$, $b_i \in \mathbb{R}^{n_i}$ and $\sum_i n_i = n$. This problem generalizes linear programming and we give an algorithm that runs in time

$$O^*((n^{\omega} + n^{2.5-\alpha/2} + n^{2+1/6}) \log(1/\delta))$$

where α is the dual exponent of matrix multiplication, and δ is the relative accuracy, and O^* hides sub-polynomial terms. Note that the runtime has only a log dependence on the condition numbers or other data dependent parameters and these are captured in δ. For the current bound $\omega \sim 2.38$ and $\alpha \sim 0.31$, our runtime $O^*(n^{\omega} \log(n/\delta))$ matches the current best for solving a dense least squares regression problem, which is a special case of the problem we consider. Very recently, [Alman'18] proved that all the current known techniques can not give a better ω below 2.168, which is larger than our $2 + 1/6$.

Our algorithm proposes two novel concepts, which can be of independent interest :
• We give a robust deterministic central path method, whereas the previous central path is a stochastic central path which updates weights by a random sparse vector.
• We propose an efficient data-structure to maintain the central path of interior point methods even when the weights update vector is dense.

To my parents:

For giving me life and and allowing me to dream

To my Heavenly Father:

For showing me unconditional love and giving me an unshakeable purpose

Contents

List of Figures

List of Tables

Acknowledgments

To my advisor Satish Rao, thank you for allowing me the privilege to work with one of the most brilliant minds in algorithms, yet also one of the most approachable professors I have ever met. Throughout these years, you have been a constant source of inspiration and encouragement in my research and have introduced me to different research directions and projects. Thank you for always believing in me and for guiding me through this journey.

Through my advisor, I also am fortunate to have the opportunity to work extensively with researchers at Google: Rina Panigrahy, Sushant Sachdeva, Ali Rahimi, as well as having stimulating discussions with Manish Purohit, Ravi Kumar, Sreenivas Gollapudi, and Yoram Singer. Thank you for all the ideas and collaborations that arose from my internship. I am grateful to have been supported by the Research community at Google both during my graduate studies and into the future.

I thank my thesis and dissertation committees for their support and hard work: David Aldous, James Demmel, Nikhil Srivastava, Satish Rao. Most of this dissertation is based on joint work with Yintat Lee, and Zhao Song, and I thank my collaborators for all the effort and discussions. I have also collaborated with some fabulous researchers not mentioned already, from whom I have learned a lot and greatly benefited, and am grateful: David Woodruff, Tandy Warnow, Aaron Schild, Frank Ban, Arun Ganesh, Di Wang, Aaron Sy and Herman Bathla.

I thank my fellow graduate students in the mathematics and CS theory group for their friendship and the creation of an encouraging atmosphere: Kenneth Hung, Alex Appleton, Alex Rusciano, Sam Wong, Fotis Iliopoulos, Manuel Sabin, Jingcheng Liu, Peihan Miao, Pasin Manurangsi, Lynn Chua, Grace Dinh, Tarun Kathuria, Seri Khoury, Rachel Lawrence, Siqi Liu, Sidhanth Mohanty, Chinmay Nirkhe, Jonathan Shafer, Nick Spooner, Akshayaram Srinivasan, Elizabeth Yang, and Morris Yau. I also thank all the faculty members for making the theory group such a wonderful place.

My time at Berkeley was made pleasant also because of my friends, especially those in my church communities. I thank all of them for the prayers and adventures, the encouragement and the fun along the journey. Finally I want to thank my family and my God for the unconditional love and support I received in my life.

Chapter 1

Interior Point Methods

1.1 Linear Programs

Linear optimization is one of the typical starting points of any optimization literature, and has unsurprisingly become a well-studied problem in theoretical computer science, convex optimization, and applied mathematics. Let $c, x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ be vectors and our constraint matrix is $A \in \mathbb{R}^{m\times n}$, we want to solve

$$\min_{x\geq 0, Ax=b} c^\top x \tag{1.1}$$

where $x \geq 0$ is an entry-wise inequality. In general, we will assume that the linear program has a feasible solution and that it is bounded. Otherwise, there are standard reductions to ensure this is the case, which we will show later. Furthermore, we assume that A contains no redundant constraints and so $n \geq m$.

Perhaps the first algorithm invented to solve 1.1 is the greedy vertex-walk algorithm of Dantzig in the early 1900's, popularly known as the Simplex Method. Each step of the Simplex Method involves a greedy pivoting procedure: first, find a variable that can be altered in order to decrease the objective function and then, decrease or increase the variable maximally until a constraint would be violated. Theoretically, the Simplex Method suffers from an exponential run-time complexity in the worst case despite a variety of pivot rules [41]. However, it is still heavily used in practice and very recently, a smoothed analysis of the Method with the shadow pivoting rule has shown polynomial average-case runtime complexity bounds under mild perturbations [82, 24].

The breakthrough paper of Karmarkar in 1984 showed that shifting from the combinatorial pivoting rules of the Simplex Method to a more continuous optimization procedure provides a provably polynomial runtime algorithm for linear programming [40]. Because Karmarkar's method always maintain a candidate point x that is in the interior of the feasible polytope, these methods became known as Interior Point Methods (IPM). In general, an IPM follows a sequence of intermediate solutions near what is known as the *central path*.

We now provide a primal-dual view of the central path for linear programs. Later, we will generalize to convex programs and provide a barrier approach to deriving the central path. Associated with our primal formulation 1.1 is the dual linear program:

$$\max_{s\geq 0, A^\top y+s=c} b^\top y \tag{1.2}$$

Let x^* denote the optimal solution of 1.1 and (y^*, s^*) denote that of the dual. By weak duality, for any feasible dual variables (y, s), $b^\top y$ provides a lower bound for $c^\top x^*$ and by strong duality, we have that $b^\top y^* = c^\top x^*$. Therefore, we see that for any feasible primal-dual pair (x, y, s), we upper bound our distance to OPT by the *duality gap*:

$$c^\top x - c^\top x^* \leq c^\top x - b^\top y = c^\top x - x^\top A^\top y = x^\top s$$

In the analysis of convergence, we will use the duality gap, $x^\top s$, as a bound on how close we are to optimum. In fact, we can state the Karush-Kuhn-Tucker (KKT) conditions that ensure optimality.

Theorem 1.1.1 (KKT Conditions for Linear Programs)**.** *[92] x^* is a solution of 1.1 if and only if there exists y^*, s^* such that*

$$\begin{aligned} Ax^* = b,\ x^* \geq 0 & \qquad \textit{(Primal Feasibility)} \\ A^\top y^* + s^* = c,\ s^* \geq 0 & \qquad \textit{(Dual Feasibility)} \\ (x^*)_i (s^*)_i = 0 & \qquad \textit{(Complementary Slackness)} \end{aligned}$$

By relaxing the complementary slackness conditions, we derive the primal dual formulation of the *central path*.

Definition 1.1.2. *(x, y, s) is a point on the central path of 1.1 if there exists $t \geq 0$*

$$\begin{aligned} Ax = b,\ x \geq 0 & \qquad \textit{(Primal Feasibility)} \\ A^\top y + s = c,\ s \geq 0 & \qquad \textit{(Dual Feasbility)} \\ x_i s_i = t & \qquad \textit{(Approximate Complementary Slackness)} \end{aligned}$$

Note that by definition, we can immediately bound the duality gap by $n \cdot t$, so often finding a point near the central path with path parameter $t = O(1/n)$ suffices to solve 1.1 by applying a rounding procedure. It is often easy to find a point on the central path with $t = \Omega(1)$; then the optimization procedure attempts to decrease t. Specifically, if we are at some feasible point (x, y, s), note that we wish to decrease $x_i s_i$ to make progress along the central path. Therefore, we would like our progress step $(\delta_x, \delta_y, \delta_s)$ to satisfy:

$$\begin{aligned} A\delta_x &= 0,\ x+\delta_x \geq 0 \\ A^\top \delta_y + \delta_s &= 0,\ s+\delta_s \geq 0 \\ (x+\delta_x)_i(s+\delta_s)_i &= 0 \end{aligned}$$

By dropping the inequality constraints and linearizing the complementary slackness conditions by dropping the quadratic term, we see that we can solve a system of linear equations to find a good direction of progress. Let X, S represent the diagonal matrices with diagonal entries that are given by the vectors x, s, respectively, then we want to solve:

$$\begin{aligned} A\delta_x &= 0 \\ A^\top \delta_y + \delta_s &= 0 \\ X\delta_s + S\delta_x &= \delta \end{aligned}$$

To center, we want to set $\delta = -XSe$, where e is the all-ones vector. Then, we simply take a step of a certain size in the direction given by this linear system. The stepsize is chosen carefully to remain in our polytope, in order to satisfy our feasibility inequalities. Specifically, note that our linear system admits an easy solution that is given by:

$$\begin{aligned} \delta_x &= X(XS)^{-1/2}(I-P)(XS)^{-1/2}\delta \\ \delta_s &= S(XS)^{-1/2}P(XS)^{-1/2}\delta \end{aligned}$$

where $P = (X/S)^{1/2}A^\top(A\frac{X}{S}A^\top)^{-1}A(X/S)^{1/2}$ is a projection matrix. Notice that when x, s are close enough to the central path, we expect $\delta = -XSe \approx -te$ and $XS \approx tI$. Therefore, since P and $I-P$ are projection matrices, we can bound

$$\|X^{-1}\delta_x\| \leq \|(I-P)e\| \leq \sqrt{n}$$

$$\|S^{-1}\delta_s\| \leq \|Pe\| \leq \sqrt{n}$$

Therefore, we often choose our stepsize α so that setting $\delta = -\alpha \cdot XSe$ with $\alpha = O(1/\sqrt{n})$ ensures that the max relative change in the primal and dual variables are bounded by 1 since $\|X^{-1}\delta_x\|_\infty \leq \|X^{-1}\delta_x\|_2 < O(1)$. This ensures that we always satisfy the inequality constraints that were dropped: $x \geq 0, s \geq 0$. The stepsize bound of $n^{-0.5}$ is intuitively why the analysis will require $O(n^{0.5})$ iterations for an IPM to converge.

In summary, to stay close to the central path, the path-following procedure is split into multiple iterations of two major steps: 1) the progress steps during which we aim to move towards a more optimal point in our polytope by decreasing t and 2) the centering steps where we recenter towards the central path via a linear system solve that finds $(\delta_x, \delta_y, \delta_s)$. Now, we rigorize the notion of staying close to the central path. The measure of proximity to the central path is usually achieved via the ℓ_2 potential:

$$\Phi_2^t(x, y, s) = \sum_i \left(\frac{x_i s_i}{t} - 1\right)^2$$

It can be shown that centering steps can maintain $\Phi_2^t \leq O(1)$ while decreasing t by a factor of $(1 - 1/\sqrt{n})$, giving us the standard $O(n^{0.5})$ iteration analysis of IPM convergence (see [92] for complete proofs). We note that there have been some notable other potentials such as an ℓ_∞-threshold potential[92] and the Tanabe-Todd-Ye potential [95] which is given by

$$\Phi^t(x, y, s) = t \log(x^\top s) - \sum_i \log(x_i s_i)$$

The centering or potential reduction steps are the computational bottleneck of the algorithm and requires a linear system solve at each iteration. Using the framework of Karmarkar, it is not hard to get a runtime complexity of $O(n^{0.5+\omega})$, where we have $O(n^{0.5})$ iterations of linear systems solves that take $O(n^\omega)$ time per iteration and ω is the matrix multiplication constant. Note that the original paper of Karmarkar stated worse runtime complexity bounds.

Since then, a steady line of work has tried to reduce the runtime complexity of solving linear programs via the IPM framework via a two-pronged approach: 1) reducing the iteration complexity and 2) reducing the runtime complexity of each linear system solve. Reducing the iteration complexity of the IPM is notoriously difficult and except for specific linear programs problems or when n is significantly larger than m[51, 19, 60, 1], there is no significant speedup in the iteration complexity of the IPM from the $O(n^{0.5})$ bound despite over decades of continuous research.

Therefore, in this thesis, we focus on the second approach, which is to reduce the runtime of each linear system solve per iteration. The main idea behind possible runtime savings is that over multiple consecutive iterations, the matrix that needs to be inverted has a very nice structure that is changing very slowly over iterations. Therefore, many have exploited this structure to achieve $O(n^2)$ work per iteration [85, 50] by maintaining the inverse of our desired matrix efficiently. This seemed optimal since a matrix vector product must take n^2 time in the worse case and implies a $O(n^{2.5})$ total runtime bound for solving linear programs.

Very recently, novel ideas of stochastic sampling of matrices and randomized linear algebra were introduced to IPMs to further speed up the runtime of the centering steps, leading to a $O^*(n^\omega)$ runtime for solving linear programs as long as $\omega > 2.17$ [18]. The main speedup comes from taking a sparse stochastic step during the centering process that only depends on $\widetilde{O}(n^{0.5})$ coordinates; and over the $O(n^{0.5})$ iterations of the IPM, we would have only moved each coordinate $\widetilde{O}(1)$ times on average. Therefore, given the current matrix multiplication time, the complexity of solving a linear program is the same, up to to sub-polynomial factors, as solving a linear system.

1.2 Self-Concordance and Convex Optimization

We consider the more general convex optimization problem:

$$\min_{x\in K, Ax=b} c^\top x \tag{1.3}$$

for some convex set K. Note that every convex optimization problem can be converted into this form and this problem generalizes our linear program by setting K to be the non-negative orthant. Now, we can rewrite this as

$$\min_{Ax=b} c^\top x + \mathbf{I}_K(x) \tag{1.4}$$

where $\mathbf{I}_K(x) = 0$ if $x \in K$ and ∞ otherwise. The indicator function is rather difficult to optimize over so convex barrier functions were introduced to serve a similar role [14]. The power of IPM is underscored by Nesterov and Nemirovsky, who introduced the idea of using self-concordant barriers to solve 1.4 in an IPM framework.

Definition 1.2.1. *We call a convex function ϕ a ν self-concordant barrier for K if $\mathrm{dom}\phi = K$ and $\phi \to \infty$ at the boundary of K and for any $x \in \mathrm{dom}\phi$ and for any $u \in \mathbb{R}^n$*

$$|D^3\phi(x)[u,u,u]| \le 2\|u\|_x^{3/2} \quad \textit{and} \quad \|\nabla\phi(x)\|_x^* \le \sqrt{\nu}$$

where $\|v\|_x := \|v\|_{\nabla^2\phi(x)}$ and $\|v\|_x^* := \|v\|_{\nabla^2\phi(x)^{-1}}$, for any vector v.

Remark 1.2.2. *It is known that $\nu \ge 1$ for any self-concordant barrier function.*

For the case of the linear program with n variables, we just need a barrier for the non-negative orthant so the typical n-self-concordant barrier is the familiar coordinate-wise log barrier:

$$\phi(x) = -\sum_i \log(x_i)$$

Nesterov and Nemirovsky introduced the concept of following a generalization of the linear programming central path, which are the path of solutions to the following optimization problem:

$$x(t) = \arg\min_{Ax=b} c^\top x + t\phi(x) \tag{1.5}$$

and as before, we want to let $t \to 0$ [68]. The proximity measure to the central path is generalized to a quantity known as the *Newton Decrement*:

$$\lambda_t(x) = \|c + t\nabla\phi(x)\|_{\nabla^2\phi(x)^{-1}}$$

Note that this exactly becomes our ℓ_2 potential, Φ_2, in the case of the log barrier. Finally, they showed that one may use a Newton step to keep $\lambda_t(x) \le O(1)$ throughout the algorithm while

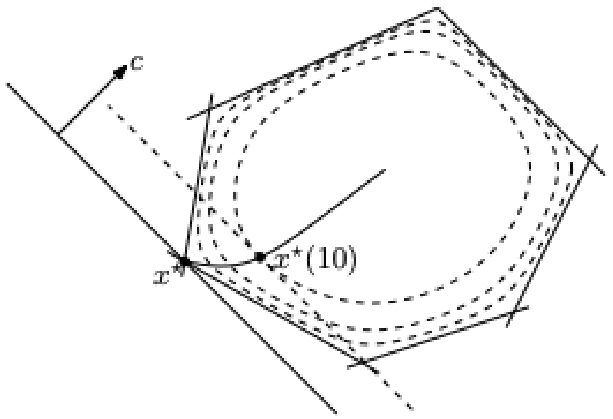

Figure 1.1: Central path for an example linear program as in 1.1. The dashed lines show different contours of the log barrier. Figure is from [14].

reducing t by a factor of $(1 - 1/\sqrt{\nu})$, where ν is the self-concordance parameter of ϕ [68, 14]. For completeness, we note that the Newton step direction is simply $-\nabla^2 f(x)^{-1}\nabla f(x)$, where $f(x)$ is the objective that we are minimizing. Interestingly, the rate at which we decrease t and subsequently the runtimes of these path-following algorithms are usually governed by the self-concordance properties of the barrier functions we use.

Nesterov and Nemirovsky showed that for any open convex set $K \subset \mathbb{R}^n$, there is a $O(n)$ self-concordant barrier function [68]. Moreover, such barriers can be explicitly constructed and furthermore, there are matching lower bounds[15]. Therefore, the IPM framework can be used to give a general optimization procedure for convex programs with strong convergence guarantees. One of the major applications of this generalized framework is for Semi-Definite Programming (SDP) with the log determinant barrier.

Why is self-concordance the natural condition that we want to impose on our barrier functions? The main result we will use about self-concordance is that the norm $\|\cdot\|_x$ is stable when we change x. This is the intuitive reason why second-order methods, such as Newton's Method, work well for centering. For two symmetric matrices A, B, we define $A \preceq B$ if $B - A$ is positive semidefinite matrix, meaning the spectrum of the matrix admits only non-negative eigenvalues. We will see that Hessian does not move much in the spectral sense.

Theorem 1.2.3 (Theorem 4.1.6 in [68]). *If ϕ is a self-concordant barrier and if $\|y - x\|_x < 1$, then we have :*

$$(1 - \|y - x\|_x)^2 \nabla^2 \phi(x) \preceq \nabla^2 \phi(y) \preceq \frac{1}{(1 - \|y - x\|_x)^2} \nabla^2 \phi(x).$$

1.3 Randomized Linear Algebra

Randomized linear algebra has been a tool that has many applications in fast algorithms for numerous optimization problems [91, 55, 54, 33, 71, 65]. Perhaps one of the first uses of randomized linear algebra comes from the dimension-reduction approach of Johnson-Lindenstrauss Lemma, which states that we may embed n points in $\mathbb{R}^d$ into dimension $O(\log(n)/\epsilon^2)$ with ℓ_2 distortion at most ϵ with high probability. Perhaps amazingly, the embedding algorithm is simply a linear transformation and it can be oblivious, meaning it is independent of the position of the n points. The canonical example of such a matrix is a random $k \times d$ Gaussian matrix S, that has S_{ij} being i.i.d. $N(0, 1/k)$.

Indeed if S is such a matrix, then for a vector v, we see that $(Sv)_i = \sum_j S_{ij} v_j$, which is distributed like a Gaussian with mean 0 and variance $\frac{1}{k}\sum_j v_j^2 = \|v\|^2/k$. Therefore, $\mathbf{E}[(Sv)_i^2] = \|v\|^2/k$ for all $1 \le i \le k$ and we have $\mathbf{E}[\|Sv\|^2] = \|v\|^2$. By a simple Chernoff bound, since each $(Sv)_i$ is independent, we see that

$$\Pr\left[\|Sv\|^2 \notin (1-\epsilon, 1+\epsilon)\|v^2\|\right] \le e^{-\Omega(k\epsilon^2)}$$

Therefore, setting $k = \log(n)/\epsilon^2$ suffices for a high probability bound to hold over all n points. While it is useful to preserve the ℓ_2 structure of a discrete point set, we often would want to preserve the ℓ_2 structure of an entire subspace so that we may compute projections and solve approximate regression problems fast. Solving regressions faster will ultimately allow for a speed-up in IPM centering steps, making randomized linear algebra an invaluable tool in reducing the runtime complexity of optimization procedures.

Definition 1.3.1 (Subspace Embedding). *[91] A $(1 \pm \epsilon)$-subspace embedding for the column space of a $n \times d$ matrix A is a $k \times n$ matrix S for which for all $x \in \mathbb{R}^d$*

$$\|SAx\|^2 \in (1-\epsilon, 1+\epsilon)\|Ax\|^2$$

Inspired by the success of random Gaussian projections, one can show that if we let S be a matrix i.i.d. Gaussian entries with an appropriate scale, then it suffices to set $k = \Omega(d/\epsilon^2)$ for S to be a $(1 \pm \epsilon)$-subspace embedding. Intuitively, this scaling of k should be correct since by a standard chaining argument on a $1/2$-net for the d-dimensional subspace [91], we see that we can just apply the Johnson-Lindenstrauss Theorem on a point set of size $O(2^d)$. Therefore, we expect $k = \Omega(d/\epsilon^2)$ for the ℓ_2 structure of an entire d-dimension structure to be preserved.

The utility of subspace embeddings can be illustrated in the following regression problem. Suppose we want to solve linear regression of the form

$$\min_x \|Ax - b\|$$

where A is a $n \times d$ matrix with $n \gg d$. Then, with a good subspace embedding S, we can instead solve

$$\min_x \|S(Ax - b)\|$$

which is a good approximation to our original regression problem. Furthermore, it can be shown that the approximate version of the regression problem can be used, via an fast iterative method along the lines of preconditioned gradient descent, to solve our original regression problem exactly [17]. This implies that if we can compute SA fast, then the regression problem can be solved in $O(d^\omega)$ time.

However, since our current subspace embedding matrix S is a dense Gaussian matrix, the runtime of simply computing the matrix product SA will dominate. Subsequently, a long line of work then proceeded to reduce the runtime of computing the matrix product SA by using a variety of different embeddings S. In [25], S was reduced to have $O(\mathrm{poly}(d)/\epsilon)$ non-zero entries per column and this was later optimized to have $O(d/\epsilon)$ non-zero entries, with tight lower bounds for arbitrary point sets of size $O(2^d)$ [39, 64]. This implies that we can compute SA in time $O(\mathrm{nnz}(A)d/\epsilon)$. Another line of reasoning exploits properties the Fast Fourier and Hadamard Transforms to speed up the product of SA, leading eventually to $O(nd\log(1/\epsilon))$ runtime [91]. Finally, the breakthrough paper of [17] showed that S can be shown to have $O(1)$ non-zero entries per column and that SA can be computed in an optimal $O(\mathrm{nnz}(A))$ runtime. For a more complete exposition of the previous work and proofs, we refer the reader to [91].

Because the IPM centering steps can be re-written as solving a least squares problem, we may use the fast regression solvers of [17] to speed up our computations to depend mainly on the rank of our constraint matrix, as was done in [52, 71]. Note that the usage of randomized linear algebra, before our proposed result, focused on preserving the ℓ_2 structure of the reduced linear system and thus could only reduce to $\Omega(d)$ dimensions at best.

Chapter 2

Empirical Risk Minimization in Matrix Multiplication Time

2.1 Introduction

Empirical Risk Minimization (ERM) problem is a fundamental question in statistical machine learning. There are a huge number of papers that have considered this topic [69, 87, 72, 67, 12, 13, 66, 61, 30, 48, 38, 88, 80, 28, 27, 32, 29, 81, 96, 97, 98, 34, 62, 70, 2, 23, 37][1] as almost all convex optimization machine learning can be phrased in the ERM framework [78, 87]. While the statistical convergence properties and generalization bounds for ERM are well-understood, a general runtime bound for general ERM is not known although fast runtime bounds do exist for specific instances [1].

Examples of applications of ERM include linear regression, LASSO [84], elastic net [99], logistic regression [22, 36], support vector machines [21], ℓ_p regression [16, 26, 15, 1], quantile regression [42, 44, 43], AdaBoost [31], kernel regression [63, 89], and mean-field variational inference [94].

The classical Empirical Risk Minimization problem is defined as

$$\min_x \sum_{i=1}^{m} f_i(a_i^\top x + b_i)$$

where $f_i : \mathbb{R} \to \mathbb{R}$ is a convex function, $a_i \in \mathbb{R}^d$, and $b_i \in \mathbb{R}$, $\forall i \in [m]$. Note that this formulation also captures most standard forms of regularization as well.

Letting $y_i = a_i^\top x + b_i$, and $z_i = f_i(a_i^\top x + b_i)$ allows us to rewrite the original problem in the

[1]Feel free to notify/email us for missing reference, we are happy to add/remove.

following sense,

$$\begin{aligned}\min_{x,y,z} \quad & \sum_{i=1} z_i \\ \text{s.t.} \quad & Ax + b = y \\ & (y_i, z_i) \in K_i = \{(y,z) : f(y) \le z\}, \forall i \in [m]\end{aligned}$$

We can consider a more general version where dimension of K_i can be arbitrary, e.g. n_i. Therefore, we come to study the general n-variable form

$$\min_{x \in \prod_{i=1}^m K_i, Ax=b} c^\top x$$

where $\sum_{i=1}^m n_i = n$. In this thesis, we show that as long as $n_i = O(1)$, we may extend the work of [18] to show that the general form of our problem can be solved in current matrix multiplication time.

Theorem 2.1.1 (Main result, informal version of Theorem 3.3.3)**.** *Given a matrix $A \in \mathbb{R}^{d \times n}$, two vectors $b \in \mathbb{R}^d$, $c \in \mathbb{R}^n$, and m compact convex sets $K_1, K_2, \cdots, K_m$. Assume that there is no redundant constraints and $n_i = O(1)$, $\forall i \in [m]$. There is an algorithm (procedure* MAIN *in Algorithm 6) that solves*

$$\min_{x \in \prod_{i=1}^m K_i, Ax=b} c^\top x$$

up to δ precision and runs in expected time

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

where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication.

For the current value of $\omega \sim 2.38$ [90, 46] and $\alpha \sim 0.31$ [47], the expected time is simply $n^{\omega+o(1)} \widetilde{O}(\log(\frac{n}{\delta}))$.

Remark 2.1.2. *More precisely, when n_i is super constant, our running time depends polynomially on $\max_{i \in [m]} n_i$ (but not exponential dependence).*

Also note that our runtime depends on diameter, but logarithmically to the diameter. So, it can be applied to linear program by imposing an artificial bound on the solution.

Related Work

First-order algorithms for ERM are well-studied and a long series of accelerated stochastic gradient descent algorithms have been developed and optimized [68, 38, 93, 79, 32, 56, 59, 8, 75, 77, 7, 76, 62, 57, 53, 6, 3, 5, 4]. However, these rates depend polynomially on the Lipschitz constant of ∇f_i and in order to achieve a $\log(1/\epsilon)$ dependence, the runtime will also have to depend on the strong convexity of the $\sum_i f_i$. In this paper, we want to focus on algorithms that depend logarithmically on diameter/smoothness/strong convexity constants, as well as the error parameter ϵ. Note that gradient descent and a direct application of Newton's method do not belong to these class of algorithms, but for example, interior point method and ellipsoid method does.

Therefore, in order to achieve high-accuracy solutions for non-smooth and non strongly convex case, most convex optimization problems will rely on second-order methods, often under the general interior point method (IPM) or some sort of iterative refinement framework. So, we note that our algorithm is thus optimal in this general setting since second-order methods require at least n^ω runtime for general matrix inversion. Indeed, note that if we can solve linear regression faster than n^ω time, then matrix inversion would also be sped up.

Our algorithm applies the interior point method framework to solve ERM. The most general interior point methods require $O(\sqrt{n})$-iterations of linear system solves [68], requiring a naive runtime bound of $O(n^{\omega+1/2})$. Using the inverse maintenance technique [86, 18], one can improve the running time for LP to $O(n^\omega)$. This essentially implies that almost all convex optimization problems can be solved, up to subpolynomial factors, as fast as linear regression or matrix inversion!

The specific case of ℓ_2 regression can be solved in $O(n^\omega)$ time since the solution is explicitly given by solving a linear system. In the more general case of ℓ_p regression, [15] proposed a $\widetilde{O}_p(n^{|1/2-1/p|})$-iteration iterative solver with a naive $O(n^\omega)$ system solve at each step. Recently, [1] improved the runtime to $\widetilde{O}_p(n^{\max(\omega,7/3)})$, which is current matrix multiplication time as $\omega > 7/3$. However, both these results depend exponentially on p and fail to be impressive for large p. Otherwise, we are unaware of other ERM formulations that have have general runtime bounds for obtaining high-accuracy solutions.

Recently several works [10, 11, 9] try to show the limitation of current known techniques for improving matrix multiplication time. Alman and Vassilevska Williams [11] proved limitations of using the Galactic method applied to many tensors of interest (including Coppersmith-Winograd tensors [20]). More recently, Alman [9] proved that by applying the Universal method on those tensors, we cannot hope to achieve any running time better than $n^{2.168}$ which is already above our $n^{2+1/6}$.

2.2 Overview of Techniques

In this section, we discuss the key ideas in this paper. Generalizing the stochastic sparse update approach of [18] to our setting is a natural first step to speeding up the matrix-vector multiplication that is needed in each iteration of the interior point method. In linear programs, maintaining approximate complementary slackness means that we maintain x, s to be close multiplicatively to the central path under some notion of distance. However, the generalized notion of complementary slackness requires a barrier-dependent notion of distance. Specifically, if $\phi(x)$ is a barrier function, then our distance is now defined as our function gradient being small in a norm depending on $\nabla^2\phi(x)$. One key fact of the stochastic sparse update is that the variance introduced does not perturb the approximation too much, which requires understanding the second derivative of the distance function. For our setting, this would require bounding the 4th derivative of $\phi(x)$, which may not exist for self-concordant functions. So, the stochastic approach may not work algorithmically (not just in the analysis) if $\phi(x)$ is assumed to be simply self-concordant. Even when assumptions on the 4th derivative of $\phi(x)$ are made, the analysis will become significantly more complicated due to the 4th derivative terms. To avoid these problems, the main contributions of this thesis is to 1) introduce a robust version of the central path and 2) exploit the robustness via sketching to apply the desired matrix-vector multiplication fast.

More generally, our main observation is that one can generally speed up an iterative method using sketching if the method is robust in a certain sense. To speed up interior point methods, in Section 2.4 and 2.5, we give a robust version of the interior point method; and in Section 3.1, we give a data structure to maintain the sketch; and in Section 3.3, we show how to combine them together. We provide several basic notations and definitions for numerical linear algebra in Section 2.3. In Section 3.4, we provide some classical lemmas from the literature of interior point methods. In Section 3.2, we prove some basic properties of the sketching matrix. Now, we first begin with an overview of our robust central path and then proceed with an overview of sketching iterative methods.

Central Path Method

We consider the following optimization problem

$$\min_{x \in \prod_{i=1}^m K_i, Ax=b} c^\top x \tag{2.1}$$

where $\prod_{i=1}^m K_i$ is the direct product of m low-dimensional convex sets K_i. We let x_i be the i-th block of x corresponding to K_i. Interior point methods consider the path of solutions to the following optimization problem:

$$x(t) = \arg\min_{Ax=b} c^\top x + t \sum_{i=1}^m \phi_i(x_i) \tag{2.2}$$

where $\phi_i : K_i \to \mathbb{R}$ are self-concordant barrier functions. This parameterized path is commonly known as the *central path*. Many algorithms solve the original problem (2.1) by following the central path as the path parameter is decreased $t \to 0$. In practice, most ERM problems lend themselves to explicit $O(1)$ self-concordant barriers for majority of the convex functions people use. For example, for the set $\{x : \|x\| < 1\}$, we use $-\log(1 - \|x\|^2)$; for the set $\{x : x > 0\}$, we use $-\log(x)$, and so on. That is the reason why we assume the gradient and hessian can be computed in $O(1)$ time. Therefore, we assume a ν_i self-concordant barrier ϕ_i is provided and that we can compute $\nabla \phi_i$ and $\nabla^2 \phi_i$ in $O(1)$ time.

In general, we can simply think of ϕ_i as a function penalizing any point $x_i \notin K_i$. It is known how to transform the original problem (2.1) by adding $O(n)$ many variables and constraints so that

- The minimizer $x(t)$ at $t = 1$ is explicitly given.
- One can obtain an approximate solution of the original problem using the minimizer at small t in linear time.

For completeness, we show how to do it in Lemma 3.4.2. Therefore, it suffices to study how we can move efficiently from $x(1)$ to $x(\epsilon)$ for some tiny ϵ where $x(t)$ is again the minimizer of the problem (2.2).

Robust Central Path

In the standard interior point method, we use a tight ℓ_2-bound to control how far we can deviate from $x(t)$ during the entirety of the algorithm. Specifically, if we denote $\gamma_i^t(x_i)$ is the Newton Decrement in each block coordinate x_i at path parameter t, then as we let $t \to 0$, the old invariant that we are maintaining is,

$$\Phi_{\text{old}}^t(x) = \sum_{i=1}^{m} \gamma_i^t(x_i)^2 \leq O(1)$$

It can be shown that a Newton step in the standard direction will allow for us to maintain Φ_{old}^t to be small even as we decrease t by a multiplicative factor of $O(m^{-1/2})$ in each iteration, thereby giving a standard $O(\sqrt{m})$ iteration analysis. Therefore, the standard approach can be seen as trying to remain within a small ℓ_2 neighborhood of the central path by centering with Newton steps after making small decreases in the path parameter t. Note however that if each γ_i can be perturbed by an error that is $\Omega(m^{-1/2})$, $\Phi_{\text{old}}^t(x)$ can easily become too large for the potential argument to work.

To make our analysis more robust, we introduce a robust version that maintains the soft-max potential:

$$\Phi_{\text{new}}^t(x) = \sum_{i=1}^{m} \exp(\lambda \gamma_i^t(x_i)) \leq O(m)$$

for some $\lambda = \Theta(\log m)$. The *robust central path* is simply the region of all x that satisfies our potential inequality. We will specify the right constants later but we always make λ large enough to ensure that $\gamma_i \leq 1$ for all x in the robust central path. Now note that a ℓ_∞ perturbation of γ translates into a small multiplicative change in Φ^t, tolerating errors on each γ_i of up to $O(1/\text{poly}\log(n))$.

However, maintaining $\Phi_{\text{new}}^t(x) \leq O(m)$ is not obvious because the robust central path is a much wider region of x than the typical ℓ_2-neighborhood around the central path. We will show later how to modify the standard Newton direction to maintain $\Phi^t_{\text{new}}(x) \leq O(m)$ as we decrease t. Specifically, we will show that a variant of gradient descent of Φ^t_{new} in the Hessian norm suffices to provide the correct guarantees.

Speeding up via Sketching

To motivate our sketching algorithm, we consider an imaginary iterative method

$$z^{(k+1)} \leftarrow z^{(k)} + P \cdot F(z^{(k)})$$

where P is some dense matrix and $F(z)$ is some simple formula that can be computed efficiently in linear time. Note that the cost per iteration is dominated by multiplying P with a vector, which takes $O(n^2)$ time. To avoid the cost of multiplication, instead of storing the solution explicitly, we store it implicitly by $z^{(k)} = P \cdot u^{(k)}$. Now, the algorithm becomes

$$u^{(k+1)} \leftarrow u^{(k)} + F(P \cdot u^{(k)}).$$

This algorithm is as expensive as the previous one except that we switch the location of P. However, if we know the algorithm is robust under perturbation of the $z^{(k)}$ term in $F(z^{(k)})$, we can instead do

$$u^{(k+1)} \leftarrow u^{(k)} + F(R^\top RP \cdot u^{(k)})$$

for some random Gaussian matrix $R : \mathbb{R}^{b \times n}$. Note that the matrix RP is fixed throughout the whole algorithm and can be precomputed. Therefore, the cost of per iteration decreases from $O(n^2)$ to $O(nb)$.

For our problem, we need to make two adjustments. First, we need to sketch the change of z, that is $F(P \cdot u^{(k)})$, instead of $z^{(k)}$ directly because the change of z is smaller and this creates a smaller error. Second, we need to use a fresh random R every iteration to avoid the randomness dependence issue in the proof. For the imaginary iterative process, it becomes

$$\begin{aligned} \overline{z}^{(k+1)} &\leftarrow \overline{z}^{(k)} + R^{(k)\top} R^{(k)} P \cdot F(\overline{z}^{(k)}), \\ u^{(k+1)} &\leftarrow u^{(k)} + F(\overline{z}^{(k)}). \end{aligned}$$

After some iterations, $\overline{z}^{(k)}$ becomes too far from $z^{(k)}$ and hence we need to correct the error by setting $z^{(k)} = P \cdot u^{(k)}$, which zeros the error.

Note that the algorithm explicitly maintains the approximate vector $\overline{z}$ while implicitly maintaining the exact vector z by $Pu^{(k)}$. This is different from the classical way to sketch Newton method [71], which is to simply run $z^{(k+1)} \leftarrow z^{(k)} + R^\top RP \cdot F(z^{(k)})$ or use another way to subsample and approximate P. Such a scheme relies on the iteration method to fix the error accumulated in the sketch, while we are actively fixing the error by having both the approximate explicit vector $\overline{z}$ and the exact implicit vector z.

Without precomputation, the cost of computing $R^{(k)}P$ is in fact higher than that of $P \cdot F(z^{(k)})$. The first one involves multiplying multiple vectors with P and the second one involves multiplying

1 vector with P. However, we can precompute $[R^{(1)\top}; R^{(2)\top}; \cdots ; R^{(T)\top}]^\top \cdot P$ by fast matrix multiplication. This decreases the cost of multiplying 1 vector with P to $n^{\omega-1}$ per vector. This is a huge saving from n^2. In our algorithm, we end up using only $\widetilde{O}(n)$ random vectors in total and hence the total cost is still roughly n^ω.

Maintaining the Sketch

The matrix P we use in interior point methods is of the form

$$P = \sqrt{W} A^\top (AWA^\top)^{-1} A \sqrt{W}$$

where W is some block diagonal matrix. [18] showed one can approximately maintain the matrix P with total cost $\widetilde{O}(n^\omega)$ across all iterations of interior point method. However, the cost of applying the dense matrix P with a vector z is roughly $O(n\|z\|_0)$ which is $O(n^2)$ for dense vectors. Since interior point methods takes at least $\sqrt{n}$ iterations in general, this gives a total runtime of $O(n^{2.5})$. The key idea in [18] is that one can design a stochastic interior point method such that each step only need to multiply P with a vector of density $\widetilde{O}(\sqrt{n})$. This bypasses the $n^{2.5}$ bottleneck.

In this paper, we do not have this issue because we only need to compute RPz which is much cheaper than Pz. We summarize why it suffices to maintain RP throughout the algorithm. In general, for interior point method, the vector z is roughly an unit vector and since P is an orthogonal projection, we have $\|Pz\|_2 = O(1)$. One simple insight we have is that if we multiply a random $\sqrt{n} \times n$ matrix R with values $\pm\frac{1}{\sqrt{n}}$ by Pz, we have $\|RPz\|_\infty = \widetilde{O}(\frac{1}{\sqrt{n}})$ (Lemma 3.2.5). Since there are $\widetilde{O}(\sqrt{n})$ iterations in interior point method, the total error is roughly $\widetilde{O}(1)$ in a correctly reweighed ℓ_∞ norm. In Section 2.5, we showed that this is exactly what interior point method needs for convergence. Furthermore, we note that though each step needs to use a fresh random matrix R_l of size $\sqrt{n} \times n$, the random matrices $[R_1^\top; R_2^\top; \cdots ; R_T^\top]^\top$ we need can all fit into $\widetilde{O}(n) \times n$ budget. Therefore, throughout the algorithm, we simply need to maintain the matrix $[R_1^\top; R_2^\top; \cdots ; R_T^\top]^\top P$ which can be done with total cost $\widetilde{O}(n^\omega)$ across all iterations using idea similar to [18].

The only reason the data structure looks complicated is that when the block matrix W changes in different location in $\sqrt{W} A^\top (AWA^\top)^{-1} A \sqrt{W}$, we need to update the matrix $[R_1; R_2; \cdots ; R_T]P$ appropriately. This gives us few simple cases to handle in the algorithm and in the proof. For the intuition on how to maintain P under W change, see [18, Section 2.2 and 5.1].

Fast rectangular matrix multiplication

Given two size $n \times n$ matrices, the time of multiplying them is $n^{2.81} < n^3$ by applying Strassen's original algorithm [83]. The current best running time takes n^ω time where $\omega < 2.373$ [90, 46]. One natural extension of multiplying two square matrices is multiplying two rectangular matrices. What is the running time of multiplying one $n \times n^a$ matrix with another $n^a \times n$ matrix? Let α denote the largest upper bound of a such that multiplying two rectangular matrices takes $n^{2+o(1)}$ time. The α is called the dual exponent of matrix multiplication, and the state-of-the-art result is $\alpha = 0.31$ [47]. We use the similar idea as [18] to delay the low-rank update when the rank is small so that fast maintenance is possible.

2.3 Preliminaries

Given a vector $x \in \mathbb{R}^n$ and m compact convex sets $K_1 \subset \mathbb{R}^{n_1}, K_2 \subset \mathbb{R}^{n_2}, \cdots, K_m \subset \mathbb{R}^{n_m}$ with $\sum_{i=1}^m n_i = n$. We use x_i to denote the i-th block of x, then $x \in \prod_{i=1}^m K_i$ if $x_i \in K_i$, $\forall i \in [m]$.

We say a block diagonal matrix $A \in \oplus_{i=1}^m \mathbb{R}^{n_i \times n_i}$ if A can be written as

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_m \end{bmatrix}$$

where $A_1 \in \mathbb{R}^{n_1 \times n_1}$, $A_2 \in \mathbb{R}^{n_2 \times n_2}$, and $A_m \in \mathbb{R}^{n_m \times n_m}$. For a matrix A, we use $\|A\|_F$ to denote its Frobenius norm and use $\|A\|$ to denote its operator norm. There are some trivial facts $\|AB\|_2 \leq \|A\|_2 \cdot \|B\|_2$ and $\|AB\|_F \leq \|A\|_F \cdot \|B\|_2$.

For notation convenience, we assume the number of variables $n \geq 10$ and there are no redundant constraints. In particular, this implies that the constraint matrix A is full rank.

For a positive integer n, let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for some absolute constant C. For any function f, we use $\mathrm{dom} f$ to denote the domain of function f.

For a vector v, We denote $\|v\|$ as the standard Euclidean norm of v and for a symmetric PSD matrix A, we let $\|v\|_A = (v^\top A v)^{1/2}$. For a convex function $f(x)$ that is clear from context, we denote $\|v\|_x = \|v\|_{\nabla^2 f(x)}$ and $\|v\|_x^* = \|v\|_{\nabla^2 f(x)^{-1}}$.

2.4 Robust Central Path

In this section we show how to move move efficiently from $x(1)$ to $x(\epsilon)$ for some tiny ϵ by staying on a robust version of the central path. Because we are maintaining values that are slightly off-center, we show that our analysis still goes through despite ℓ_∞ perturbations on the order of $O(1/\text{poly}\log(n))$.

Newton Step

To follow the path $x(t)$, we consider the optimality condition of (2.2):

$$\begin{aligned} s/t + \nabla\phi(x) &= 0, \\ Ax &= b, \\ A^\top y + s &= c \end{aligned}$$

where $\nabla\phi(x) = (\nabla\phi_1(x_1), \nabla\phi_2(x_2), \cdots, \nabla\phi_m(x_m))$. To handle the error incurred in the progress, we consider the perturbed central path

$$\begin{aligned} s/t + \nabla\phi(x) &= \mu, \\ Ax &= b, \\ A^\top y + s &= c \end{aligned}$$

where μ represent the error between the original central path and our central path. Each iteration, we decrease t by a certain factor. It may increase the error term μ. Therefore, we need a step to decrease the norm of μ. The Newton method to move μ to $\mu + h$ is given by

$$\begin{aligned} \frac{1}{t} \cdot \delta_s^{(\text{ideal})} + \nabla^2\phi(x) \cdot \delta_x^{(\text{ideal})} &= h, \\ A\delta_x^{(\text{ideal})} &= 0, \\ A^\top \delta_y^{(\text{ideal})} + \delta_s^{(\text{ideal})} &= 0 \end{aligned}$$

where $\nabla^2\phi(x)$ is a block diagonal matrix with the i-th block is given by $\nabla^2\phi_i(x_i)$. Letting $W = (\nabla^2\phi(x))^{-1}$, we can solve this:

$$\begin{aligned} \delta_y^{(\text{ideal})} &= -t \cdot \left(AWA^\top\right)^{-1} AWh, \\ \delta_s^{(\text{ideal})} &= t \cdot A^\top \left(AWA^\top\right)^{-1} AWh, \\ \delta_x^{(\text{ideal})} &= Wh - WA^\top \left(AWA^\top\right)^{-1} AWh. \end{aligned}$$

We define projection matrix $P \in \mathbb{R}^{n\times n}$ as follows

$$P = W^{1/2} A^\top \left(AWA^\top\right)^{-1} AW^{1/2}$$

and then we rewrite them

$$\delta_x^{(\text{ideal})} = W^{1/2}(I-P)W^{1/2}\delta_\mu, \tag{2.3}$$
$$\delta_s^{(\text{ideal})} = tW^{-1/2}PW^{1/2}\delta_\mu. \tag{2.4}$$

One standard way to analyze the central path is to measure the error by $\|\mu\|_{\nabla^2\phi(x)^{-1}}$ and uses the step induced by $h=-\mu$. One can easily prove that if $\|\mu\|_{\nabla^2\phi(x)^{-1}} < \frac{1}{10}$, one step of Newton step decreases the norm by a constant factor. Therefore, one can alternatively decrease t and do a Newton step to follow the path.

Robust Central Path Method

In this section, we develop a central path method that is robust under certain ℓ_∞ perturbations. Due to the ℓ_∞ perturbation, we measure the error μ by a soft max instead of the ℓ_2 type potential:

Definition 2.4.1. *For each $i\in[m]$, let $\mu_i^t(x,s)\in\mathbb{R}^{n_i}$ and $\gamma_i^t(x,s)\in\mathbb{R}$ be defined as follows:*

$$\mu_i^t(x,s) = s_i/t + \nabla\phi_i(x_i), \tag{2.5}$$
$$\gamma_i^t(x,s) = \|\mu_i^t(x,s)\|_{\nabla^2\phi_i(x_i)^{-1}}, \tag{2.6}$$

and we define potential function Φ as follows:

$$\Phi^t(x,s) = \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s))$$

where $\lambda = O(\log m)$.

The *robust central path* is the region (x,s) that satisfies $\Phi^t(x,s)\le O(m)$. To run our convergence argument, we will be setting λ appropriately so that staying on the robust central path will guarantee a ℓ_∞ bound on γ. Then, we will show how to maintain $\Phi^t(x,s)$ to be small throughout the algorithm while decreasing t, always staying on the robust central path. This is broken into a two step analysis: the progress step (decreasing t) and the centering step (moving x,s to decrease γ).

It is important to note that to follow the robust central path, we no longer pick the standard Newton direction by setting $h=-\mu$. To explain how we pick our centering step, suppose we can move $\mu\to\mu+h$ arbitrarily with the only restriction on the distance $\|h\|_{\nabla^2\phi(x)^{-1}}=\alpha$. Then, the natural step would be

$$h = \arg\min_{\|h\|_{\nabla^2\phi(x)^{-1}}=\alpha} \langle\nabla f(\mu(x,s)), h\rangle$$

where $f(\mu)=\sum_{i=1}^m \exp(\lambda\|\mu\|_{\nabla^2\phi_i(x_i)^{-1}})$. Note that

$$\nabla f(\mu^t(x,s))_i = \lambda\exp(\lambda\gamma_i^t(x,s))/\gamma_i^t(x,s)\cdot\nabla^2\phi_i(x_i)^{-1}\mu_i^t(x,s).$$

Therefore, the solution for the minimization problem is

$$h_i^{(\mathrm{ideal})} = -\alpha \cdot c_i^t(x,s)^{(\mathrm{ideal})} \mu_i^t(x,s) \in \mathbb{R}^{n_i},$$

where $\mu_i^t(x,s) \in \mathbb{R}^{n_i}$ is defined as Eq. (2.5) and $c_i^t(x,s) \in \mathbb{R}$ is defined as

$$c_i^t(x,s)^{(\mathrm{ideal})} = \frac{\exp(\lambda \gamma_i^t(x,s))/\gamma_i^t(x,s)}{(\sum_{i=1}^m \exp(2\lambda \gamma_i^t(x,s)))^{1/2}}.$$

Eq. (2.3) and Eq. (2.4) gives the corresponding ideal step on x and s.

Now, we discuss the perturbed version of this algorithm. Instead of using the exact x and s in the formula of h, we use a $\overline{x}$ which is approximately close to x and a $\overline{s}$ which is close to s. Precisely, we have

$$h_i = -\alpha \cdot c_i^t(\overline{x},\overline{s}) \mu_i^t(\overline{x},\overline{s}) \tag{2.7}$$

where

$$c_i^t(x,s) = \begin{cases} \frac{\exp(\lambda \gamma_i^t(x,s))/\gamma_i^t(x,s)}{(\sum_{i=1}^m \exp(2\lambda \gamma_i^t(x,s)))^{1/2}} & \text{if } \gamma_i^t(x,s) \geq 96\sqrt{\alpha} \\ 0 & \text{otherwise} \end{cases}. \tag{2.8}$$

Note that our definition of c_i^t ensures that $c_i^t(x,s) \leq \frac{1}{96\sqrt{\alpha}}$ regardless of the value of $\gamma_i^t(x,s)$. This makes sure we do not move too much in any coordinates and indeed when γ_i^t is small, it is fine to set $c_i^t = 0$. Furthermore, for the formula on δ_x and δ_s, we use some matrix $\widetilde{V}$ that is close to $(\nabla^2 \phi(x))^{-1}$. Precisely, we have

$$\delta_x = \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2} h, \tag{2.9}$$

$$\delta_s = t \cdot \widetilde{V}^{-1/2} \widetilde{P}\, \widetilde{V}^{1/2} h. \tag{2.10}$$

where

$$\widetilde{P} = \widetilde{V}^{1/2} A^\top (A \widetilde{V} A^\top)^{-1} A \widetilde{V}^{1/2}.$$

Here we give a quick summary of our algorithm. (The more detailed of our algorithm can be found in Algorithm 5 and 6 in Section 3.3.)

```
• ROBUSTIPM($A, b, c, \phi, \delta$)
    – $\lambda = 2^{16}\log(m)$, $\alpha = 2^{-20}\lambda^{-2}$, $\kappa = 2^{-10}\alpha$.
    – $\delta = \min(\frac{1}{\lambda}, \delta)$.
    – $\nu = \sum_{i=1}^{m} \nu_i$ where $\nu_i$ are the self-concordant parameters of $\phi_i$.
    – Modify the convex problem and obtain an initial $x$ and $s$ according to Lemma 3.4.2.
    – $t = 1$.
    – While $t > \frac{\delta^2}{4\nu}$
        ∗ Find $\overline{x}$ and $\overline{s}$ such that $\|\overline{x}_i - x_i\|_{\overline{x}_i} < \alpha$ and $\|\overline{s}_i - s_i\|_{\overline{x}_i}^{*} < t\alpha$ for all $i$.
        ∗ Find $\widetilde{V}_i$ such that $(1-\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1} \preceq \widetilde{V}_i \preceq (1+\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1}$ for all $i$.
        ∗ Compute $h = -\alpha \cdot c_i^t(\overline{x},\overline{s})\mu_i^t(\overline{x},\overline{s})$ where
              $$c_i^t(\overline{x},\overline{s}) = \begin{cases} \frac{\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))/\gamma_i^t(\overline{x},\overline{s})}{(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}} & \text{if } \gamma_i^t(\overline{x},\overline{s}) \geq 96\sqrt{\alpha} \\ 0 & \text{otherwise} \end{cases}.$$
          and $\mu_i^t(\overline{x},\overline{s}) = \overline{s}_i/t + \nabla\phi_i(\overline{x}_i)$ and $\gamma_i^t(\overline{x},\overline{s}) = \|\mu_i^t(\overline{x},\overline{s})\|_{\nabla^2\phi_i(\overline{x}_i)^{-1}}$
        ∗ Let $\widetilde{P} = \widetilde{V}^{1/2}A^\top(A\widetilde{V}A^\top)^{-1}A\widetilde{V}^{1/2}$.
        ∗ Compute $\delta_x = \widetilde{V}^{1/2}(I-\widetilde{P})\widetilde{V}^{1/2}h$ and $\delta_s = t\cdot\widetilde{V}^{-1/2}\widetilde{P}\,\widetilde{V}^{1/2}h$.
        ∗ Move $x \leftarrow x + \delta_x$, $s \leftarrow s + \delta_s$.
        ∗ $t^{\text{new}} = (1 - \frac{\kappa}{\sqrt{\nu}})t$.
    – Return an approximation solution of the convex problem according to Lemma 3.4.2.
```

Theorem 2.4.2 (Robust Interior Point Method)**.** *Consider a convex problem* $\min_{Ax=b, x\in\prod_{i=1}^{m}K_i} c^\top x$ *where* K_i *are compact convex sets. For each* i*, we are given a* ν_i*-self concordant barrier function* ϕ_i *for* K_i*. Also, we are given* $x^{(0)} = \arg\min_x \sum_{i=1}^{m}\phi_i(x_i)$*. Assume that*

1. *Diameter of the set: For any* $x \in \prod K_i$*, we have that* $\|x\|_2 \leq R$*.*
2. *Lipschitz constant of the program:* $\|c\|_2 \leq L$*.*

Then, the algorithm ROBUSTIPM *finds a vector* x *such that*

$$c^\top \overline{x}_{1:n} \leq \min_{Ax=b, x\in\prod_i K_i} c^\top x + LR\cdot\delta,$$

$$\|Ax-b\|_1 \leq 3\delta \cdot \left(R\sum_{i,j}|A_{i,j}| + \|b\|_1\right),$$

$$x \in \prod_i K_i.$$

in $O(\sqrt{\nu}\log^2 m \log(\frac{\nu}{\delta}))$ *iterations.*

Proof. Lemma 3.4.2 shows that the initial x and s satisfies

$$\|s + \nabla\phi(x)\|_x^* \leq \delta \leq \frac{1}{\lambda}$$

where the last inequality is due to our step $\delta \leftarrow \min(\frac{1}{\lambda}, \delta)$. This implies that $\gamma_i^1(x,s) = \|s_i + \nabla\phi_i(x_i)\|_{x_i}^* \leq \frac{1}{\lambda}$ and hence $\Phi^1(x,s) \leq e \cdot m \leq 80\frac{m}{\alpha}$ for the initial x and s. Apply Lemma 2.5.8 repetitively, we have that $\Phi^t(x,s) \leq 80\frac{m}{\alpha}$ during the whole algorithm. In particular, we have this at the end of the algorithm. This implies that

$$\|s_i + \nabla\phi_i(x_i)\|_{x_i}^* \leq \frac{\log(80\frac{m}{\alpha})}{\lambda} \leq 1$$

at the end. Therefore, we can apply Lemma 3.4.3 to show that

$$\langle c, x\rangle \leq \langle c, x^*\rangle + 4t\nu \leq \langle c, x^*\rangle + \delta^2$$

where we used the stop condition for t at the end. Note that this guarantee holds for the modified convex program. Since the error is δ^2, Lemma 3.4.2 shows how to get an approximate solution for the original convex program with error $LR \cdot \delta$.

The number of steps follows from the fact we decrease t by $1 - \frac{1}{\sqrt{\nu}\log^2 m}$ factor every iteration. $\square$

Statement	Parameters
Lemma 2.5.2	$\mu_i^t(x,s) \to \mu_i^t(x^{\text{new}}, s^{\text{new}})$
Lemma 2.5.4	$\gamma_i^t(x,x,s) \to \gamma_i^t(x^{\text{new}}, x, s^{\text{new}})$
Lemma 2.5.5	$\Phi(x,x,s) \to \Phi(x^{\text{new}}, x, s^{\text{new}})$
Lemma 2.5.6	$\Phi(x^{\text{new}}, x, s^{\text{new}}) \to \Phi(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})$
Lemma 2.5.7	$\Phi^t \to \Phi^{t^{\text{new}}}$
Lemma 2.5.8	$\Phi^t(x,s) \to \Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}})$

Table 2.1: Bounding the changes of different variables

2.5 Analysis of Robust Central Path

Basically, the main proof is just a simple calculation on how $\Phi^t(x,s)$ changes during 1 iteration. It could be compared to the proof of ℓ_∞ potential reduction arguments for the convergence of long-step interior point methods, although the main difficulty arises from the perturbations from stepping using $\overline{x}, \overline{s}$ instead of x, s.

To organize the calculations, we note that the term $\gamma_i^t(x,s) = \|\mu_i^t(x,s)\|_{\nabla^2\phi_i(x_i)^{-1}}$ has two terms involving x, one in the μ term and one in the Hessian. Hence, we separate how different x affect the potential by defining

$$\begin{aligned} \gamma_i^t(x,z,s) &= \|\mu_i^t(x,s)\|_{\nabla^2\phi_i(z_i)^{-1}}, \\ \Phi^t(x,z,s) &= \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,z,s)). \end{aligned}$$

One difference between our proof and standard ℓ_2 proofs of interior point is that we assume the barrier function is decomposable. We define $\alpha_i = \|\delta_{x,i}\|_{\overline{x}_i}$ is the "step" size of the coordinate i. One crucial fact we are using is that sum of squares of the step sizes is small.

Lemma 2.5.1. *For all $i \in [m]$, let $\alpha_i = \|\delta_{x,i}\|_{\overline{x}_i}$. Then,*

$$\sum_{i=1}^m \alpha_i^2 \le 4\alpha^2.$$

Proof. Note that

$$\sum_{i=1}^m \alpha_i^2 = \|\delta_x\|_{\overline{x}}^2 = h^\top \widetilde{V}^{1/2}(I-\widetilde{P})\widetilde{V}^{1/2}\nabla^2\phi(\overline{x})\widetilde{V}^{1/2}(I-\widetilde{P})\widetilde{V}^{1/2}h.$$

Since $(1-\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1} \preceq \widetilde{V}_i \preceq (1+\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1}$, we have that

$$(1-\alpha)(\nabla^2\phi(\overline{x}))^{-1} \preceq \widetilde{V} \preceq (1+\alpha)(\nabla^2\phi(\overline{x}))^{-1}.$$

Using $\alpha \leq \frac{1}{10000}$, we have that

$$\sum_{i=1}^{m} \alpha_i^2 \leq 2h^\top \widetilde{V}^{1/2}(I - \widetilde{P})(I - \widetilde{P})\widetilde{V}^{1/2}h \leq 2h^\top \widetilde{V}h$$

where we used that $I - \widetilde{P}$ is an orthogonal projection at the end. Finally, we note that

$$\begin{aligned} h^\top \widetilde{V} h &\leq 2 \sum_{i=1}^{m} \|h_i\|_{\overline{x}_i}^{*2} \\ &= 2\alpha^2 \sum_{i=1}^{m} c_i^t(\overline{x}, \overline{s})^2 \|\mu_i^t(\overline{x}, \overline{s})\|_{\overline{x}_i}^{*2} \\ &\leq 2\alpha^2 \sum_{i=1}^{m} \frac{\exp(2\lambda\gamma_i^t(\overline{x}, \overline{s}))/\gamma_i^t(\overline{x}, \overline{s})^2}{\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(\overline{x}, \overline{s}))^{1/2}} \|\mu_i^t(\overline{x}, \overline{s})\|_{\overline{x}_i}^{*2} \\ &= 2\alpha^2 \frac{\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(\overline{x}, \overline{s}))}{\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(\overline{x}, \overline{s}))} \\ &= 2\alpha^2 \end{aligned}$$

where the second step follows from definition of h_i (2.7), the third step follows from definition c_i^t (2.8), the fourth step follows from definition of γ_i^t (2.6).

Therefore, putting it all together, we can show

$$\sum_{i=1}^{m} \alpha_i^2 \leq 4\alpha^2.$$

□

Changes in μ and γ

We provide basic lemmas that bound changes in μ, γ due to the centering steps.

Lemma 2.5.2 (Changes in μ)**.** *For all $i \in [m]$, let*

$$\mu_i^t(x^{\text{new}}, s^{\text{new}}) = \mu_i^t(x, s) + h_i + \epsilon_i^{(\mu)}.$$

Then, $\|\epsilon_i^{(\mu)}\|_{x_i}^* \leq 10\alpha \cdot \alpha_i$.

Proof. Let $x^{(u)} = ux^{\text{new}} + (1-u)x$ and $\mu_i^{\text{new}} = \mu_i^t(x^{\text{new}}, s^{\text{new}})$. The definition of μ (2.5) shows that

$$\begin{aligned} \mu_i^{\text{new}} &= \mu_i + \frac{1}{t}\delta_{s,i} + \nabla\phi_i(x_i^{\text{new}}) - \nabla\phi_i(x_i) \\ &= \mu_i + \frac{1}{t}\delta_{s,i} + \int_0^1 \nabla^2\phi_i(x_i^{(u)})\delta_{x,i}\,\mathrm{d}u \\ &= \mu_i + \frac{1}{t}\delta_{s,i} + \nabla^2\phi_i(\overline{x}_i)\delta_{x,i} + \int_0^1 \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\,\mathrm{d}u. \end{aligned}$$

By the definition of δ_x and δ_s (2.9) and (2.10), we have that $\frac{1}{t}\delta_{s,i} + \widetilde{V}_i^{-1}\delta_{x,i} = h_i$. Hence, we have

$$\mu_i^{\text{new}} = \mu_i + h_i + \epsilon_i^{(\mu)}$$

where

$$\epsilon_i^{(\mu)} = \int_0^1 \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\,\mathrm{d}u + (\nabla^2\phi_i(\overline{x}_i) - \widetilde{V}_i^{-1})\delta_{x,i}. \tag{2.11}$$

To bound $\epsilon_i^{(\mu)}$, we note that

$$\|x_i^{(t)} - \overline{x}_i\|_{\overline{x}_i} \leq \|x_i^{(t)} - x_i\|_{\overline{x}_i} + \|x_i - \overline{x}_i\|_{\overline{x}_i} \leq \|\delta_{x,i}\|_{\overline{x}_i} + \alpha \leq 3\alpha$$

where we used Lemma 2.5.1. Using $\alpha \leq \frac{1}{100}$, Theorem 1.2.3 shows that

$$-7\alpha \cdot \nabla^2\phi_i(\overline{x}_i) \preceq \nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i) \preceq 7\alpha \cdot \nabla^2\phi_i(\overline{x}_i).$$

Equivalently, we have

$$(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)) \cdot (\nabla^2\phi_i(\overline{x}_i))^{-1} \cdot (\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)) \preceq (7\alpha)^2 \cdot \nabla^2\phi_i(\overline{x}_i).$$

Using this, we have

$$\begin{aligned}\left\|\int_0^1 \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\,\mathrm{d}u\right\|_{\overline{x}_i}^* &\leq \int_0^1 \left\|\left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\right\|_{\overline{x}_i}^* \mathrm{d}u \\ &\leq 7\alpha\|\delta_{x,i}\|_{\overline{x}_i} = 7\alpha \cdot \alpha_i. \end{aligned} \tag{2.12}$$

For the other term in $\epsilon_i^{(\mu)}$, we note that

$$(1-2\alpha) \cdot (\nabla^2\phi_i(\overline{x}_i)) \preceq \widetilde{V}_i^{-1} \preceq (1+2\alpha) \cdot (\nabla^2\phi_i(\overline{x}_i)).$$

Hence, we have

$$\left\|(\nabla^2\phi_i(\overline{x}_i) - \widetilde{V}_i^{-1})\delta_{x,i}\right\|_{\overline{x}_i}^* \leq 2\alpha\|\delta_{x,i}\|_{\overline{x}_i} = 2\alpha \cdot \alpha_i. \tag{2.13}$$

Combining (2.11), (2.12) and (2.13), we have

$$\|\epsilon_i^{(\mu)}\|_{\overline{x}_i}^* \leq 9\alpha \cdot \alpha_i.$$

Finally, we use the fact that x_i and $\overline{x}_i$ are α close and hence again by self-concordance, $\|\epsilon_i^{(\mu)}\|_{x_i}^* \leq 10\alpha \cdot \alpha_i$.

□

Before bounding the change of γ, we first prove a helper lemma:

Lemma 2.5.3. *For all $i \in [m]$, we have*

$$\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^* \leq 4\alpha.$$

Proof. Note that

$$\|\mu_i^t(x,s)-\mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^* = \frac{1}{t}\|s_i-\overline{s}_i\|_{\overline{x}_i}^* + \|\nabla\phi_i(x_i)-\nabla\phi_i(\overline{x}_i)\|_{\overline{x}_i}^*.$$

For the first term, we have $\|s_i-\overline{s}_i\|_{\overline{x}_i}^* \leq t\alpha$.

For the second term, let $x_i^{(u)} = ux_i + (1-u)\overline{x}_i$. Since x_i is close enough to $\overline{x}_i$, Theorem 1.2.3 shows that $\nabla^2\phi_i(x_i^{(u)}) \preceq 2\cdot\nabla^2\phi_i(\overline{x}_i)$. Hence, we have

$$\|\nabla\phi_i(x_i)-\nabla\phi_i(\overline{x}_i)\|_{\overline{x}_i}^* = \left\|\int_0^1 \nabla^2\phi_i(x_i^{(u)})\cdot(x_i-\overline{x}_i)du\right\|_{\overline{x}_i}^* \leq 2\|x_i-\overline{x}_i\|_{\overline{x}_i} = 2\alpha.$$

Hence, we have $\|\mu_i^t(x,s)-\mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^* \leq 3\alpha$ and using again x_i is close enough to $\overline{x}_i$ to get the final result.

□

Lemma 2.5.4 (Changes in γ). *For all $i\in[m]$, let*

$$\gamma_i^t(x^{\text{new}},x,s^{\text{new}}) \leq (1-\alpha\cdot c_i^t(\overline{x},\overline{s}))\gamma_i^t(x,x,s)+\epsilon_i^{(\gamma)}.$$

then $\epsilon_i^{(\gamma)} \leq 10\alpha\cdot(\alpha c_i^t(\overline{x},\overline{s})+\alpha_i)$. Furthermore, we have $|\gamma_i^t(x^{\text{new}},x,s^{\text{new}})-\gamma_i^t(x,x,s)| \leq 3\alpha$.

Proof. For the first claim, Lemma 2.5.2, the definition of γ (2.6), h (2.7) and c (2.8) shows that

$$\begin{aligned}\gamma_i^t(x^{\text{new}},x,s^{\text{new}}) &= \|\mu_i^t(x,s)+h_i+\epsilon_i^{(\mu)}\|_{x_i}^* \\ &= \|(1-\alpha\cdot c_i^t(\overline{x},\overline{s}))\mu_i^t(x,s)+\epsilon_i\|_{x_i}^*\end{aligned}$$

where $\epsilon_i = \alpha\cdot c_i^t(\overline{x},\overline{s})(\mu_i^t(x,s)-\mu_i^t(\overline{x},\overline{s}))+\epsilon_i^{(\mu)}$.

From the definition of c_i^t, we have that $c_i^t \leq \frac{1}{96\sqrt{\alpha}} \leq \frac{1}{\alpha}$ and hence $0 \leq 1-\alpha\cdot c_i^t(\overline{x},\overline{s}) \leq 1$. Therefore, we have

$$\gamma_i^t(x^{\text{new}},x,s^{\text{new}}) \leq (1-\alpha\cdot c_i^t(\overline{x},\overline{s}))\gamma_i^t(x,x,s)+\|\epsilon_i\|_{x_i}^*. \tag{2.14}$$

Now, we bound $\|\epsilon_i\|_{x_i}^*$:

$$\begin{aligned}\|\epsilon_i\|_{x_i}^* &\leq \alpha c_i^t(\overline{x},\overline{s})\cdot\|\mu_i^t(x,s)-\mu_i^t(\overline{x},\overline{s})\|_{x_i}^* + \|\epsilon_i^{(\mu)}\|_{x_i}^* \\ &\leq 4\alpha^2 c_i^t(\overline{x},\overline{s})+10\alpha\cdot\alpha_i \qquad (2.15)\end{aligned}$$

where we used Lemma 2.5.3 and Lemma 2.5.2 at the end.

For the second claim, we have

$$\left|\gamma_i^t(x^{\text{new}},x,s^{\text{new}})-\gamma_i^t(x,x,s)\right| \leq \|h_i+\epsilon_i^{(\mu)}\|_{x_i}^* \leq 2\alpha+10\alpha\cdot\alpha_i$$

where we used (2.15) and that $\|h_i\|_{x_i}^* \leq 2\|h\|_{\overline{x}}^* \leq 2\alpha$. From Lemma 2.5.1 and that $\alpha \leq \frac{1}{10000}$, we have $10\alpha\cdot\alpha_i \leq 20\alpha^2 \leq \alpha$.

□

Movement from (x, x, s) to $(x^{\text{new}}, x, s^{\text{new}})$

In the previous section, we see that γ_i will be expected to decrease by a factor of $\alpha \cdot c_i^t$ up to some small perturbations. We show that our potential Φ^t will therefore decrease significantly.

Lemma 2.5.5 (Movement along the first and third parameters)**.** *Assume that* $\gamma_i^t(x, x, s) \leq 1$ *for all* i*. We have*

$$\Phi^t(x^{\text{new}}, x, s^{\text{new}}) \leq \Phi^t(x, x, s) - \frac{\alpha\lambda}{5}\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x}, \overline{s}))\right)^{1/2} + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}).$$

Proof. Let $\Phi^{\text{new}} = \Phi^t(x^{\text{new}}, x, s^{\text{new}})$, $\Phi = \Phi^t(x, x, s)$,

$$\gamma^{(u)} = u\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) + (1-u)\gamma_i^t(x, x, s).$$

Then, we have that

$$\Phi^{\text{new}} - \Phi = \sum_{i=1}^m (e^{\lambda\gamma_i^{(1)}} - e^{\lambda\gamma_i^{(0)}}) = \lambda \sum_{i=1}^m e^{\lambda\gamma_i^{(\zeta)}}(\gamma_i^{(1)} - \gamma_i^{(0)})$$

for some $0 \leq \zeta \leq 1$. Let $v_i = \gamma_i^{(1)} - \gamma_i^{(0)}$. Lemma 2.5.4 shows that

$$v_i \leq -\alpha \cdot c_i^t(\overline{x}, \overline{s}) \cdot \gamma_i^t(x, x, s) + \epsilon_i^{(\gamma)} = -\alpha \cdot c_i^t(\overline{x}, \overline{s}) \cdot \gamma_i^{(0)} + \epsilon_i^{(\gamma)}$$

and hence

$$\frac{\Phi^{\text{new}} - \Phi}{\lambda} \leq -\alpha \sum_{i=1}^m c_i^t(\overline{x}, \overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda\gamma_i^{(\zeta)}) + \sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^{(\zeta)}). \tag{2.16}$$

To bound the first term in (2.16), we first relate $\gamma_i^{(0)}$, $\gamma_i^{(\zeta)}$ and $\gamma_i^t(\overline{x}, \overline{s})$. Lemma 2.5.4 shows that

$$|\gamma_i^{(0)} - \gamma_i^{(\zeta)}| \leq |\gamma_i^{(0)} - \gamma_i^{(1)}| \leq 3\alpha. \tag{2.17}$$

Lemma 2.5.3 with standard self-concordance argument shows that

$$\begin{aligned}\left|\gamma_i^t(\overline{x}, \overline{s}) - \gamma_i^{(0)}\right| &\leq \left|\gamma_i^t(\overline{x}, \overline{x}, \overline{s}) - \gamma_i^t(x, \overline{x}, s)\right| + \left|\gamma_i^t(x, \overline{x}, s) - \gamma_i^t(x, x, s)\right| \\ &\leq \|\mu_i^t(x, s) - \mu_i^t(\overline{x}, \overline{s})\|_{\overline{x}_i}^* + \left|\|\mu_i^t(x, s)\|_{\overline{x}_i}^* - \|\mu_i^t(x, s)\|_{x_i}^*\right| \\ &\leq 2\|\mu_i^t(x, s) - \mu_i^t(\overline{x}, \overline{s})\|_{x_i}^* + 2\alpha\|\mu_i^t(x, s)\|_{x_i}^* \\ &\leq 8\alpha + 2\alpha \leq 10\alpha \end{aligned} \tag{2.18}$$

where we used that $\|\mu_i^t(x, s)\|_{x_i}^* = \gamma_i^t(x, x, s) \leq 1$ for all i.

Using (2.17) and (2.18), we have

$$
\begin{aligned}
&\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda \gamma_i^{(\zeta)}) \\
=& \sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda \gamma_i^t(\overline{x},\overline{s}) - \lambda \gamma_i^t(\overline{x},\overline{s}) + \lambda \gamma_i^{(0)} - \lambda \gamma_i^{(0)} + \lambda \gamma_i^{(\zeta)}) \\
\geq& \sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda \gamma_i^t(\overline{x},\overline{s}) - 13\lambda\alpha) \\
\geq& \frac{1}{2} \sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda \gamma_i^t(\overline{x},\overline{s})) \\
\geq& \frac{1}{2} \sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^t(\overline{x},\overline{s}) \exp(\lambda \gamma_i^t(\overline{x},\overline{s})) - 3\alpha \sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \exp(\lambda \gamma_i^t(\overline{x},\overline{s})). \qquad (2.19)
\end{aligned}
$$

where the third step follows from $\exp(-13\lambda\alpha) \geq 1/2$.

For the first term in (2.19), we have

$$
\begin{aligned}
&\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^t(\overline{x},\overline{s}) \exp(\lambda \gamma_i^t(\overline{x},\overline{s})) \\
=& \sum_{\gamma_i^t(\overline{x},\overline{s}) \geq 96\sqrt{\alpha}} \frac{\exp(2\lambda \cdot \gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})))^{1/2}} \\
=& \sum_{i=1}^{m} \frac{\exp(2\lambda \cdot \gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})))^{1/2}} - \sum_{\gamma_i^t(\overline{x},\overline{s}) < 96\sqrt{\alpha}} \frac{\exp(2\lambda \cdot \gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})))^{1/2}} \\
\geq& \left(\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})) \right)^{1/2} - \frac{m \cdot \exp(192\lambda \cdot \sqrt{\alpha})}{(\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})))^{1/2}}.
\end{aligned}
$$

So, if $\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})) \geq m \cdot \exp(192\lambda \cdot \sqrt{\alpha})$, we have

$$
\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^t(\overline{x},\overline{s}) \exp(\lambda \gamma_i^t(\overline{x},\overline{s})) \geq \left(\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})) \right)^{1/2} - \sqrt{m} \cdot \exp(192\lambda\sqrt{\alpha}).
$$

Note that if $\sum_{i=1}^{m} \exp(2\lambda \gamma_i^t(\overline{x},\overline{s})) \leq m \cdot \exp(192\lambda \cdot \sqrt{\alpha})$, this is still true because left hand side is lower bounded by 0. For the second term in (2.19), we have

$$
\begin{aligned}
\sum_{i=1}^m c_i^t(\overline{x},\overline{s}) \exp(\lambda \gamma_i^t(\overline{x},\overline{s})) &= \sum_{\gamma_i^t(\overline{x},\overline{s}) \geq 96\sqrt{\alpha}} \frac{\exp(\lambda \cdot \gamma_i^t(\overline{x},\overline{s}))/\gamma_i^t(\overline{x},\overline{s})}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}} \exp(\lambda\gamma_i^t(\overline{x},\overline{s})) \\
&\leq \frac{1}{96\sqrt{\alpha}} \sum_{\gamma_i^t(\overline{x},\overline{s}) \geq 96\sqrt{\alpha}} \frac{\exp(2\lambda \cdot \gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}} \\
&\leq \frac{1}{96\sqrt{\alpha}} \sum_{i=1}^m \frac{\exp(2\lambda \cdot \gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}} \\
&= \frac{1}{96\sqrt{\alpha}} \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})) \right)^{1/2}.
\end{aligned}
$$

where the second step follows $\frac{1}{\gamma_i^t(\overline{x},\overline{s})} \leq \frac{1}{96\sqrt{\alpha}}$, and the third step follows from each term in the summation is non-negative.

Combining the bounds for both first and second term in (2.19), we have

$$
\begin{aligned}
\sum_{i=1}^m c_i^t(\overline{x},\overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda\gamma_i^{(\zeta)}) &\geq \frac{1}{2} \left(\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})) \right)^{1/2} - \sqrt{m} \cdot \exp(192\lambda\sqrt{\alpha}) \right) \\
&\quad - \frac{3\alpha}{96\sqrt{\alpha}} \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})) \right)^{1/2} \\
&\geq \frac{2}{5} \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})) \right)^{1/2} - \sqrt{m} \cdot \exp(192\lambda\sqrt{\alpha}). \qquad (2.20)
\end{aligned}
$$

where the last step follows from $\frac{1}{2} - \frac{3\alpha}{96\sqrt{\alpha}} \geq \frac{1}{2} - \frac{3}{96} = \frac{45}{96} \geq \frac{2}{5}$.

For the second term in (2.16), we note that $|\gamma_i^{(\zeta)} - \gamma_i^t(\overline{x},\overline{s})| \leq 13\alpha \leq \frac{1}{2\lambda}$ by (2.17) and (2.18). Hence,

$$
\sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^{(\zeta)}) \leq 2 \sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^t(\overline{x},\overline{s})).
$$

Now, we use $\epsilon_i^{(\gamma)} \leq 10\alpha \cdot (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i)$ (Lemma 2.5.4) to get

$$
\begin{aligned}
\sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^{(\zeta)}) &\leq 20\alpha \sum_{i=1}^m (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i) \cdot \exp(\lambda\gamma_i^t(\overline{x},\overline{s})) \\
&\leq 20\alpha \left(\sum_{i=1}^m (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i)^2 \right)^{1/2} \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})) \right)^{1/2}.
\end{aligned}
$$

where the last step follows from Cauchy-Schwarz inequality.

Note that by using Cauchy-Schwarz,

$$\begin{aligned}\left(\sum_{i=1}^m (\alpha c_i^t(\overline{x},\overline{s})+\alpha_i)^2\right)^{1/2} &\leq \alpha\left(\sum_{i=1}^m c_i^t(\overline{x},\overline{s})^2\right)^{1/2}+\left(\sum_{i=1}^m \alpha_i^2\right)^{1/2}\\ &\leq \alpha\cdot\frac{1}{96\sqrt{\alpha}}+2\alpha\leq\frac{\sqrt{\alpha}}{90}.\end{aligned}$$

where we used the definition of c_i^t, Lemma 2.5.1 and $\alpha\leq\frac{1}{2^{24}}$. Together, we conclude

$$\sum_{i=1}^m \epsilon_i^{(\gamma)}\exp(\lambda\gamma_i^{(\zeta)})\leq\frac{1}{5}\alpha\left(\sum_{i=1}^m\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}. \tag{2.21}$$

Combining (2.20) and (2.21) to (2.16) gives

$$\begin{aligned}\frac{\Phi^{\text{new}}-\Phi}{\lambda}\leq &-\frac{2}{5}\alpha\left(\sum_{i=1}^m\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}+\sqrt{m}\cdot\exp(192\lambda\sqrt{\alpha})+\frac{1}{5}\alpha\left(\sum_{i=1}^m\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}\\ =&-\frac{1}{5}\alpha\left(\sum_{i=1}^m\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}+\sqrt{m}\cdot\exp(192\lambda\sqrt{\alpha}).\end{aligned}$$

where the last step follows from merging the first term with the third term. □

Movement from $(x^{\text{new}},x,s^{\text{new}})$ to $(x^{\text{new}},x^{\text{new}},s^{\text{new}})$

Next, we must analyze the potential change when we change the second term.

Lemma 2.5.6 (Movement along the second parameter)**.** *Assume that* $\|\gamma_i^t(x,s)\|_\infty\leq 1$. *Then we have*

$$\Phi^t(x^{\text{new}},x^{\text{new}},s^{\text{new}})\leq\Phi^t(x^{\text{new}},x,s^{\text{new}})+12\alpha(\|\gamma_i^t(x,s)\|_\infty+3\alpha)\lambda\left(\sum_{i=1}^m\exp(2\lambda\gamma_i^t(x,x,s))\right)^{1/2}.$$

Proof. We can upper bound $\Phi^t(x^{\text{new}},x^{\text{new}},s^{\text{new}})$ as follows

$$\begin{aligned}\Phi^t(x^{\text{new}},x^{\text{new}},s^{\text{new}}) &=\sum_{i=1}^m\exp(\lambda\gamma_i^t(x^{\text{new}},x^{\text{new}},s^{\text{new}}))\\ &\leq\sum_{i=1}^m\exp(\lambda\gamma_i^t(x^{\text{new}},x,s^{\text{new}})(1+2\alpha_i)).\end{aligned}$$

where the second step follows from $\gamma_i^t(x^{\text{new}},x^{\text{new}},s^{\text{new}})\leq\gamma_i^t(x^{\text{new}},x,s^{\text{new}})\cdot(1+2\alpha_i)$ by self-concordance (Theorem 1.2.3) and $\|x_i^{\text{new}}-x_i\|_{x_i}\leq 2\|x_i^{\text{new}}-x_i\|_{\overline{x}_i}\leq 2\alpha_i$.

Now, by Lemma 2.5.4, we note that $\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) \le \gamma_i^t(x, x, s) + 3\alpha \le 1 + 3\alpha$ and that $\alpha \le \frac{1}{100\lambda}$. Hence, by a simple taylor expansion, we have

$$\begin{aligned}&\Phi^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})\\ \le&\sum_{i=1}^m \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}})) + 3\sum_{i=1}^m \alpha_i \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}))\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}).\end{aligned}$$

Finally, we bound the last term by

$$\begin{aligned}&\sum_{i=1}^m \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}))\gamma_i^t(x^{\text{new}}, x, s^{\text{new}})\alpha_i\\ \le&\ \sum_{i=1}^m \exp(\lambda\gamma_i^t(x, s) + 3\lambda\alpha)(\gamma_i^t(x, s) + 3\alpha)\alpha_i\\ \le&\ 2(\|\gamma_i^t(x, s)\|_\infty + 3\alpha)\sum_{i=1}^m \exp(\lambda\gamma_i^t(x, s))\alpha_i\\ \le&\ 2(\|\gamma_i^t(x, s)\|_\infty + 3\alpha)\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x, s))\right)^{1/2}\left(\sum_i \alpha_i^2\right)^{1/2}\\ \le&\ 4\alpha(\|\gamma_i^t(x, s)\|_\infty + 3\alpha)\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x, x, s))\right)^{1/2},\end{aligned}$$

where the third step follows from Cauchy-Schwarz inequality, the last step follows from $\sum_{i=1}^m \alpha_i^2 \le 4\alpha^2$.

□

Movement of t

Lastly, we analyze the effect of setting $t \to t^{\text{new}}$.

Lemma 2.5.7 (Movement in t)**.** *For any x, s such that $\gamma_i^t(x, s) \le 1$ for all i, let $t^{\text{new}} = \left(1 - \frac{\kappa}{\sqrt{\nu}}\right)t$ where $\nu = \sum_{i=1}^m \nu_i$, we have*

$$\Phi^{t^{\text{new}}}(x, s) \le \Phi^t(x, s) + 10\kappa\lambda\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x, s))\right)^{1/2}.$$

Proof. Note that

$$\begin{aligned}
\gamma_i^{t^{\text{new}}}(x,s) &= \left\| \frac{s}{t^{\text{new}}} + \nabla\phi_i(x_i) \right\|_{x_i}^* \\
&= \left\| \frac{s}{t(1-\kappa/\sqrt{\nu})} + \nabla\phi_i(x_i) \right\|_{x_i}^* \\
&\leq (1+2\kappa/\sqrt{\nu})\gamma_i^t(x,s) + 2\|(\kappa/\sqrt{\nu})\nabla\phi_i(x_i)\|_{x_i}^* \\
&\leq (1+2\kappa/\sqrt{\nu})\gamma_i^t(x,s) + 3\kappa\sqrt{\nu_i}/\sqrt{\nu} \\
&\leq \gamma_i^t(x,s) + 5\kappa\sqrt{\nu_i}/\sqrt{\nu}
\end{aligned}$$

where the first step follows from definition, the second step follows from $t^{\text{new}} = t(1-\kappa/\sqrt{\nu})$, the second last step follows from the fact that our barriers are ν_i-self-concordant and the last step used $\gamma_i^t(x,s) \leq 1$ and $\nu_i \geq 1$. Using that $5\kappa \leq \frac{1}{10\lambda}$ and $\gamma_i^t(x,s) \leq 1$, we have by simple taylor expansion,

$$\begin{aligned}
\Phi^{t^{\text{new}}}(x,s) &\leq \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s)) + 2\lambda\sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s))\left(5\kappa\sqrt{\nu_i/\nu}\right) \\
&= \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s)) + 10\kappa\lambda\sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s))\left(\sqrt{\nu_i/\nu}\right) \\
&\leq \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s)) + 10\kappa\lambda\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x,s))\right)^{1/2}\left(\sum_{i=1}^m \frac{\nu_i}{\nu}\right)^{1/2} \\
&= \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,s)) + 10\kappa\lambda\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x,s))\right)^{1/2},
\end{aligned}$$

where the third step follows from Cauchy-Schwarz, and the last step follows from $\sum_{i=1}^m \nu_i = \nu$. □

Potential Maintenance

Putting it all together, we can show that our potential Φ^t can be maintained to be small throughout our algorithm.

Lemma 2.5.8 (Potential Maintenance)**.** *If* $\Phi^t(x,s) \leq 80\frac{m}{\alpha}$, *then*

$$\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \leq \left(1 - \frac{\alpha\lambda}{40\sqrt{m}}\right)\Phi^t(x,s) + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}).$$

In particularly, we have $\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \leq 80\frac{m}{\alpha}$.

Proof. Let

$$\zeta(x,s) = \left(\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(x,s))\right)^{1/2}.$$

By combining our previous lemmas,

$$\begin{aligned}
&\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \\
\leq&\Phi^t(x^{\text{new}}, s^{\text{new}}) + 10\kappa\lambda\cdot\zeta(x^{\text{new}}, s^{\text{new}}) \\
\leq&\Phi^t(x^{\text{new}}, x, s^{\text{new}}) + 12\alpha\lambda(\|\gamma_i^t(x,s)\|_\infty + 3\alpha)\cdot\zeta(x,s) + 10\kappa\lambda\cdot\zeta(x^{\text{new}}, s^{\text{new}}) \\
\leq&\Phi^t(x,x,s) - \frac{\alpha\lambda}{5}\zeta(\overline{x},\overline{s}) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha}) \\
&+ 12\alpha\lambda(\|\gamma_i^t(x,s)\|_\infty + 3\alpha)\cdot\zeta(x,s) + 10\kappa\lambda\cdot\zeta(x^{\text{new}}, s^{\text{new}}) \qquad (2.22)
\end{aligned}$$

where the first step follows from Lemma 2.5.7, the second step follows from Lemma 2.5.6, and the last step follows from Lemma 2.5.5. We note that in all lemma above, we used that fact that $\|\gamma_i^t\|_\infty \leq 1$ (for different combination of x, $\overline{x}$, x^{new}, s, $\overline{s}$, s^{new}) which we will show later.

By Lemma 2.5.4, we have that

$$\gamma_i^t(x^{\text{new}}, s^{\text{new}}) \leq \gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) + 2\alpha \leq \gamma_i^t(x,x,s) + 5\alpha. \qquad (2.23)$$

Hence, $\zeta(x^{\text{new}}, s^{\text{new}}) \leq 2\zeta(x,s)$.

Lemma 2.5.3 shows that $\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^* \leq 4\alpha$ and hence

$$\begin{aligned}
\zeta(\overline{x},\overline{s}) &\geq \frac{2}{3}\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},x,\overline{s}))\right)^{1/2} \\
&\geq \frac{2}{3}\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(x,x,s) - 8\alpha\lambda)\right)^{1/2} \\
&\geq \frac{1}{2}\zeta(x,s). \qquad (2.24)
\end{aligned}$$

Combining (2.23) and (2.24) into (2.22) gives

$$\begin{aligned}
&\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \\
\geq&\Phi^t(x,s) + \left(12\alpha\lambda(\|\gamma_i^t(x,s)\|_\infty + 3\alpha) + 20\kappa\lambda - \frac{\alpha\lambda}{10}\right)\cdot\zeta(x,s) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha}) \\
\geq&\Phi^t(x,s) + \left(12\alpha\lambda\|\gamma_i^t(x,s)\|_\infty - \frac{\alpha\lambda}{20}\right)\cdot\zeta(x,s) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha})
\end{aligned}$$

where we used $\kappa \leq \frac{\alpha}{1000}$ and $\alpha \leq \frac{1}{10000}$ on the second equation.

Finally, we need to bound $\|\gamma_i^t(x,s)\|_\infty$. The bound for other $\|\gamma_i^t\|_\infty$ are similar. We note that

$$\Phi^t(x,s) \leq 80\frac{m}{\alpha}$$

implies that $\|\gamma_i^t(x,s)\|_\infty \leq \frac{\log 80\frac{m}{\alpha}}{\lambda}$. Hence, by our choice of λ and α, we have that $\lambda \geq 480\log(80\frac{m}{\alpha})$ and hence

$$12\alpha\lambda\|\gamma_i^t(x,s)\|_\infty \leq \frac{\alpha\lambda}{40}.$$

Finally, using $\Phi^t(x,s) \leq \sqrt{m}\cdot\zeta(x,s)$, we have

$$\begin{aligned}\Phi^{t^{\text{new}}}(x^{\text{new}},s^{\text{new}}) &\geq \Phi^t(x,s) - \frac{\alpha\lambda}{40}\zeta(x,s) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha}) \\ &\geq \left(1-\frac{\alpha\lambda}{40\sqrt{m}}\right)\Phi^t(x,s) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha}).\end{aligned}$$

Since $\lambda \leq \frac{1}{400\sqrt{\alpha}}$, we have $\Phi^t(x,s) \leq 80\frac{m}{\alpha}$ implies $\Phi^{t^{\text{new}}}(x^{\text{new}},s^{\text{new}}) \leq 80\frac{m}{\alpha}$.

$\square$

Chapter 3

Inverse Maintenance

3.1 Central Path Maintenance

The goal of this section is to present a data-structure to perform our centering steps in $\widetilde{O}(n^{\omega-1/2})$ amortized time and prove a theoretical guarantee of it. The original idea of inverse maintenance is from Michael B. Cohen [49], then [18] used it to get faster running time for solving Linear Programs. Because a simple matrix vector product would require $O(n^2)$ time, our speedup comes via a low-rank embedding that provides ℓ_∞ guarantees, which is unlike the sparse vector approach of [18]. In fact, we are unsure if moving in a sparse direction h can have sufficiently controlled noise to show convergence. Here, we give a stochastic version that is faster for dense direction h.

We first present the guarantees of the inverse maintenance along with the full algorithm and proofs. The sketching guarantees for ℓ_∞ perturbations are given in Section 3.2. We combine our robust central path with the data structure guarantees to prove our final theorem in Section 3.3. Finally, we show initialization and termination can be done efficiently in Section 3.4.

Theorem 3.1.1 (Central path maintenance)**.** *Given a full rank matrix $A \in \mathbb{R}^{d\times n}$ with $n \geq d$, a*

Name	Type	Statement	Algorithm	Input	Output
INITIALIZE	public	Lemma 3.1.4	Alg. 1	$A, x, s, \overline{W}, \epsilon_{mp}, a, b$	$\emptyset$
UPDATE	public	Lemma 3.1.5	Alg. 2	$\overline{W}$	$\emptyset$
FULLUPDATE	private	Lemma 3.1.7	Alg. 3	$\overline{W}$	$\emptyset$
PARTIALUPDATE	private	Lemma 3.1.6	Alg. 2	$\overline{W}$	$\emptyset$
QUERY	public	Lemma 3.1.8	Alg. 1	$\emptyset$	$\overline{x}, \overline{s}$
MULTIPLYMOVE	public	Lemma 3.1.11	Alg. 4	h, t	$\emptyset$
MULTIPLY	private	Lemma 3.1.10	Alg. 4	h, t	$\emptyset$
MOVE	private	Lemma 3.1.9	Alg. 4	$\emptyset$	$\emptyset$

Table 3.1: Summary of data structure CENTRALPATHMAINTENANCE

tolerance parameter $0 < \epsilon_{mp} < 1/4$ and a block diagonal structure $n = \sum_{i=1}^m n_i$. Given any positive number a such $a \leq \alpha$ where α is the dual exponent of matrix multiplication. Given any linear sketch of size b, there is a randomized data structure CENTRALPATHMAINTENANCE *(in Algorithm 1, 2, 4) that approximately maintains the projection matrices*

$$\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$$

for positive block diagonal psd matrix $W \oplus_i \mathbb{R}^{n_i \times n_i}$; exactly implicitly maintains central path parameters (x, s) and approximately explicitly maintains path parameters through the following five operations:

1. INITIALIZE$(\overline{W}^{(0)}, \cdots)$ *: Assume $\overline{W}^{(0)} \in \otimes_i \mathbb{R}^{n_i \times n_i}$. Initialize all the parameters in $O(n^\omega)$ time.*

2. UPDATE$(\overline{W})$ *: Assume $\overline{W} \in \oplus_i \mathbb{R}^{n_i \times n_i}$. Output a block diagonal matrix $\widetilde{V} \oplus_i \mathbb{R}^{n_i \times n_i}$ such that*

$$(1 - \epsilon_{mp})\widetilde{v}_i \preceq \overline{w}_i \preceq (1 + \epsilon_{mp})\widetilde{v}_i.$$

3. QUERY() *: Output $(\overline{x}, \overline{s})$ such that $\|\overline{x} - x\|_{\widetilde{V}^{-1}} \leq \epsilon_{mp}$ and $\|\overline{s} - s\|_{\widetilde{V}} \leq t\epsilon_{mp}$ where t is the last t used in* MULTIPLYMOVE*, where $\epsilon_{mp} = \alpha \log^2(nT)\frac{n^{1/4}}{\sqrt{b}}$ and the success probability is $1 - 1/\operatorname{poly}(nT)$. This step takes $O(n)$ time.*

4. MULTIPLYMOVE(h, t) *: It outputs nothing. It implicitly maintains:*

$$x = x + \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h, s = s + t\widetilde{V}^{-1/2}\widetilde{P}\widetilde{V}^{1/2}h.$$

where $\widetilde{P} = \widetilde{V}^{1/2}A^\top(A\widetilde{V}A^\top)^{-1}A\widetilde{V}^{1/2}$. It also explicitly maintains $\overline{x}, \overline{s}$. Assuming t is decreasing, each call takes $O(nb + n^{a\omega+o(1)} + n^a\|h\|_0 + n^{1.5})$ amortized time.

Let $\overline{W}^{(0)}$ be the initial matrix and $\overline{W}^{(1)}, \cdots, \overline{W}^{(T)}$ be the (random) update sequence. Under the assumption that there is a sequence of matrix $W^{(0)}, \cdots, W^{(T)} \in \oplus_{i=1}^m \mathbb{R}^{n_i \times n_i}$ satisfies for all k

$$\left\|w_i^{-1/2}(\overline{w}_i - w_i)w_i^{-1/2}\right\|_F \leq \epsilon_{mp},$$

$$\sum_{i=1}^m \left\|(w_i^{(k)})^{-1/2}(\mathbf{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F^2 \leq C_1^2,$$

$$\sum_{i=1}^m \left(\mathbf{E}\left[\left\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F^2\right]\right)^2 \leq C_2^2,$$

$$\left\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F \leq \frac{1}{4}.$$

where $w_i^{(k)}$ is the i-th block of $W^{(k)}$, $\forall i \in [m]$.

Then, the amortized expected time per call of UPDATE(w) *is*

$$(C_1/\epsilon_{mp} + C_2/\epsilon_{mp}^2) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}).$$

Remark 3.1.2. *For our algorithm, we have $C_1 = O(1/\log^2 n)$, $C_2 = O(1/\log^4 n)$ and $\epsilon_{mp} = O(1/\log^2 n)$. Note that the input of* UPDATE *$\overline{W}$ can move a lot. It is working as long as $\overline{W}$ is close to some W that is slowly moving. In our application, our W satisfies C_1, C_2 deterministically. We keep it for possible future applications.*

Proof of Theorem 3.1.1

Definition of X and Y. Consider the k-th round of the algorithm. For all $i \in [m]$, matrix $\overline{y}_i^{(k)} \in \mathbb{R}^{n_i \times n_i}$ is constructed based on procedure UPDATE (Algorithm 2) :

$$\overline{y}_i^{(k)} = \frac{\overline{w}_i^{(k+1)}}{v_i^{(k)}} - I.$$

and $\overline{\pi}$ is a permutation such that $\|\overline{y}_{\overline{\pi}(i)}^{(k)}\|_F \geq \|\overline{y}_{\overline{\pi}(i+1)}^{(k)}\|_F$.

For the purpose of analysis : for all $i \in [m]$, we define $x_i^{(k)}$, $x_i^{(k)}$ and $y_i^{(k)} \in \mathbb{R}^{n_i \times n_i}$ as follows:

$$x_i^{(k)} = \frac{w_i^{(k)}}{v_i^{(k)}} - I, \quad y_i^{(k)} = \frac{w_i^{(k+1)}}{v_i^{(k)}} - I, \quad x_i^{(k+1)} = \frac{w_i^{(k+1)}}{v_i^{(k+1)}} - I,$$

where $\frac{w_i^{(k)}}{v_i^{(k)}}$ denotes $(v_i^{(k)})^{-1/2} w_i^{(k)} (v_i^{(k)})^{-1/2}$.

Note that the difference between $x_i^{(k)}$ and $y_i^{(k)}$ is that w is changing. The difference between $y_i^{(k)}$ and $x_i^{(k+1)}$ is that v is changing. For simplicity, we define $\beta_i = \|(w_i^{(k)})^{-1/2}(\mathbf{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F$, then one of assumption becomes $\sum_{i=1}^m \beta_i^2 \leq C_1^2$.

Assume sorting. Assume the diagonal blocks of matrix $x^{(k)} \in \oplus_{i=1}^m \mathbb{R}^{n_i \times n_i}$ are sorted such that $\|x_i^{(k)}\|_F \geq \|x_{i+1}^{(k)}\|_F$. Let τ and π are permutations such that $\|x_{\tau(i)}^{(k+1)}\|_F \geq \|x_{\tau(i+1)}^{(k+1)}\|_F$ and $\|y_{\pi(i)}^{(k)}\|_F \geq \|y_{\pi(i+1)}^{(k)}\|_F$.

Definition of Potential function. Let g be defined as

$$g_i = \begin{cases} n^{-a}, & \text{if } i < n^a; \\ i^{\frac{\omega-2}{1-a}} n^{-\frac{a(\omega-2)}{1-a}}, & \text{otherwise.} \end{cases}$$

Let ψ : square matrix $\to \mathbb{R}$ be defined by

$$\psi(x) = \begin{cases} \frac{\|x\|_F^2}{2\epsilon_{mp}}, & \|x\|_F \in [0, \epsilon_{mp}]; \\ \epsilon_{mp} - \frac{(4\epsilon_{mp}^2 - \|x\|_F^2)^2}{18\epsilon_{mp}^3}, & \|x\|_F \in (\epsilon_{mp}, 2\epsilon_{mp}]; \\ \epsilon_{mp}, & \|x\|_F \in (2\epsilon_{mp}, +\infty). \end{cases} \tag{3.1}$$

Algorithm 1 Central Path Maintenance Data Structure - Initial, Query, Move

```
1: datastructure CENTRALPATHMAINTENANCE    ▷ Theorem 3.1.1
2:
3: private : members
4:     $\overline{W} \in \otimes_{i\in[m]}\mathbb{R}^{n_i\times n_i}$    ▷ Target vector, $\overline{W}$ is $\epsilon_w$-close to $W$
5:     $V, \widetilde{V} \in \otimes_{i\in[m]}\mathbb{R}^{n_i\times n_i}$    ▷ Approximate vector
6:     $A \in \mathbb{R}^{d\times n}$    ▷ Constraints matrix
7:     $M \in \mathbb{R}^{n\times n}$    ▷ Approximate Projection Matrix
8:     $\epsilon_{mp} \in (0, 1/4)$    ▷ Tolerance
9:     $a \in (0, \alpha]$    ▷ Batch Size for Update ($n^a$)
10:    $b \in \mathbb{Z}_+$    ▷ Sketch size of one sketching matrix
11:    $R \in \mathbb{R}^{n^{1+o(1)}\times n}$    ▷ A list of sketching matrices
12:    $Q \in \mathbb{R}^{b\times n}$    ▷ Sketched matrices
13:    $u_1 \in \mathbb{R}^n, F \in \mathbb{R}^{n\times n}, u_2 \in \mathbb{R}^n$    ▷ Implicit representation of $x$, $x = u_1 + F\cdot u_2$
14:    $u_3 \in \mathbb{R}^n, G \in \mathbb{R}^{n\times n}, u_4 \in \mathbb{R}^n$    ▷ Implicit representation of $s$, $s = u_3 + G\cdot u_4$
15:    $\overline{x}, \overline{s} \in \mathbb{R}^n$    ▷ Central path parameters, maintain explicitly
16:    $l \in \mathbb{Z}_+$    ▷ Randomness counter, $R_l \in \mathbb{R}^{b\times n}$
17:    $t^{\mathrm{pre}} \in \mathbb{R}_+$    ▷ Tracking the changes of $t$
18: end members
19:
20: public : procedure INITIALIZE($A, x, s, W, \epsilon_{mp}, a, b$)    ▷ Lemma 3.1.4
21:                                ▷ parameters will *never change* after initialization
22:    $A \leftarrow A$, $a \leftarrow a$, $b \leftarrow b$, $\epsilon_{mp} \leftarrow \epsilon_{mp}$
23:                                ▷ parameters will *still change* after initialization
24:    $\overline{W} \leftarrow W$, $V \leftarrow W$, $\widetilde{V} \leftarrow V$
25:    Choose $R_l \in \mathbb{R}^{b\times n}$ to be sketching matrix, $\forall l \in [\sqrt{n}]$    ▷ Lemma 3.2.5
26:    $R \leftarrow [R_1^\top, R_2^\top, \cdots]^\top$    ▷ Batch them into one matrix $R$
27:    $M \leftarrow A^\top(AVA^\top)^{-1}A$, $Q \leftarrow R\sqrt{\widetilde{V}}M$    ▷ Initialize projection matrices
28:    $u_1 \leftarrow x$, $u_2 \leftarrow 0$, $u_3 \leftarrow s$, $u_4 \leftarrow 0$    ▷ Initialize $x$ and $s$
29:    $\overline{x} \leftarrow x$, $\overline{s} \leftarrow s$
30:    $l \leftarrow 1$
31: end procedure
32:
33: public : procedure QUERY()    ▷ Lemma 3.1.8
34:    return $(\overline{x}, \overline{s})$
35: end procedure
36:
37: end datastructure
```

where $\|x\|_F$ denotes the Frobenius norm of square matrix x, and let $L_1 = \max_x D_x\psi[h]/\|H\|_F$, $L_2 = \max_x D_x^2\psi[h,h]/\|H\|_F^2$ where h is the vectorization of matrix H.

Algorithm 2 Central Path Maintenance Data Structure - Update and PartialUpdate

```
1: datastructure CENTRALPATHMAINTENANCE                                ▷ Theorem 3.1.1
2:
3: public : procedure UPDATE($\overline{W}^{\text{new}}$)      ▷ Lemma 3.1.5, $\overline{W}^{\text{new}}$ is close to $W^{\text{new}}$
4:     $\overline{y}_i \leftarrow v_i^{-1/2}\overline{w}_i^{\text{new}}v_i^{-1/2} - 1, \forall i \in [m]$
5:     $r \leftarrow$ the number of indices $i$ such that $\|\overline{y}_i\|_F \geq \epsilon_{mp}$
6:     if $r < n^a$ then
7:         PARTIALUPDATE($\overline{W}^{\text{new}}$)
8:     else
9:         FULLUPDATE($\overline{W}^{\text{new}}$)                                ▷ Algorithm 3
10:    end if
11: procedure
12:
13: private : procedure PARTIALUPDATE($\overline{W}^{\text{new}}$)                ▷ Lemma 3.1.6
14:    $\overline{W} \leftarrow \overline{W}^{\text{new}}$
15:    $\widetilde{v}_i^{\text{new}} \leftarrow \begin{cases} v_i & \text{if } (1-\epsilon_{mp})v_i \preceq \overline{w}_i \preceq (1+\epsilon_{mp})v_i \\ w_i & \text{otherwise} \end{cases}$
16:    $F^{\text{new}} \leftarrow F + ((\widetilde{V}^{\text{new}})^{1/2} - (\widetilde{V})^{1/2})M$      ▷ only takes $n^{1+a}$ time, instead of $n^2$
17:    $G^{\text{new}} \leftarrow G + ((\widetilde{V}^{\text{new}})^{-1/2} - (\widetilde{V})^{-1/2})M$
18:    $u_1 \leftarrow u_1 + (F - F^{\text{new}})u_2$, $u_3 \leftarrow u_3 + (G - G^{\text{new}})u_4$
19:    $F \leftarrow F^{\text{new}}$, $G \leftarrow G^{\text{new}}$
20:    Let $\widehat{S}$ denote the blocks where $\widetilde{V}$ and $\widetilde{V}^{\text{new}}$ are different
21:    $\overline{x}_{\widehat{S}} \leftarrow (u_1)_{\widehat{S}} + (Fu_2)_{\widehat{S}}$, $\overline{s}_{\widehat{S}} \leftarrow (u_3)_{\widehat{S}} + (Gu_2)_{\widehat{S}}$ ▷ make sure $x$ and $\overline{x}$ are close, similarly for
   $s$ and $\overline{s}$
22: end procedure
23:
24: end datastructure
```

We define the potential at the k-th round by

$$\Phi_k = \sum_{i=1}^{m} g_i \cdot \psi(x_{\tau_k(i)}^{(k)})$$

where $\tau_k(i)$ is the permutation such that $\|x_{\tau_k(i)}^{(k)}\|_F \geq \|x_{\tau_k(i+1)}^{(k)}\|_F$.

Algorithm 3 Central Path Maintenance Data Structure - Full Update

```
1: datastructure CENTRALPATHMAINTENANCE                                        ▷ Theorem 3.1.1
2:
3: private : procedure FULLUPDATE($\overline{W}^{\text{new}}$)                 ▷ Lemma 3.1.7
4:     $\overline{y}_i \leftarrow v_i^{-1/2}\overline{w}_i^{\text{new}}v_i^{-1/2} - 1, \forall i \in [m]$
5:     $r \leftarrow$ the number of indices $i$ such that $\|\overline{y}_i\|_F \geq \epsilon_{mp}$
6:     Let $\overline{\pi} : [m] \rightarrow [m]$ be a sorting permutation such that $\|\overline{y}_{\overline{\pi}(i)}\|_F \geq \|\overline{y}_{\overline{\pi}(i+1)}\|_F$
7:     while $1.5 \cdot r < m$ and $\|\overline{y}_{\overline{\pi}(1.5r)}\|_F \geq (1 - 1/\log m)\|\overline{y}_{\overline{\pi}(r)}\|_F$
8:         $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, m)$
9:     end while
10:    $v_{\overline{\pi}(i)}^{\text{new}} \leftarrow \begin{cases} \overline{w}_{\overline{\pi}(i)}^{\text{new}} & i \in \{1, 2, \cdots, r\} \\ v_{\overline{\pi}(i)} & i \in \{r+1, \cdots, m\} \end{cases}$
11:                          ▷ Compute $M^{\text{new}} = A^\top(AV^{\text{new}}A^\top)^{-1}A$ via Matrix Woodbury
12:    $\Delta \leftarrow V^{\text{new}} - V$                                  ▷ $\Delta \in \mathbb{R}^{n\times n}$ and $\|\Delta\|_0 = r$
13:    $\Gamma \leftarrow \sqrt{V^{\text{new}}} - \sqrt{V}$
14:    Let $S \leftarrow \overline{\pi}([r])$ be the first $r$ indices in the permutation
15:    Let $M_{*,S} \in \mathbb{R}^{n\times O(r)}$ be the $r$ column-blocks from $S$ of $M$
16:    Let $M_{S,S}, \Delta_{S,S} \in \mathbb{R}^{O(r)\times O(r)}$ be the $r$ row-blocks and column-blocks from $S$ of $M$, $\Delta$
17:    $M^{\text{new}} \leftarrow M - M_{*,S} \cdot (\Delta_{S,S}^{-1} + M_{S,S})^{-1} \cdot (M_{*,S})^\top$      ▷ Update $M$
18:    $Q^{\text{new}} \leftarrow Q + R \cdot (\Gamma \cdot M^{\text{new}}) + R \cdot \sqrt{V} \cdot (M^{\text{new}} - M)$     ▷ Update $Q$
19:    $\overline{W} \leftarrow \overline{W}^{\text{new}}$, $V \leftarrow V^{\text{new}}$, $M \leftarrow M^{\text{new}}$, $Q \leftarrow Q^{\text{new}}$      ▷ Update in memory
20:    $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } (1-\epsilon_{mp})v_i \preceq \overline{w}_i \preceq (1+\epsilon_{mp})v_i \\ w_i & \text{otherwise} \end{cases}$
21:    $F^{\text{new}} \leftarrow \sqrt{\widetilde{V}}M$, $G^{\text{new}} \leftarrow \frac{1}{\sqrt{\widetilde{V}}}M$
22:    $u_1 \leftarrow u_1 + (F - F^{\text{new}})u_2$, $u_3 \leftarrow u_3 + (G - G^{\text{new}})u_4$
23:    $F \leftarrow F^{\text{new}}$, $G \leftarrow G^{\text{new}}$
24:    Let $\widehat{S}$ denote the blocks where $\widetilde{V}$ and $\widetilde{V}^{\text{new}}$ are different
25:    $\overline{x}_{\widehat{S}} \leftarrow (u_1)_{\widehat{S}} + (Fu_2)_{\widehat{S}}$, $\overline{s}_{\widehat{S}} \leftarrow (u_3)_{\widehat{S}} + (Gu_2)_{\widehat{S}}$ ▷ make sure $x$ and $\overline{x}$ are close, similarly for $s$ and $\overline{s}$
26:    $t^{\text{pre}} \leftarrow t$
27: end procedure
28:
29: end datastructure
```

Bounding the potential. We can express $\Phi_{k+1} - \Phi_k$ as follows:

$$\begin{aligned}\Phi_{k+1} - \Phi_k &= \sum_{i=1}^m g_i \cdot \left(\psi(x_{\tau(i)}^{(k+1)}) - \psi(x_i^{(k)})\right) \\ &= \sum_{i=1}^m g_i \cdot \underbrace{\left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_i^{(k)})\right)}_{w \text{ move}} - \sum_{i=1}^n g_i \cdot \underbrace{\left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\tau(i)}^{(k+1)})\right)}_{v \text{ move}}\end{aligned}$$

Algorithm 4 Central Path Maintenance Data Structure - Multiply and Move

```
datastructure CentralPathMaintenance                                   ▷ Theorem 3.1.1

public : procedure MultiplyAndMove$(h, t)$                              ▷ Lemma 3.1.11
    Multiply$(h, t)$
    Move$()$
end procedure

private : procedure Multiply $(h, t)$                                   ▷ Lemma 3.1.10
    Let $\widetilde{S}$ be the indices $i$ such that $(1-\epsilon_{mp})v_i \preceq \overline{w}_i \preceq (1+\epsilon_{mp})v_i$ is false.
    $\widetilde{\Delta} \leftarrow \widetilde{V} - V$
    $\widetilde{\Gamma} \leftarrow \sqrt{\widetilde{V}} - \sqrt{V}$
    $\delta_m \leftarrow ((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot ((M_{\widetilde{S},*})^\top \sqrt{\widetilde{V}} h))$      ▷ $|\widetilde{S}| \le n^a$
                                                  ▷ Compute $\widetilde{\delta}_x = \widetilde{V}^{1/2}(I - R^\top R\widetilde{P})\widetilde{V}^{1/2}h$
    $\widetilde{\delta}_x \leftarrow \widetilde{V}h - \Big((R_l^\top \cdot ((Q_l + R_l \cdot \widetilde{\Gamma} \cdot M) \cdot \sqrt{\widetilde{V}} \cdot h)) - (R_l^\top \cdot ((Q_{l,\widetilde{S}} + R_l \cdot \widetilde{\Gamma} \cdot M_{\widetilde{S},*}) \cdot \delta_m))\Big)$
                                                  ▷ Compute $\widetilde{\delta}_s = t\widetilde{V}^{-1/2}R^\top R\widetilde{P}\widetilde{V}^{1/2}h$
    $\widetilde{\delta}_s \leftarrow t \cdot \widetilde{V}^{-1} \cdot \Big((R_l^\top \cdot ((Q + R_l \cdot \widetilde{\Gamma} \cdot M) \cdot \sqrt{\widetilde{V}} \cdot h)) - (R_l^\top \cdot ((Q_{l,\widetilde{S}} + R_l \cdot \widetilde{\Gamma} \cdot M_{\widetilde{S},*}) \cdot \delta_m))\Big)$
    $l \leftarrow l + 1$    ▷ Increasing the randomness counter, and using the new randomness next time
                                                  ▷ Implicitly maintain $x = x + \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h$
    $u_1 \leftarrow u_1 + \widetilde{V}h$
    $u_2 \leftarrow u_2 - \sqrt{\widetilde{V}}h + \mathbf{1}_{\widetilde{S}}\delta_m$
                                                  ▷ Implicitly maintain $s = s + t\widetilde{V}^{-1/2}\widetilde{P}\widetilde{V}^{1/2}h$
    $u_3 \leftarrow u_3 + 0$
    $u_4 \leftarrow u_4 - t\sqrt{\widetilde{V}}h + t\mathbf{1}_{\widetilde{S}}\delta_m$
end procedure

private : procedure Move$()$                                            ▷ Lemma 3.1.9
    if $l > \sqrt{n}$ or $t \ge t^{\mathrm{pre}}/2$                       ▷ Variance is large enough
        $x \leftarrow u_1 + Fu_2$, $s \leftarrow u_3 + Fu_4$
        Initialize$(A, x, s, \overline{W}, \epsilon_{mp}, a, b)$              ▷ Algorithm 1
    else
        $\overline{x} \leftarrow \overline{x} + \widetilde{\delta}_x$, $\overline{s} \leftarrow \overline{s} + \widetilde{\delta}_s$          ▷ Update $\overline{x}, \overline{s}$
    end if
return $(\overline{x}, \overline{s})$
end procedure

end datastructure
```

Initialization time, update time, query time, move time, multiply time

Remark 3.1.3. *In terms of implementing this data-structure, we only need three operations* INITIALIZE, UPDATE, *and* QUERY. *However, in order to make the proof more understoodable, we split* UPDATE *into many operations :* FULLUPDATE, PARTIALUPDATE, MULTIPLY *and* MOVE. *We give a list of operations in Table 3.1.*

Lemma 3.1.4 (Initialization)**.** *The initialization time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 1) is $O(n^{\omega+o(1)})$.*

Proof. The running time is mainly dominated by two parts, the first part is computing $A^\top(AVA^\top)^{-1}A$, this takes $O(n^2 d^{\omega-2})$ time.

The second part is computing $R\sqrt{\widetilde{V}}M$. This takes $O(n^{\omega+o(1)})$ time. □

Lemma 3.1.5 (Update time)**.** *The update time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 2) is $O(rg_r n^{2+o(1)})$ where r is the number of indices we updated in V.*

Proof. It is trivially follows from combining Lemma 3.1.6 and Lemma 3.1.7. □

Lemma 3.1.6 (Partial Update time)**.** *The partial update time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 2) is $O(n^{1+a})$.*

Proof. We first analyze the running time of F update, the update equation of F in algorithm is

$$\begin{aligned} F^{\text{new}} &\leftarrow F + ((\widetilde{V}^{\text{new}})^{1/2} - (\widetilde{V})^{1/2})M \\ F &\leftarrow F^{\text{new}} \end{aligned}$$

which can be implemented as

$$F \leftarrow F + ((\widetilde{V}^{\text{new}})^{1/2} - (\widetilde{V})^{1/2})M$$

where we only need to change n^a row-blocks of F. It takes $O(n^{1+a})$ time.

Similarly, for the update time of G.

Next we analyze the update time of u_1, the update equation of u_1 is

$$u_1 \leftarrow u_1 + (F - F^{\text{new}})u_2$$

Note that the difference between F and F^{new} is only n^a row-blocks, thus it takes n^{1+a} time to update.

Finally we analyze the update time of $\overline{x}$. Let $\widehat{S}$ denote the blocks where $\widetilde{V}$ and $\widetilde{V}^{\text{new}}$ are different.

$$\overline{x}_{\widehat{S}} \leftarrow (u_1)_{\widehat{S}} + (Fu_2)_{\widehat{S}}$$

This also can be done in n^{1+a} time, since $\widehat{S}$ indicates only n^a blocks.

Therefore, the overall running time is $O(n^{1+a})$. □

Lemma 3.1.7 (Full Update time)**.** *The full update time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 3) is $O(rg_r n^{2+o(1)})$ where r is the number of indices we updated in V.*

Proof. The update equation we use for Q is

$$Q^{\text{new}} \leftarrow Q + R \cdot (\Gamma \cdot M^{\text{new}}) + R \cdot \sqrt{V} \cdot (M^{\text{new}} - M).$$

It can be re-written as

$$Q^{\text{new}} \leftarrow Q + R \cdot (\Gamma \cdot M^{\text{new}}) + R \cdot \sqrt{V} \cdot (-M_{*,S} \cdot (\Delta_{S,S}^{-1} + M_{S,S})^{-1} \cdot (M_{*,S})^{\top})$$

The running time of computing second term is multiplying a $n \times r$ matrix with another $r \times n$ matrix. The running time of computing third term is also dominated by multiplying a $n \times r$ matrix with another $r \times n$ matrix.

Thus running time of processing Q update is the same as the processing M update.

For the running time of other parts, it is dominated by the time of updating M and Q.

Therefore, the rest of the proof is almost the same as Lemma 5.4 in [18], we omitted here. □

Lemma 3.1.8 (Query time)**.** *The query time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 1) is $O(n)$ time.*

Proof. This takes only $O(n)$ time, since we stored $\overline{x}$ and $\overline{s}$. □

Lemma 3.1.9 (Move time)**.** *The move time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 4) is $O(n^{\omega+o(1)})$ time in the worst case, and is $O(n^{\omega-1/2+o(1)})$ amortized cost per iteration.*

Proof. In one case, it takes only $O(n)$ time. For the other case, the running time is dominated by INITIALIZE, which takes $n^{\omega+o(1)}$ by Lemma 3.1.7.

□

Lemma 3.1.10 (Multiply time)**.** *The multiply time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 4) is $O(nb + n^{1+a+o(1)})$ for dense vector $\|h\|_0 = n$, and is $O(nb + n^{a\omega+o(1)} + n^a\|h\|_0)$ for sparse vector h.*

Proof. We first analyze the running time of computing vector δ_m, the equation is

$$\delta_m \leftarrow \left(((\widetilde{\Delta}_{\widetilde{S},\widetilde{S}})^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot (M_{\widetilde{S},*})^{\top} \sqrt{\widetilde{V}} h \right)$$

where $\widetilde{\Delta} = \widetilde{V} - V$. Let $\widetilde{r} = \sum_{i \in \widetilde{S}} n_i = O(r)$ where r is the number of blocks are different in $\widetilde{V}$ and V.

It contains several parts:

1. Computing $\widetilde{M}_{\widetilde{S}}^{\top} \cdot (\sqrt{\widetilde{V}} h) \in \mathbb{R}^{\widetilde{r}}$ takes $O(\widetilde{r})\|h\|_0$.

2. Computing $(\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} \in \mathbb{R}^{O(\widetilde{r}) \times O(\widetilde{r})}$ that is the inverse of a $O(\widetilde{r}) \times O(\widetilde{r})$ matrix takes $O(\widetilde{r}^{\omega+o(1)})$ time.

3. Computing matrix-vector multiplication between $O(\widetilde{r}) \times O(\widetilde{r})$ matrix $((\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1})$ and $O(\widetilde{r}) \times 1$ vector $((\widetilde{M}_{\widetilde{S},*})^\top \sqrt{\widetilde{V}} h)$ takes $O(\widetilde{r}^2)$ time.

Thus, the running time of computing δ_m is

$$O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + \widetilde{r}^2) = O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)}).$$

Next, we want to analyze the update equation of $\widetilde{\delta}_x$

$$\widetilde{\delta}_x \leftarrow \widetilde{V}h - \Big((R_l^\top \cdot ((Q_l + R_l\sqrt{\widetilde{\Delta}} M) \cdot \sqrt{\widetilde{V}} \cdot h)) - (R_l^\top \cdot ((Q_{l,\widetilde{S}} + R_l \widetilde{\Gamma} M_{\widetilde{S}}) \cdot \delta_m)) \Big)$$

where $\widetilde{\Gamma} = \sqrt{\widetilde{V}} - \sqrt{V}$ has $O(r)$ non-zero blocks.

It is clear that the running time is dominated by the second term in the equation. We only focus on that term.

1. Computing $R_l^\top Q_l \sqrt{\widetilde{V}} h$ takes $O(bn)$ time, because $Q_l, R_l \in \mathbb{R}^{b \times n}$.

2. Computing $R_l^\top R_l \sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h$ takes $O(bn + b\widetilde{r} + \widetilde{r}\|h\|_0)$ time. The reason is, computing $\sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h$ takes $\widetilde{r}\|h\|_0$ time, computing $R_l \cdot (\sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h)$ takes $b\widetilde{r}$, then finally computing $R_l^\top \cdot (R_l \sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h)$ takes nb.

Last, the update equation of u_1, u_2, u_3, u_4 only takes the $O(n)$ time.

Finally, we note that $r \leq O(n^a)$ due to the guarantee of FULLUPDATE and PARTIALUPDATE.

Thus, overall the running time of the MULTIPLY is

$$\begin{aligned} O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + \widetilde{r}^2 + b\widetilde{r} + nb) &= O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + nb) \\ &= O(r\|h\|_0 + r^{\omega+o(1)} + nb) \\ &= O(n^a\|h\|_0 + n^{a\omega+o(1)} + nb) \end{aligned}$$

where the first step follows from $b\widetilde{r} \leq nb$ and $\widetilde{r}^2 \leq \widetilde{r}^{\omega+o(1)}$, and the second step follows from $\widetilde{r} = O(r)$, and the last step follows from $r = O(n^a)$.

If h is the dense vector, then the overall time is

$$O(nb + n^{1+a} + n^{a\omega+o(1)}).$$

Based on Lemma 5.5 in [18], we know that $a\omega \leq 1 + a$. Thus, it becomes $O(nb + n^{1+a+o(1)})$ time.

If h is a sparse vector, then the overall time is

$$O(nb + n^a\|h\|_0 + n^{a\omega+o(1)}).$$

□

Lemma 3.1.11 (MULTIPLYMOVE)**.** *The running time of* MULTIPLYMOVE *(Algorithm 3.1.11) is the* MULTIPLY *time plus* MOVE *time.*

Bounding W move

Lemma 3.1.12 (W move)**.**

$$\sum_{i=1}^{m} g_i \cdot \mathbf{E}\left[\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\pi(i)}^{(k)})\right] = O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}).$$

Proof. Let $I \subseteq [m]$ be the set of indices such that $\|x_i^{(k)}\|_F \leq 1$. We separate the term into two :

$$\sum_{i=1}^{m} g_i \cdot \mathbf{E}[\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\pi(i)}^{(k)})] = \sum_{i\in I} g_{\pi^{-1}(i)} \cdot \mathbf{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] + \sum_{i\in I^c} g_{\pi^{-1}(i)} \cdot \mathbf{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})].$$

Case 1: Terms from I. Let $\mathrm{vec}(y_i^{(k)})$ denote the vectorization of matrix $y_i^{(k)}$. Similarly, $\mathrm{vec}(x_i^{(k)})$ denotes the vectorization of $x_i^{(k)}$. Mean value theorem shows that

$$\begin{aligned}
\psi(y_i^{(k)}) - \psi(x_i^{(k)}) &= \langle \psi'(x_i^{(k)}), y_i^{(k)} - x_i^{(k)} \rangle + \frac{1}{2}\mathrm{vec}(y_i^{(k)} - x_i^{(k)})^\top \psi''(\zeta)\mathrm{vec}(y_i^{(k)} - x_i^{(k)}) \\
&\leq \langle \psi'(x_i^{(k)}), y_i^{(k)} - x_i^{(k)} \rangle + \frac{L_2}{2}\|y_i^{(k)} - x_i^{(k)}\|_F^2 \\
&= \langle \psi'(x_i^{(k)}), (v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2} \rangle \\
&\quad + \frac{L_2}{2}\|(v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F^2
\end{aligned}$$

where the second step follows from definition of L_2 (see Part 4 of Lemma 3.1.17).

Taking conditional expectation given $w^{(k)}$ on both sides

$$\begin{aligned}
\mathbf{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] &\leq \langle \psi'(x_i^{(k)}), (v_i^{(k)})^{-1/2}(\mathbf{E}[w_i^{(k+1)}] - w_i^{(k)})(v_i^{(k)})^{-1/2} \rangle \\
&\quad + \frac{L_2}{2}\mathbf{E}[\|(v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F^2] \\
&\leq L_1\|(v_i^{(k)})^{-1/2}(\mathbf{E}[w_i^{(k+1)}] - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F \\
&\quad + \frac{L_2}{2}\mathbf{E}[\|(v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F^2] \\
&\leq L_1\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 \cdot \|(w_i^{(k)})^{-1/2}(\mathbf{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F \\
&\quad + \frac{L_2}{2}\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^4 \cdot \mathbf{E}[\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F^2] \\
&= L_1\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 \cdot \beta_i + \frac{L_2}{2}\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^4 \cdot \gamma_i
\end{aligned}$$

where the second step follows from definition of L_2 (see Part 4 of Lemma 3.1.17), the third step follows from $\|AB\|_F \leq \|A\|_F \cdot \|B\|$, and the last step follows from defining β_i and γ_i as follows:

$$\begin{aligned}
\beta_i &= \left\|(w_i^{(k)})^{-1/2}(\mathbf{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F \\
\gamma_i &= \mathbf{E}\left[\left\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F^2\right]
\end{aligned}$$

To bound $\sum_{i\in I} g_{\pi^{-1}(i)}\,\mathbf{E}[\psi(y_i^{(k)})-\psi(x_i^{(k)})]$, we need to bound the following two terms,

$$\sum_{i\in I} g_{\pi^{-1}(i)}L_1\left\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\right\|^2\beta_i,\text{ and }\sum_{i\in I} g_{\pi^{-1}(i)}\frac{L_2}{2}\left\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\right\|^4\gamma_i$$

For the first term, we have

$$\begin{aligned}\sum_{i\in I} g_{\pi^{-1}(i)}L_1\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2\beta_i &\le \left(\sum_{i\in I}\left(g_{\pi^{-1}(i)}L_1\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2\right)^2\sum_{i\in I}\beta_i^2\right)^{1/2}\\ &\le O(L_1)\left(\sum_{i=1}^n g_i^2\cdot C_1^2\right)^{1/2}\\ &= O(C_1L_1\|g\|_2). \qquad (3.2)\end{aligned}$$

where the first step follows from Cauchy-Schwarz inequality, the second step follows from $n_i = O(1)$ and $\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 = O(1)$.

For the second term, we have

$$\sum_{i\in I} g_{\pi^{-1}(i)}\frac{L_2}{2}\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^4 n_i\gamma_i \le O(L_2)\cdot\sum_{i=1}^m g_i\cdot\gamma_i = O(C_2L_2\|g\|_2). \qquad (3.3)$$

Now, combining Eq. (3.2) and Eq. (3.3) and using that $L_1 = O(1)$, $L_2 = O(1/\epsilon_{mp})$ (from part 4 of Lemma 3.1.17) and $\|g\|_2 \le \sqrt{\log n}\cdot O(n^{-a/2}+n^{\omega-5/2})$ (from Lemma 3.1.13), we have

$$\sum_{i\in I} g_{\pi^{-1}(i)}\cdot\mathbf{E}[\psi(y_i^{k)})-\psi(x_i^{(k)})] \le O(C_1+C_2/\epsilon_{mp})\cdot\sqrt{\log n}\cdot(n^{-a/2}+n^{\omega-5/2}).$$

Case 2: Terms from I^c.

For all $i\in I^c$, we have $\|x_i^{(k)}\|_F \ge 1$. Note that $\psi(x)$ is constant for $\|x\|_F^2 \ge (2\epsilon_{mp})^2$ and that $\epsilon_{mp} \le 1/4$. Therefore, if $\|y_i^{(k)}\|_F \ge 1/2$, we have that $\psi(y_i^{(k)})-\psi(x_i^{(k)}) = 0$. Hence, we only need to consider the $i\in I^c$ such that $\|y_i^{(k)}\|_F < 1/2$. For these i, we have that

$$\begin{aligned}\frac{1}{2} &< \|y_i^{(k)}-x_i^{(k)}\|_F\\ &= \|(v_i^{(k)})^{-1/2}(w_i^{(k+1)}-w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F\\ &= \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\cdot(w_i^{(k)})^{-1/2}(w_i^{(k+1)}-w_i^{(k)})(w_i^{(k)})^{-1/2}\cdot(w_i^{(k)})^{1/2}(v_i^{(k)})^{-1/2}\|_F\\ &\le \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|\cdot\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)}-w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F\cdot\|(w_i^{(k)})^{1/2}(v_i^{(k)})^{-1/2}\|\\ &= \|(v_i^{(k)})^{-1/2}w_i^{(k)}(v_i^{(k)})^{-1/2}\|\cdot\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)}-w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F\\ &\le \frac{3}{2}\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)}-w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F\end{aligned}$$

where we used that $\|y_i^{(k)}\|_F = \|\frac{w_i^{(k+1)}}{v_i^{(k)}} - I\|_F \le 1/2$. Hence, the above long equation gives us

$$\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F > 1/3$$

and hence it has $> 1/4$, which is impossible (because we assume it is $\le 1/4$).

Thus, we have

$$\sum_{i \in I^c} g_{\pi^{-1}(i)} \cdot \mathbf{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] = 0.$$

□

We state a Lemma that was proved in previous work [18].

Lemma 3.1.13 (Lemma 5.8 in [18])**.**

$$\left(\sum_{i=1}^n g_i^2\right)^{1/2} \le \sqrt{\log n} \cdot O(n^{-a/2} + n^{\omega - 5/2})$$

Bounding V move

Definition 3.1.14. *We define block diagonal matrices $X^{(k)}$, $Y^{(k)}$, $X^{(k+1)}$ and $\overline{Y}^{(k)}$ $\otimes_{i \in [m]} \mathbb{R}^{n_i \times n_i}$ as follows*

$$x_i^{(k)} = \frac{w_i^{(k)}}{v_i^{(k)}} - I, \quad y_i^{(k)} = \frac{w_i^{(k+1)}}{v_i^{(k)}} - I, \quad x_i^{(k+1)} = \frac{w_i^{(k+1)}}{v_i^{(k+1)}} - I, \quad \overline{y}_i^{(k)} = \frac{\overline{w}_i^{(k+1)}}{v_i^{(k)}} - I.$$

Let ϵ_w denote the error between W and $\overline{W}$

$$\|W_i^{-1/2}(\overline{W}_i - W_i)W_i^{-1/2}\|_F \le \epsilon_w.$$

Lemma 3.1.15 (V move)**.** *We have,*

$$\sum_{i=1}^n g_i \cdot \left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\tau(i)}^{(k+1)})\right) \ge \Omega(\epsilon_{mp} r_k g_{r_k} / \log n).$$

Proof. To prove the Lemma, we will split the proof into two cases.

Let us first understand several simple facts which are useful in the later proof. Note that from the definition of the algorithm, we only change the block if $\|\overline{y}_i^{(k)}\|_F$ is larger than the error between w_i and $\overline{w}_i$. Hence, all the changes only decreases the norm, namely $\psi(y_i^{(k)}) \ge \psi(x_i^{(k+1)})$ for all i. So is their sorted version $\psi(y_{\pi(i)}^{(k)}) \ge \psi(x_{\tau(i)})^{(k+1)}$ for all i.

Case 1. We exit the while loop when $1.5r_k \geq n$.

Let u^* denote the largest u such that $\|\overline{y}^{(k)}_{\overline{\pi}(u)}\|_F \geq \epsilon_{mp}$. If $u^* = r_k$, we have that $\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq \epsilon_{mp} \geq \epsilon_{mp}/100$. Otherwise, the condition of the loop shows that

$$\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq (1-1/\log n)^{\log_{1.5} r_k - \log_{1.5} u^*} \|\overline{y}^{(k)}_{\overline{\pi}(u^*)}\|_F \geq (1-1/\log n)^{\log_{1.5} n} \epsilon_{mp} \geq \epsilon_{mp}/100.$$

where we used that $n \geq 4$.

According to definition of $x^{(k+1)}_{\tau(i)}$, we have

$$\begin{aligned}\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) &\geq \sum_{i=n/3+1}^{2n/3} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) \\ &\geq \sum_{i=n/3+1}^{2n/3} g_i \cdot (\Omega(\epsilon_{mp}) - O(\epsilon_w)) \\ &\geq \sum_{i=n/3+1}^{2n/3} g_i \cdot \Omega(\epsilon_{mp}) \\ &= \Omega(r_k g_{r_k} \epsilon_{mp}).\end{aligned}$$

where the second step follows from $\|y^{(k)}_{\pi(i)}\|_F \geq \|y^{(k)}_{\pi(r_k)}\|_F \geq (1-O(\epsilon_w))\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq \epsilon_{mp}/200$ for all $i < 2n/3$.

Case 2. We exit the while loop when $1.5r_k < n$ and $\|\overline{y}^{(k)}_{\overline{\pi}(1.5r_k)}\|_F < (1-1/\log n)\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F$.

By the same argument as Case 1, we have that $\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq \epsilon_{mp}/100$. Part 3 of Lemma 3.1.17 together with the fact

$$\|\overline{y}^{(k)}_{\overline{\pi}(1.5r)}\|_F < \min\left(\epsilon_{mp}, \|\overline{y}^{(k)}_{\overline{\pi}(r)}\|_F \cdot (1-1/\log n)\right),$$

shows that

$$\psi(\overline{y}^{(k)}_{\overline{\pi}(1.5r)}) - \psi(\overline{y}^{(k)}_{\overline{\pi}(r)}) = \Omega(\epsilon_{mp}/\log n). \tag{3.4}$$

Now the question is, how to relax $\psi(\overline{y}^{(k)}_{\overline{\pi}(1.5r)})$ to $\psi(y^{(k)}_{\pi(1.5r)})$ and how to relax $\psi(\overline{y}^{(k)}_{\overline{\pi}(r)})$ to $\psi(y^{(k)}_{\pi(r)})$

Note that $\|y^{(k)}_i\|_F \geq \|x^{(k+1)}_i\|_F$ for all i. Hence, we have $\psi(y^{(k)}_{\pi(i)}) \geq \psi(x^{(k+1)}_{\tau(i)})$ for all i.

Recall the definition of y, $\overline{y}$, π and $\overline{\pi}$,

$$y^{(k)}_i = \frac{w^{(k+1)}_i}{v^{(k)}_i} - I, \quad \overline{y}^{(k)}_i = \frac{\overline{w}^{(k+1)}_i}{v^{(k)}_i} - I.$$

and π and $\overline{\pi}$ denote the permutations such that $\|y^{(k)}_{\pi(i)}\|_F \geq \|y^{(k)}_{\pi(i+1)}\|_F$ and $\|\overline{y}^{(k)}_{\overline{\pi}(i)}\|_F \geq \|\overline{y}^{(k)}_{\overline{\pi}(i+1)}\|_F$.

Using Fact 3.1.16 and $\|\cdot\|_2 = \Theta(1)\|\cdot\|_F$ when the matrix has constant dimension

$$\|y_{\pi(i)}^{(k)} - \overline{y}_{\pi(i)}^{(k)}\|_F \leq O(\epsilon_w).$$

where ϵ_w is the error between W and $\overline{W}$.

Next, $\forall i$, we have

$$\psi(y_{\pi(i)}^{(k)}) = \psi(\overline{y}_{\pi(i)}^{(k)}) \pm O(\epsilon_w \epsilon_{mp}) \tag{3.5}$$

Next, we note that all the blocks the algorithm updated must lies in the range $i = 1, \cdots, \frac{3r_k}{2} - 1$. After the update, the error of r_k of these block becomes so small that its rank will much higher than r_k. Hence, $r_k/2$ of the unchanged blocks in the range $i = 1, \cdots, \frac{3r_k}{2}$ will move earlier in the rank. Therefore, the $r_k/2$-th element in $x^{(k+1)}$ must be larger than the $\frac{3}{2}r_k$-th element in $y^{(k)}$. In short, we have $\psi(x_{\tau(i)}^{(k+1)}) \leq \psi(y_{\pi(1.5r_k)}^{(k)})$ for all $i \geq r_k/2$.

Putting it all together, we have

$$\begin{aligned}
&\sum_{i=1}^{n} g_i \cdot \left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\tau(i)}^{(k+1)})\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\tau(i)}^{(k+1)})\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(y_{\pi(i)}^{(k)}) - \psi(y_{\tau(1.5r_k)}^{(k+1)})\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(\overline{y}_{\pi(i)}^{(k)}) - \psi(\overline{y}_{\tau(1.5r_k)}^{(k+1)}) - O(\epsilon_w)\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(\overline{y}_{\pi(r_k)}^{(k)}) - \psi(\overline{y}_{\tau(1.5r_k)}^{(k+1)}) - O(\epsilon_w)\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_{r_k} \cdot \left(\Omega(\frac{\epsilon_{mp}}{\log n}) - O(\epsilon_w)\right) && \text{by (3.4)} \\
\geq &\sum_{i=r_k/2}^{r_k} g_{r_k} \cdot \Omega(\frac{\epsilon_{mp}}{\log n}) && \text{by } \epsilon_w < O(\epsilon_{mp}/\log n) \\
= &\,\Omega\left(\epsilon_{mp} r_k g_{r_k} / \log n\right).
\end{aligned}$$

Therefore, we complete the proof. □

Fact 3.1.16. *Given two length n positive vectors a, b. Let a be sorted such that $a_i \geq a_{i+1}$. Let π denote the permutation such that $b_{\pi(i)} \geq b_{\pi(i+1)}$. If for all $i \in [n]$, $|a_i - b_i| \leq \epsilon a_i$. Then for all $i \in [n]$, $|a_i - b_{\pi(i)}| \leq \epsilon a_i$.*

Proof. Case 1. $\pi(i) = i$. This is trivially true.

Case 2. $\pi(i) < i$. We have

$$b_{\pi(i)} \geq b_i \geq (1-\epsilon)a_i$$

Since $\pi(i) < i$, we know that there exists a $j > i$ such that $\pi(j) < \pi(i)$. Then we have

$$b_{\pi(i)} \leq b_{\pi(j)} \leq (1+\epsilon)a_j \leq (1+\epsilon)a_i$$

Combining the above two inequalities, we have $(1-\epsilon)a_i \leq b_{\pi(i)} \leq (1+\epsilon)a_i$.

Case 3. $\pi(i) > i$. We have

$$b_{\pi(i)} \leq b_i \leq (1+\epsilon)a_i$$

Since $\pi > i$, we know that there exists $j < i$ such that $\pi(j) > \pi(i)$. Then we have

$$b_{\pi(i)} \geq b_{\pi(j)} \geq (1-\epsilon)a_j \geq (1-\epsilon)a_i.$$

Combining the above two inequalities gives us $(1-\epsilon)a_i \leq b_{\pi(i)} \leq (1+\epsilon)a_i$.

Therefore, putting all the three cases together completes the proof.

$\square$

Potential function ψ

[18] used a scalar version potential function. Here, we generalize it to the matrix version.

Lemma 3.1.17 (Matrix version of Lemma 5.10 in [18]). *Let function ψ : square matrix $\rightarrow \mathbb{R}$ (defined as Eq. (3.1)) satisfies the following properties :*
1. Symmetric ($\psi(x) = \psi(-x)$) and $\psi(0) = 0$
2. If $\|x\|_F \geq \|y\|_F$, then $\psi(x) \geq \psi(y)$
3. $|f'(x)| = \Omega(1/\epsilon_{mp}), \forall x \in [(0.01\epsilon_{mp})^2, \epsilon_{mp}^2]$
4. $L_1 \overset{\text{def}}{=} \max_x \frac{D_x\psi[H]}{\|H\|_F} = 2$ and $L_2 \overset{\text{def}}{=} \max_x \frac{D_x^2\psi[H,H]}{\|H\|_F^2} = 10/\epsilon_{mp}$
5. $\psi(x)$ is a constant for $\|x\|_F \geq 2\epsilon_{mp}$

Proof. Let $f : \mathbb{R}_+ \rightarrow \mathbb{R}$ be defined as

$$f(x) = \begin{cases} \frac{x^2}{2\epsilon_{mp}^3}, & x \in [0, \epsilon_{mp}^2]; \\ \epsilon_{mp} - \frac{(4\epsilon_{mp}^2 - x)^2}{18\epsilon_{mp}^3}, & x \in (\epsilon_{mp}^2, 4\epsilon_{mp}^2]; \\ \epsilon_{mp}, & x \in (4\epsilon_{mp}^2, +\infty). \end{cases}$$

We can see that

$$f(x)' = \begin{cases} \frac{x}{\epsilon_{mp}^3}, & x \in [0, \epsilon_{mp}^2]; \\ \frac{4\epsilon_{mp}^2 - x}{9\epsilon_{mp}^3}, & x \in (\epsilon_{mp}^2, 4\epsilon_{mp}^2]; \\ 0, & x \in (4\epsilon_{mp}^2, +\infty). \end{cases} \text{ and } f(x)'' = \begin{cases} \frac{1}{\epsilon_{mp}^3}, & x \in [0, \epsilon_{mp}^2]; \\ -\frac{1}{9\epsilon_{mp}^3}, & x \in (\epsilon_{mp}^2, 4\epsilon_{mp}^2]; \\ 0, & |x| \in (4\epsilon_{mp}^2, +\infty). \end{cases}$$

It implies that $\max_x |f(x)'| \leq \frac{1}{\epsilon_{mp}}$ and $\max_x |f(x)''| \leq \frac{1}{\epsilon_{mp}^3}$. Let $\psi(x) = f(\|X\|_F^2)$.

Proof of Part 1,2 and 5. These proofs are pretty standard from definition of ψ.

Proof of Part 3. This is trivially following from definition of scalar function f.

Proof of Part 4. By chain rule, we have

$$\begin{aligned} D_x\psi[h] &= 2f'(\|X\|_F^2) \cdot \operatorname{tr}[XH] \\ D_x^2\psi[h,h] &= 2f''(\|X\|_F^2) \cdot (\operatorname{tr}[XH])^2 + 2f'(\|x\|_F^2) \cdot \operatorname{tr}[H^2] \end{aligned}$$

where x is the vectorization of matrix X and h is the vectorization of matrix H. We can upper bound

$$|D_x\psi[h]| \leq 2|f'(\|X\|_F^2)| \cdot |\operatorname{tr}[XH]| \leq 2|f'(\|X\|_F^2)| \cdot \|X\|_F \cdot \|H\|_F$$

Then, we have

$$|f'(\|X\|_F^2)| \cdot \|X\|_F = \begin{cases} \|X\|_F^3/\epsilon_{mp}^3 \leq 1, & \|X\|_F \in [0, \epsilon_{mp}] \\ (4\epsilon_{mp}^2 - \|X\|_F^2)\|X\|_F/9\epsilon_{mp} \leq 2/3, & \|X\|_F \in (\epsilon_{mp}, 2\epsilon_{mp}] \\ 0, & \|X\|_F \in (2\epsilon_{mp}, +\infty) \end{cases}$$

It implies that $|D_x\psi[h]| \leq 2\|H\|_F, \forall x$.

By case analysis, we have

$$|f''(\|X\|_F^2)| \cdot \|X\|_F^2 \leq \begin{cases} \frac{1}{\epsilon_{mp}^3}\|X\|_F^2 \leq 4/\epsilon_{mp}, & \|X\|_F^2 \in [0, 4\epsilon_{mp}^2] \\ 0, & \|X\|_F^2 \in (4\epsilon_{mp}, +\infty) \end{cases}$$

We can also upper bound

$$\begin{aligned} |D_x^2\psi[h,h]| &\leq 2|f''(\|X\|_F^2)| \cdot (\operatorname{tr}[XH])^2 + 2|f'(\|X\|_F^2)| \cdot \operatorname{tr}[H^2] \\ &\leq 2|f''(\|X\|_F^2)| \cdot (\|X\|_F\|H\|_F)^2 + 2|f'(\|X\|_F^2)| \cdot \|H\|_F^2 \\ &\leq 2 \cdot \frac{4}{\epsilon_{mp}}\|H\|_F^2 + 2 \cdot \frac{1}{\epsilon_{mp}}\|H\|_F^2 \\ &= \frac{10}{\epsilon_{mp}}\|H\|_F^2. \end{aligned}$$

□

x and $\overline{x}$ are close

Lemma 3.1.18 (x and $\overline{x}$ are close in term of $\widetilde{V}^{-1}$)**.** *With probability* $1-\delta$ *over the randomness of sketching matrix* $R \in \mathbb{R}^{b\times n}$*, we have*

$$\|\overline{x}_i - x_i\|_{\widetilde{V}_i^{-1}} \leq \epsilon_x$$

$\epsilon_x = O(\alpha \log^2(n/\delta) \cdot \frac{n^{1/4}}{\sqrt{b}})$*, b is the size of sketching matrix.*

Proof. Recall the definition of $\widetilde{\delta}_x$ and δ_x, we have

$$\widetilde{\delta}_{x,i} - \delta_{x,i} = \widetilde{V}_i^{1/2}(I - R^\top R\widetilde{P})\widetilde{V}^{1/2}h - \widetilde{V}_i^{1/2}(I-\widetilde{P})\widetilde{V}^{1/2}h = \widetilde{V}_i^{1/2}(\widetilde{P} - R^\top R\widetilde{P})\widetilde{V}^{1/2}h$$

For iteration t, the definition should be

$$\widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)} = (\widetilde{V}_i^{(t)})^{1/2}(\widetilde{P}^{(t)} - (R^{(t)})^\top R^{(t)}\widetilde{P}^{(t)})(\widetilde{V}^{(t)})^{1/2}h.$$

For any i, let k be the current iteration, k_i be the last when we changed the $\widetilde{V}_i$. Then, we have that

$$x_i^{(k)} - \overline{x}_i^{(k)} = \sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)}$$

because we have $x_i^{(k_i)} = \overline{x}_i^{(k_i)}$ (guaranteed by our algorithm). Since $\widetilde{V}_i^{(t)}$ did not change during iteration k_i to k for the block i. (However, the whole other parts of matrix $\widetilde{V}$ could change). We consider

$$\begin{aligned}(x_i^{(k)} - \overline{x}_i^{(k)})^\top \cdot (\widetilde{V}_i^{(k)})^{-1} \cdot (x_i^{(k)} - \overline{x}_i^{(k)}) &= \left(\sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)}\right)^\top \cdot (\widetilde{V}_i^{(k)})^{-1} \cdot \left(\sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)}\right) \\ &= \left\|\sum_{t=k_i}^{k} \left((I - (R^{(t)})^\top R^{(t)})\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)}\right)_i\right\|_2^2.\end{aligned}$$

We consider block i and a coordinate $j \in$ block i. We define random vector $X_t \in \mathbb{R}^{n_i}$ as follows:

$$X_t = \left((I - R^{(t)\top}R^{(t)})\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)}\right)_i.$$

Let $(X_t)_j$ denote the j-th coordinate of X_t, for each $j \in [n_i]$.

By Lemma 3.2.5 in Section 3.2, we have for each t,

$$\mathbf{E}[X_t] = 0, \quad \text{and} \quad \mathbf{E}[(X_t)_j^2] = \frac{1}{b}\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2^2$$

and with probability $1-\delta$,

$$|(X_t)_j| \le \|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 \frac{\log(n/\delta)}{\sqrt{b}} := M.$$

Now, we apply Bernstein inequality (Lemma 3.2.3),

$$\Pr\left[\sum_t (X_t)_j > \tau\right] \le \exp\left(-\frac{\tau^2/2}{\sum_t \mathbf{E}[(X_t)_j^2] + M\tau/3}\right)$$

Choosing $\tau = 10^3 \frac{\sqrt{T}}{\sqrt{b}} \log^2(n/\delta) \cdot \|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2$

$$\begin{aligned}
&\Pr\left[\sum_t (X_t)_j > 10^3 \frac{\sqrt{T}}{\sqrt{b}} \log^2(n/\delta) \cdot \|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2\right] \\
\le\ &\exp\left(-\frac{10^6 \frac{T}{b}\log^4(n/\delta)\cdot \|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2^2/2}{\frac{T}{b}\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2^2 + 10^3\frac{\sqrt{T}}{b}\log^3(n/\delta)\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2^2/3}\right) \\
\le\ &\exp(-100\log(n/\delta))
\end{aligned}$$

Now, taking a union, we have

$$\begin{aligned}
\left\|\sum_{t=k_i}^{k}\left((I-(R^{(t)})^\top R^{(t)})\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)}\right)_i\right\|_2 &= O\left(\frac{\sqrt{T}}{\sqrt{b}}\log^2(n/\delta)\left\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\right\|_2\right) \\
&\le O\left(\frac{\sqrt{T}}{\sqrt{b}}\log^2(n/\delta)\alpha\right)
\end{aligned}$$

where we use that $\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 \le \|((\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 = O(\alpha)$, $n_i = O(1)$.

Finally, we use the fact that the algorithm reset $\overline{x} = x$, $\overline{s} = s$ in less than $\sqrt{n}$ iterations.

□

s and $\overline{s}$ are close

Lemma 3.1.19 (s and $\overline{s}$ are close)**.** *With probability $1-\delta$ over the randomness of sketching matrix $R \in \mathbb{R}^{b\times n}$, we have*

$$t^{-1}\|\overline{s}_i - s_i\|_{\widetilde{V}_i} \le \epsilon_s,$$

$\epsilon_s = O(\alpha\log^2(n/\delta)\cdot\frac{n^{1/4}}{\sqrt{b}}$, *and b is the size of sketching matrix.*

Proof. Recall the definition of $\widetilde{\delta}_s$, δ_s, we have

$$\widetilde{\delta}_{s,i} - \delta_{s,i} = t\widetilde{V}_i^{-1/2}(R^\top R - I)\widetilde{P}\widetilde{V}^{1/2}h$$

The rest of the proof is identical to Lemma 3.1.18 except we use also the fact we make $\overline{s} = s$ whenever our t changed by a constant factor. We omitted the details here. □

Data structure is maintaining (x, s) implicitly over all the iterations

Lemma 3.1.20. *Over all the iterations, $u_1 + Fu_2$ is always maintaining x implicitly, $u_3 + Gu_4$ is always maintaining s implicitly.*

Proof. We only focus on the PARTIALUPDATE. The FULLUPDATE is trivial, we ignore the proof.

For x.

Note that M is not changing. Let's assume that $u_1 + Fu_2 = x$, we want to show that

$$u_1^{\text{new}} + F^{\text{new}} u_2^{\text{new}} = x^{\text{new}}.$$

which is equivalent to prove

$$u_1^{\text{new}} + F^{\text{new}} u_2^{\text{new}} - (u_1 + Fu_2) = \delta_x$$

Let $\Delta u_1 = u_1^{\text{new}} - u_1$ be the change of u_1 over iteration t, then

$$\Delta u_1 = \widetilde{V}^{\text{new}} h + (F - F^{\text{new}}) u_2$$

Let $\Delta u_2 = u_2^{\text{new}} - u_2$ be the change of u_2 over iteration t, then

$$\Delta u_2 = -(\widetilde{V}^{\text{new}})^{1/2} h + \mathbf{1}_{\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S}}^{\top} (\widetilde{V}^{\text{new}})^{1/2} h.$$

By definition of δ_x at iteration t, we have

$$\delta_x = \widetilde{V}^{\text{new}} h - \left(\sqrt{\widetilde{V}^{\text{new}}} M \sqrt{\widetilde{V}^{\text{new}}} h - \sqrt{\widetilde{V}^{\text{new}}} M_{\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} (M_{\widetilde{S}})^{\top} \sqrt{\widetilde{V}^{\text{new}}} h \right).$$

We can compute

$$\begin{aligned}
& u_1^{\text{new}} + F^{\text{new}} u_2^{\text{new}} - (u_1 + Fu_2) \\
= & \Delta u_1 + (F^{\text{new}} u_2^{\text{new}} - Fu_2) \\
= & \widetilde{V}^{\text{new}} h + (F - F^{\text{new}}) u_2 + (F^{\text{new}} u_2^{\text{new}} - Fu_2) \\
= & \widetilde{V}^{\text{new}} h + F^{\text{new}} (u_2^{\text{new}} - u_2) \\
= & \widetilde{V}^{\text{new}} h + F^{\text{new}} \Delta u_2 \\
= & \widetilde{V}^{\text{new}} h - F^{\text{new}} \sqrt{\widetilde{V}^{\text{new}}} h + F^{\text{new}} \mathbf{1}_{\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} (M_{\widetilde{S}})^{\top} \sqrt{\widetilde{V}^{\text{new}}} h \\
= & \delta_x
\end{aligned}$$

where we used $F^{\text{new}} = \sqrt{\widetilde{V}^{\text{new}}} M$ in the last step.

For s.

We have

$$G^{\text{new}} = \frac{1}{\sqrt{\widetilde{V}^{\text{new}}}} M, G = \frac{1}{\sqrt{\widetilde{V}}} M$$

Let $\Delta u_3 = u_3^{\mathrm{new}} - u_3$ be the change of u_3 over iteration t, then

$$\Delta u_3 = (G - G^{\mathrm{new}})u_4$$

Let $\Delta u_4 = u_4^{\mathrm{new}} - u_4$ be the change of u_4 over iteration t, then

$$\Delta u_4 = t \cdot \Delta u_2$$

By definition of δ_s in iteration t,

$$\delta_s = \left(\frac{1}{\sqrt{\widetilde{V}^{\mathrm{new}}}} M \sqrt{\widetilde{V}^{\mathrm{new}}}(th) - \frac{1}{\sqrt{\widetilde{V}^{\mathrm{new}}}} M_{\widetilde{S}} (\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} (M_{\widetilde{S}})^\top \sqrt{\widetilde{V}^{\mathrm{new}}}(th) \right)$$

We can compute

$$\begin{aligned}
(u_3^{\mathrm{new}} + G^{\mathrm{new}} u_4^{\mathrm{new}}) - (u_3 + G u_4) &= \Delta u_3 + (G^{\mathrm{new}} u_4^{\mathrm{new}} - G u_4) \\
&= (G - G^{\mathrm{new}}) u_4 + (G^{\mathrm{new}} u_4^{\mathrm{new}} - G u_4) \\
&= G^{\mathrm{new}} (u_4^{\mathrm{new}} - u_4) \\
&= G^{\mathrm{new}} t \Delta u_2 \\
&= \delta_s
\end{aligned}$$

where the last step follows by definition of Δu_2.

$\square$

3.2 Basic Properties of Subsampled Hadamard Transform Matrix

This section provides some standard calculations about sketching matrices, it can be found in previous literatures [91, 74]. Usually, the reason for using subsampled randomized Hadamard/Fourier transform [58] is multiplying the matrix with k vectors only takes $kn \log n$ time. Unfortunately, in our application, the best way to optimize the running is using matrix multiplication directly (without doing any fast Fourier transform, or more fancy Fourier transform [35, 73]). In order to have an easy analysis, we still use subsampled randomized Hadamard/Fourier matrix.

Concentration inequalities

We first state a useful for concentration,

Lemma 3.2.1 (Lemma 1 on page 1325 of [45]). *Let $X \sim \mathcal{X}_k^2$ be a chi-squared distributed random variable with k degrees of freedom. Each one has zero mean and σ^2 variance. Then*

$$\Pr[X - k\sigma^2 \geq (2\sqrt{kt} + 2t)\sigma^2] \leq \exp(-t)$$
$$\Pr[k\sigma^2 - X \geq 2\sqrt{kt}\sigma^2] \leq \exp(-t)$$

Lemma 3.2.2 (Khintchine's Inequality). *Let $\sigma_1, \cdots, \sigma_n$ be i.i.d. sign random variables, and let $z_1, \cdots, z_n$ be real numbers. Then there are constants $C, C' > 0$ so that*

$$\Pr\left[\left|\sum_{i=1}^{n} z_i\sigma_i\right| \geq Ct\|z\|_2\right] \leq \exp(-C't^2)$$

Lemma 3.2.3 (Bernstein Inequality). *Let $X_1, \cdots, X_n$ be independent zero-mean random variables. Suppose that $|X_i| \leq M$ almost surely, for all i. Then, for all positive t,*

$$\Pr\left[\sum_{i=1}^{n} X_i > t\right] \leq \exp\left(-\frac{t^2/2}{\sum_{j=1}^{n} \mathbf{E}[X_j^2] + Mt/3}\right)$$

Properties obtained by random projection

Remark 3.2.4. *The Subsampled Randomized Hadamard Transform [58] can be defined as $R = SH_n\Sigma \in \mathbb{R}^{b\times}$, where Σ is an $n \times n$ diagonal matrix with i.i.d. diagonal entries $\Sigma_{i,i}$ in which $\Sigma_{i,i} = 1$ with probability $1/2$, and $\Sigma_{i,i} = -1$ with probability $1/2$. H_n refers to the Hadamard matrix of size n, which we assume is a power of 2. The $b \times n$ matrix S samples b coordinates of n dimensional vector uniformly at random. If we replace the definition of sketching matrix in Lemma 3.2.5 by Subsampled Randomized Hadamard Transform and let $\overline{R} = SH_n$, then the same proof will go through.*

Lemma 3.2.5 (Expectation, variance, absolute guarantees for sketching a fixed vector)**.** *Let $h \in \mathbb{R}^n$ be a fixed vector. Let $\overline{R} \in \mathbb{R}^{b \times n}$ denote a random matrix where each entry is i.i.d. sampled from $+1/\sqrt{b}$ with probability $1/2$ and $-1/\sqrt{b}$ with probability $1/2$. Let $\Sigma \in \mathbb{R}^{n \times n}$ denote a diagonal matrix where each entry is 1 with probability $1/2$ and -1 with probability $1/2$. Let $R = \overline{R}\Sigma$, then we have*

$$\mathbf{E}[R^\top R h] = h, \quad \mathbf{E}[(R^\top R h)_i^2] \leq h_i^2 + \frac{1}{b}\|h\|_2^2, \quad \Pr\left[|(R^\top R h)_i - h_i| > \|h\|_2 \frac{\log(n/\delta)}{\sqrt{b}} \right] \leq \delta.$$

Proof. Let $R_{i,j}$ denote the entry at i-th row and j-th column in matrix $R \in \mathbb{R}^{b \times n}$. Let $R_{*,i} \in \mathbb{R}^b$ denote the vector in i-th column of R.

We first show expectation,

$$\begin{aligned}
\mathbf{E}[(R^\top R h)_i] = {} & \mathbf{E}[\langle R, R_{*,i} h^\top \rangle] \\
= {} & \mathbf{E}\left[\sum_{j=1}^{b} \sum_{l=1}^{n} R_{j,l} R_{j,i} h_l \right] \\
= {} & \mathbf{E}\left[\sum_{j=1}^{b} R_{j,i}^2 h_i \right] + \mathbf{E}\left[\sum_{j=1}^{b} \sum_{l \in [n] \backslash i} R_{j,l} R_{j,i} h_l \right] \\
= {} & h_i + 0 \\
= {} & h_i
\end{aligned}$$

Secondly, we prove the variance is small

$$\begin{aligned}
\mathbf{E}[(R^\top R h_i)^2] = {} & \mathbf{E}[\langle R, R_{*,i} h^\top \rangle^2] \\
= {} & \mathbf{E}\left[\left(\sum_{j=1}^{b} \sum_{l=1}^{n} R_{j,l} R_{j,i} h_l \right)^2 \right] \\
= {} & \mathbf{E}\left[\left(\sum_{j=1}^{b} R_{j,i}^2 h_i + \sum_{j=1}^{b} \sum_{l \in [n] \backslash i} R_{j,l} R_{j,i} h_l \right)^2 \right] \\
= {} & \mathbf{E}\left[\left(\sum_{j=1}^{b} R_{j,i}^2 h_i \right)^2 \right] + 2\,\mathbf{E}\left[\sum_{j'=1}^{b} R_{j',i}^2 h_i \sum_{j=1}^{b} \sum_{l \in [n] \backslash i} R_{j,l} R_{j,i} h_l \right] \\
& + \mathbf{E}\left[\left(\sum_{j=1}^{b} \sum_{l \in [n] \backslash i} R_{j,l} R_{j,i} h_l \right)^2 \right] \\
= {} & C_1 + C_2 + C_3,
\end{aligned}$$

where the last step follows from defining those terms to be C_1, C_2 and C_3. For the term C_1, we have

$$C_1 = h_i^2 \mathbf{E}\left[\left(\sum_{j=1}^b R_{j,i}^2\right)^2\right] = h_i^2 \mathbf{E}\left[\sum_{j=1}^b R_{j,i}^4 + \sum_{j'\neq j} R_{j,i}^2 R_{j',i}^2\right] = h_i^2\left(b\cdot\frac{1}{b^2} + b(b-1)\cdot\frac{1}{b^2}\right) = h_i^2$$

For the second term C_2,

$$C_2 = 0.$$

For the third term C_3,

$$\begin{aligned} C_3 = & \mathbf{E}\left[\left(\sum_{j=1}^b \sum_{l\in[n]\backslash i} R_{j,l}R_{j,i}h_l\right)^2\right] \\ = & \mathbf{E}\left[\sum_{j=1}^b \sum_{l\in[n]\backslash i} R_{j,l}^2 R_{j,i}^2 h_l^2\right] + \mathbf{E}\left[\sum_{j=1}^b \sum_{l\in[n]\backslash i} R_{j,l}R_{j,i}h_l \sum_{j'\in[b]\backslash j}\sum_{l'\in[n]\backslash i\backslash l} R_{j',l'}R_{j',i}h_{l'}\right] \\ = & \sum_{j=1}^b \sum_{l\in[n]\backslash i} \frac{1}{d}\frac{1}{d}h_l^2 + 0 \leq \frac{1}{d}\|h\|_2^2 \end{aligned}$$

Therefore, we have

$$\mathbf{E}[(R^\top Rh)_i^2] \leq C_1 + C_2 + C_3 \leq h_i^2 + \frac{1}{b}\|h\|_2^2.$$

Third, we prove the worst case bound with high probability. We can write $(R^\top Rh)_i - h_i$ as follows

$$\begin{aligned} (R^\top Rh)_i - h_i &= \langle R, R_{*,i}h^\top\rangle - h_i \\ &= \sum_{j=1}^b \sum_{l=1}^n R_{j,l}\cdot R_{j,i}\cdot h_l - h_i \\ &= \sum_{j=1}^b R_{j,i}^2 h_i - h_i + \sum_{j=1}^b \sum_{l\in[n]\backslash i} R_{j,l}R_{j,i}\cdot h_l \\ &= \sum_{j=1}^b \sum_{l\in[n]\backslash i} R_{j,l}R_{j,i}\cdot h_l && \text{by } R_{j,i}^2 = 1/b \\ &= \sum_{l\in[n]\backslash i} h_l\langle R_{*,l}, R_{*,i}\rangle \\ &= \sum_{l\in[n]\backslash i} h_l\cdot\langle \sigma_l\overline{R}_{*,l}, \sigma_i\overline{R}_{*,i}\rangle && \text{by } R_{*,l} = \sigma_l\overline{R}_{*,l} \end{aligned}$$

First, we apply Khintchine's inequality, we have

$$\Pr\left[\left|\sum_{l\in[n]\backslash i} h_l\cdot\sigma_l\cdot\langle\overline{R}_{*,l},\sigma_i\overline{R}_{*,i}\rangle\right|\geq Ct\left(\sum_{l\in[n]\backslash i} h_l^2(\langle\overline{R}_{*,l},\sigma_i\overline{R}_{*,i}\rangle)^2\right)^{1/2}\right]\leq\exp(-C't^2)$$

and choose $t=\sqrt{\log(n/\delta)}$.

For each $l\neq i$, using [58] we have

$$\Pr\left[|\langle\overline{R}_{*,l},\overline{R}_{*,i}\rangle|\geq\frac{\sqrt{\log(n/\delta)}}{\sqrt{b}}\right]\leq\delta/n.$$

Taking a union bound over all $l\in[n]\backslash i$, we have

$$|(R^\top Rh)_i-h_i|\leq\|h\|_2\frac{\log(n/\delta)}{\sqrt{b}}$$

with probability $1-\delta$. $\square$

3.3 Combining Robust Central Path with Data Structure

The goal of this section is to combine Section 2.5 and Section 3.1.

Notation	Choice of Parameter	Statement	Comment
C_1	$\Theta(1/\log^2 n)$	Lem. 3.3.1, Thm. 3.1.1	ℓ_2 accuracy of W sequence
C_2	$\Theta(1/\log^4 n)$	Lem. 3.3.1, Thm. 3.1.1	ℓ_4 accuracy of W sequence
ϵ_{mp}	$\Theta(1/\log^2 n)$	ROBUSTIPM Alg in Sec. 2.5	accuracy for data structure
T	$\Theta(\sqrt{n}\log^2 n\log(n/\delta))$	Thm. 2.4.2	#iterations
α	$\Theta(1/\log^2 n)$	ROBUSTIPM Alg in Sec. 2.5	step size in Hessian norm
b	$\Theta(\sqrt{n}\log^6(nT)$	Lem. 3.1.18, Lem. 3.1.19, Lem. 3.3.2	sketch size
ϵ_x	$\Theta(1/\log^3 n)$	Lem. 3.1.18	accuracy of $\overline{x}$ (respect to x)
ϵ_s	$\Theta(1/\log^3 n)$	Lem. 3.1.19	accuracy of $\overline{s}$ (respect to s)
ϵ_w	$\Theta(1/\log^3 n)$	Lem. 3.3.2	accuracy of $\overline{W}$ (respect to W)
a	$\min(2/3, \alpha_m)$	α_m is the dual exponent of MM	batch size

Table 3.2: Summary of parameters

Guarantee for W matrices

Lemma 3.3.1 (Guarantee of a sequence of W). *Let $x^{\text{new}} = x + \delta_x$. Let $W^{\text{new}} = (\nabla^2\phi(x^{\text{new}}))^{-1}$ and $W = (\nabla^2\phi(x))^{-1}$. Then we have*

$$\sum_{i=1}^{m} \left\| w_i^{-1/2}(w_i^{\text{new}} - w_i)w_i^{-1/2} \right\|_F^2 \leq C_1^2,$$

$$\sum_{i=1}^{m} \left\| w_i^{-1/2}(w_i^{\text{new}} - w_i)w_i^{-1/2} \right\|_F^4 \leq C_2^2,$$

$$\left\| w_i^{-1/2}(w_i^{\text{new}} - w_i)w_i^{-1/2} \right\|_F \leq \frac{1}{4}.$$

where $C_2 = \Theta(\alpha^2)$ and $C_1 = \Theta(\alpha)$.

Proof. For each $i \in [m]$, we have

$$\begin{aligned}
&\left\| W_i^{-1/2} (W_i^{\mathrm{new}} - W_i) W_i^{-1/2} \right\|_F^2 \\
=\ & n_i \left\| W_i^{-1/2} (W_i^{\mathrm{new}} - W_i) W_i^{-1/2} \right\|^2 \\
=\ & n_i \left\| (\nabla^2 \phi(x_i))^{1/2} (\nabla^2 \phi(x_i^{\mathrm{new}})^{-1} - \nabla^2 \phi(x_i)^{-1}) (\nabla^2 \phi(x_i))^{1/2} \right\|^2 \\
\leq\ & \left(\frac{1}{(1 - \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2 \phi(x_i)})^2} - 1 \right)^2 \cdot \left\| (\nabla^2 \phi(x_i))^{1/2} \nabla^2 \phi(x_i)^{-1} (\nabla^2 \phi(x_i))^{1/2} \right\|^2 \\
=\ & n_i \left(\frac{1}{(1 - \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2 \phi(x_i)})^2} - 1 \right)^2 \\
\leq\ & 100 n_i \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2 \phi(x_i)}^2,
\end{aligned}$$

where the second step follows by Theorem 1.2.3.

In our problem, we assume that $n_i = O(1)$. It remains to bound

$$\|x_i^{\mathrm{new}} - x_i\|_{\nabla^2 \phi(x_i)}^2 = \|\delta_{x,i}\|_{\nabla^2 \phi(x_i)}^2 \lesssim \|\delta_{x,i}\|_{\overline{x}_i}^2 = \alpha_i^2$$

where the last step follows from definition $\alpha_i = \|\delta_{x,i}\|_{\overline{x}_i}$.

Then, we have

$$\sum_{i=1}^m \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2 \phi(x_i)}^2 \leq \sum_{i=1}^m O(\alpha_i^2) \leq O(\alpha^2).$$

where the last step follows by Lemma 2.5.1. □

Lemma 3.3.2 (Accuracy of $\overline{W}$)**.** *Let x and $\overline{x}$ be the vectors maintained by data-structure* StochasticProjectionMaintenance*. Let $W = (\nabla^2 \phi(x))^{-1}$ and $\overline{W} = (\nabla^2 \phi(\overline{x}))^{-1}$. Then we have*

$$\|w_i^{-1/2} (\overline{w}_i - w_i) w_i^{-1/2}\|_F \leq \epsilon_w,$$

where $\epsilon_w = O\Big(\alpha \log^2(nT) \cdot \frac{n^{1/4}}{\sqrt{b}}\Big)$, b is the size of sketching matrix.

Proof. By similar calculation, we have

$$\|w_i^{-1/2} (\overline{w}_i - w_i) w_i^{-1/2}\|_F = O(1) \cdot \|\overline{x}_i - x_i\|_{\nabla^2 \phi(x_i)}.$$

Then, using Lemma 3.1.18 with $\delta = 1/T$

$$\|\overline{x}_i - x_i\|_{\nabla^2 \phi(x_i)} \leq O\left(\alpha \log^2(nT) \cdot \frac{\sqrt{n^{1/4}}}{\sqrt{b}} \right).$$

□

Algorithm 5 Robust Central Path

```
1:  procedure CentralPathStep($\overline{x}, \overline{s}, t, \lambda, \alpha$)
2:      for $i = 1 \to m$ do                                              ▷ Figure out direction $h$
3:          $\mu_i^t \leftarrow \overline{s}_i/t + \nabla\phi_i(\overline{x}_i)$                          ▷ According to Eq. (2.5)
4:          $\gamma_i^t \leftarrow \|\mu_i^t\|_{\nabla^2\phi_i(\overline{x}_i)^{-1}}$                          ▷ According to Eq. (2.6)
5:          $c_i^t \leftarrow \frac{\exp(\lambda\gamma_i^t)/\gamma_i^t}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t))^{1/2}}$ if $\gamma_i^t \geq 96\sqrt{\alpha}$ and $c_i^t \leftarrow 0$ otherwise   ▷ According to Eq. (2.8)
6:          $h_i \leftarrow -\alpha \cdot c_i^t \cdot \mu_i^t$                                  ▷ According to Eq. (2.7)
7:      end for
8:      $\overline{W} \leftarrow (\nabla^2\phi(\overline{x}))^{-1}$                    ▷ Computing block-diagonal matrix $\overline{W}$
9:      return $h, \overline{W}$
10: end procedure
11:
12: procedure RobustCentralPath(mp, $t, \lambda, \alpha$)                             ▷ Lemma 2.5.8
13:                                         ▷ Standing at $(x, s)$ implicitly via data-structure
14:                                         ▷ Standing at $(\overline{x}, \overline{s})$ explicitly via data-structure
15:     $(\overline{x}, \overline{s}) \leftarrow$ mp.Query()                               ▷ Algorithm 1, Lemma 3.1.8
16:
17:     $h, \overline{W} \leftarrow$ CentralPathStep($\overline{x}, \overline{s}, t, \lambda, \alpha$)
18:
19:     mp.Update($\overline{W}$)                                        ▷ Algorithm 2, Lemma 3.1.5
20:     mp.MultiplyMove($h, t$)                      ▷ Algorithm 4, Lemma 3.1.10, Lemma 3.1.9
21:                  ▷ $x \leftarrow x + \delta_x$, $s \leftarrow s + \delta_s$, achieved by data-structure implicitly
22:                  ▷ $\overline{x} \leftarrow \overline{x} + \widetilde{\delta}_x$, $\overline{s} \leftarrow \overline{s} + \widetilde{\delta}_s$, achieved by data-structure explicitly
23:                                              ▷ If $x$ is far from $\overline{x}$, then $\overline{x} \leftarrow x$
24: end procedure
```

Main result

Theorem 3.3.3 (Main result, formal version of Theorem 2.1.1). *Consider a convex problem*

$$\min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x$$

where K_i are compact convex set. For each i, we are given a ν_i-self concordant barrier function ϕ_i for K_i. Also, we are given $x^{(0)} = \arg\min_x \sum_i \phi_i(x_i)$. Assume that

1. *Diameter of the set: For any $x \in \prod K_i$, we have that $\|x\|_2 \leq R$.*
2. *Lipschitz constant of the program: $\|c\|_2 \leq L$.*

Algorithm 6 Our main algorithm (More detailed version of RobustIPM in Section 2.4)

```
1: procedure Main($A, b, c, \phi, \delta$)                                  ▷ Theorem 2.1.1, Theorem 3.3.3
2:     $\lambda \leftarrow 2^{16}\log(m)$, $\alpha \leftarrow 2^{-20}\lambda^{-2}$ , $\kappa \leftarrow 2^{-10}\alpha$
3:     $\delta \leftarrow \min(\frac{1}{\lambda}, \delta)$                                   ▷ Choose the target accuracy
4:     $a \leftarrow \min(2/3, \alpha_m)$                                        ▷ Choose the batch size
5:     $b_{\text{sketch}} \leftarrow 2^{10}\sqrt{\nu}\log^6(n/\delta) \cdot \log\log(1/\delta)$      ▷ Choose the size of sketching matrix
6:     Modify the ERM($A, b, c, \phi$) and obtain an initial $x$ and $s$
7:     CentralPathMaintenance mp                              ▷ Algorithm 1, Theorem 3.1.1
8:     mp.Initialize($A, x, s, \alpha, a, b_{\text{sketch}}$)                     ▷ Algorithm 1, Lemma 3.1.4
9:     $\nu \leftarrow \sum_{i=1}^{m} \nu_i$                  ▷ $\nu_i$ are the self-concordant parameters of $\phi_i$
10:    $t \leftarrow 1$
11:    while $t > \delta^2/(4\nu)$ do
12:        $t^{\text{new}} \leftarrow (1 - \frac{\kappa}{\sqrt{\nu}})t$
13:        RobustCentralPath(mp, $t, \lambda, \alpha$)                            ▷ Algorithm 5
14:        $t \leftarrow t^{\text{new}}$
15:    end while
16:    Return an approximate solution of the original ERM according to Section 3.4
17: end procedure
```

Then, the algorithm Main *finds a vector x such that*

$$c^\top \overline{x}_{1:n} \leq \min_{Ax=b, x\in\prod_i K_i} c^\top x + LR \cdot \delta,$$

$$\|Ax - b\|_1 \leq 3\delta \cdot \left(R \sum_{i,j} |A_{i,j}| + \|b\|_1 \right),$$

$$x \in \prod_i K_i.$$

in time

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

where ω is the exponent of matrix multiplication [90, 46], and α is the dual exponent of matrix multiplication [47].

Proof. The number of iterations is

$$O(\sqrt{\nu}\log^2(m)\log(\nu/\delta)) = O(\sqrt{n}\log^2(n)\log(n/\delta)).$$

For each iteration, the amortized cost per iteration is

$$
\begin{aligned}
& O(nb + n^{1+a} + n^{1.5}) + O(C_1/\epsilon_{mp} + C_2/\epsilon_{mp}^2) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}) + O(n^{\omega-1/2+o(1)}) \\
= & O(nb + n^{1+a} + n^{1.5}) + O(\alpha + \alpha^2) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}) + O(n^{\omega-1/2+o(1)}) \\
= & O(nb + n^{1+a} + n^{1.5}) + O(1/\log^4 n) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}) + O(n^{\omega-1/2+o(1)}) \\
= & O(n^{1.5+o(1)} \log^6 \log(1/\delta) + n^{1+a+o(1)}) + O(n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}).
\end{aligned}
$$

where the last step follows from choice of b (see Table 3.2).

Finally, we have

$$
\begin{aligned}
& \text{total time} \\
= & \#\text{iterations} \cdot \text{cost per iteration} \\
= & \underbrace{O\left(\sqrt{n} \log^2 n \log(n/\delta)\right)}_{\#\text{iterations}} \cdot \underbrace{O\left(n^{1.5+o(1)} \log^6 \log(1/\delta) + n^{1+a+o(1)} + n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}\right)}_{\text{cost per iteration}} \\
= & O\left(n^{1.5+a+o(1)} + n^{\omega+o(1)} + n^{2.5-a/2+o(1)}\right) \cdot \log(n/\delta) \cdot \log^6 \log(1/\delta) \\
= & O\left(n^{2+1/6+o(1)} + n^{\omega+o(1)} + n^{2.5-\alpha_m/2+o(1)}\right) \cdot \log(n/\delta) \cdot \log^6 \log(1/\delta)
\end{aligned}
$$

where we pick $a = \min(2/3, \alpha_m)$ and α_m is the dual exponent of matrix multiplication[47].

Thus, we complete the proof. $\square$

3.4 Initial Point and Termination Condition

We first need some result about self concordance.

Lemma 3.4.1 (Theorem 4.1.7, Lemma 4.2.4 in [68]). *Let ϕ be any ν-self-concordant barrier. Then, for any $x, y \in \mathrm{dom}\phi$, we have*

$$\langle \nabla\phi(x), y - x\rangle \le \nu,$$
$$\langle \nabla\phi(y) - \nabla\phi(x), y - x\rangle \ge \frac{\|y-x\|_x^2}{1+\|y-x\|_x}.$$

Let $x^* = \arg\min_x \phi(x)$. For any $x \in \mathbb{R}^n$ such that $\|x - x^*\|_{x^*} \le 1$, we have that $x \in \mathrm{dom}\phi$.

$$\|x^* - y\|_{x^*} \le \nu + 2\sqrt{\nu}.$$

Lemma 3.4.2. *Consider a convex problem $\min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x$ where K_i are compact convex set. For each i, we are given a ν_i-self concordant barrier function ϕ_i for K_i. Also, we are given $x^{(0)} = \arg\min_x \sum_i \phi_i(x_i)$. Assume that*

1. *Diameter of the set: For any $x \in \prod K_i$, we have that $\|x\|_2 \le R$.*
2. *Lipschitz constant of the program: $\|c\|_2 \le L$.*

For any $\delta > 0$, the modified program $\min_{\overline{A}\overline{x}=\overline{b}, \overline{x}\in\prod_{i=1}^m K_i\times\mathbb{R}_+} \overline{c}^\top \overline{x}$ with

$$\overline{A} = [A \mid b - Ax^{(0)}], \overline{b} = b, \text{ and } \overline{c} = \begin{bmatrix} \frac{\delta}{LR}\cdot c \\ 1 \end{bmatrix}$$

satisfies the following:

1. *$\overline{x} = \begin{bmatrix} x^{(0)} \\ 1 \end{bmatrix}$, $\overline{y} = 0_d$ and $\overline{s} = \begin{bmatrix} \frac{\delta}{LR}\cdot c \\ 1 \end{bmatrix}$ are feasible primal dual vectors with $\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|_{\overline{x}}^* \le \delta$ where $\overline{\phi}(\overline{x}) = \sum_{i=1}^m \phi_i(\overline{x}_i) - \log(\overline{x}_{m+1})$.*
2. *For any $\overline{x}$ such that $\overline{A}\overline{x} = \overline{b}, \overline{x} \in \prod K_i \times \mathbb{R}_+$ and $\overline{c}^\top\overline{x} \le \min_{\overline{A}\overline{x}=\overline{b},\overline{x}\in\prod K_i\times\mathbb{R}_+} \overline{c}^\top\overline{x} + \delta^2$, the vector $\overline{x}_{1:n}$ ($\overline{x}_{1:n}$ is the first n coordinates of $\overline{x}$) is an approximate solution to the original convex program in the following sense*

$$c^\top\overline{x}_{1:n} \le \min_{Ax=b, x\in\prod K_i} c^\top x + LR\cdot\delta,$$
$$\|A\overline{x}_{1:n} - b\|_1 \le 3\delta\cdot\left(R\sum_{i,j}|A_{i,j}| + \|b\|_1\right),$$
$$\overline{x}_{1:n} \in \prod K_i.$$

Proof. For the first result, straightforward calculations show that $(\overline{x}, \overline{y}, \overline{s})$ are feasible.

To compute $\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|_{\overline{x}}^*$, note that

$$\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|_{\overline{x}}^* = \|\frac{\delta}{LR}\cdot c\|_{\nabla^2\phi(x^{(0)})^{-1}}.$$

Lemma 3.4.1 shows that $x \in \mathbb{R}^n$ such that $\|x - x^{(0)}\|_{x^{(0)}} \leq 1$, we have that $x \in \prod K_i$ because $x^{(0)} = \arg\min_x \sum_i \phi_i(x_i)$. Hence, for any v such that $v^\top \nabla^2 \phi(x^{(0)}) v \leq 1$, we have that $x^{(0)} \pm v \in \prod K_i$ and hence $\|x^{(0)} \pm v\|_2 \leq R$. This implies $\|v\|_2 \leq R$ for any $v^\top \nabla^2 \phi(x^{(0)}) v \leq 1$. Hence, $(\nabla^2 \phi(x^{(0)}))^{-1} \preceq R^2 \cdot I$. Hence, we have

$$\|\overline{s} + \nabla \overline{\phi}(\overline{x})\|_{\overline{x}}^* = \|\frac{\delta}{LR} \cdot c\|_{\nabla^2 \phi(x^{(0)})^{-1}} \leq \|\frac{\delta}{L} \cdot c\|_2 \leq \delta.$$

For the second result, we let $\mathrm{OPT} = \min_{Ax=b, x \in \prod K_i} c^\top x$ and $\overline{\mathrm{OPT}} = \min_{\overline{A}\overline{x}=b, \overline{x} \in \prod K_i \times \mathbb{R}_+} \overline{c}^\top \overline{x}$. For any feasible x in the original problem, $\overline{x} = \begin{bmatrix} x \\ 0 \end{bmatrix}$ is a feasible in the modified problem. Therefore, we have that

$$\overline{\mathrm{OPT}} \leq \frac{\delta}{LR} \cdot c^\top x = \frac{\delta}{LR} \cdot \mathrm{OPT}.$$

Given a feasible $\overline{x}$ with additive error δ^2. Write $\overline{x} = \begin{bmatrix} \overline{x}_{1:n} \\ \tau \end{bmatrix}$ for some $\tau \geq 0$. We can compute $\overline{c}^\top \overline{x}$ which is $\frac{\delta}{LR} \cdot c^\top \overline{x}_{1:n} + \tau$. Then, we have

$$\frac{\delta}{LR} \cdot c^\top \overline{x}_{1:n} + \tau \leq \overline{\mathrm{OPT}} + \delta^2 \leq \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2. \tag{3.6}$$

Hence, we can upper bound the OPT of the transformed program as follows:

$$c^\top \overline{x}_{1:n} = \frac{LR}{\delta} \cdot \frac{\delta}{LR} c^\top \overline{x}_{1:n} \leq \frac{LR}{\delta} \left(\frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2 \right) = \mathrm{OPT} + LR \cdot \delta,$$

where the second step follows by (3.6).

For the feasibility, we have that $\tau \leq -\frac{\delta}{LR} \cdot c^\top \overline{x}_{1:n} + \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2 \leq \delta + \delta + \delta$ because $\mathrm{OPT} = \min_{Ax=b, x \geq 0} c^\top x \leq LR$ and that $c^\top \overline{x}_{1:n} \leq LR$. The constraint in the new polytope shows that

$$A\overline{x}_{1:n} + (b - Ax^{(0)})\tau = b.$$

Rewriting it, we have $A\overline{x}_{1:n} - b = (Ax^{(0)} - b)\tau$ and hence

$$\|A\overline{x}_{1:n} - b\|_1 \leq \|Ax^{(0)} - b\|_1 \cdot \tau.$$

□

Lemma 3.4.3. *Let $\phi_i(x_i)$ be a ν_i-self-concordant barrier. Suppose we have $\frac{s_i}{t} + \nabla \phi_i(x_i) = \mu_i$ for all $i \in [m]$, $A^\top y + s = c$ and $Ax = b$. Suppose that $\|\mu_i\|_{x,i}^* \leq 1$ for all i, we have that*

$$\langle c, x \rangle \leq \langle c, x^* \rangle + 4t\nu$$

where $x^* = \arg\min_{Ax=b, x \in \prod_i K_i} c^\top x$ and $\nu = \sum_{i=1}^m \nu_i$.

Proof. Let $x_\alpha = (1-\alpha)x + \alpha x^*$ for some α to be chosen. By Lemma 3.4.1, we have that $\langle \nabla\phi(x_\alpha), x^* - x_\alpha\rangle \leq \nu$. Hence, we have $\frac{\nu}{1-\alpha} \geq \langle \nabla\phi(x_\alpha), x^* - x\rangle$. Hence, we have

$$\begin{aligned}
\frac{\nu\alpha}{1-\alpha} &\geq \langle \nabla\phi(x_\alpha), x_\alpha - x\rangle \\
&= \langle \nabla\phi(x_\alpha) - \nabla\phi(x), x_\alpha - x\rangle + \left\langle \mu - \frac{s}{t}, x_\alpha - x\right\rangle \\
&\geq \sum_{i=1}^{m} \frac{\|x_{\alpha,i} - x_i\|_{x_i}^2}{1 + \|x_{\alpha,i} - x_i\|_{x_i}} + \langle \mu, x_\alpha - x\rangle - \frac{1}{t}\left\langle c - A^\top y, x_\alpha - x\right\rangle \\
&\geq \sum_{i=1}^{m} \frac{\alpha^2\|x_i^* - x_i\|_{x_i}^2}{1 + \alpha\|x_i^* - x_i\|_{x_i}} - \alpha\sum_{i=1}^{m} \|\mu_i\|_{x_i}^* \|x_i^* - x_i\|_{x_i} - \frac{\alpha}{t}\left\langle c, x^* - x\right\rangle.
\end{aligned}$$

where we used Lemma 3.4.1 on the second first, $Ax_\alpha = Ax$ on the second inequality. Hence, we have

$$\frac{\langle c, x\rangle}{t} \leq \frac{\langle c, x^*\rangle}{t} + \frac{\nu}{1-\alpha} + \sum_{i=1}^{m} \|\mu_i\|_{x_i}^* \|x_i^* - x_i\|_{x_i} - \sum_{i=1}^{m} \frac{\alpha\|x_i^* - x_i\|_{x_i}^2}{1 + \alpha\|x_i^* - x_i\|_{x_i}}.$$

Using $\|\mu_i\|_{x_i}^* \leq 1$ for all i, we have

$$\frac{\langle c, x\rangle}{t} \leq \frac{\langle c, x^*\rangle}{t} + \frac{\nu}{1-\alpha} + \sum_{i=1}^{m} \frac{\|x_i^* - x_i\|_{x_i}}{1 + \alpha\|x_i^* - x_i\|_{x_i}} \leq \frac{\langle c, x^*\rangle}{t} + \frac{\nu}{1-\alpha} + \frac{m}{\alpha}.$$

Setting $\alpha = \frac{1}{2}$, we have $\langle c, x\rangle \leq \langle c, x^*\rangle + 2t(\nu + m) \leq \langle c, x^*\rangle + 4t\nu$ because the self-concordance ν_i is always larger than 1.

$\square$

Bibliography

[1] Deeksha Adil et al. “Iterative Refinement for ℓ_p-norm Regression”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2019, pp. 1405–1424.

[2] Naman Agarwal et al. “Leverage Score Sampling for Faster Accelerated Regression and ERM”. In: *arXiv preprint arXiv:1711.08426* (2017).

[3] Zeyuan Allen-Zhu. “Katyusha: The first direct acceleration of stochastic gradient methods”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM. 2017, pp. 1200–1205.

[4] Zeyuan Allen-Zhu. “Katyusha X: Practical Momentum Method for Stochastic Sum-of-Nonconvex Optimization”. In: *Proceedings of the 35th International Conference on Machine Learning*. ICML ’18. Full version available at http://arxiv.org/abs/1802.03866. 2018.

[5] Zeyuan Allen-Zhu. “Natasha 2: Faster Non-Convex Optimization Than SGD”. In: *Proceedings of the 32nd Conference on Neural Information Processing Systems*. NIPS ’18. Full version available at http://arxiv.org/abs/1708.08694. 2018.

[6] Zeyuan Allen-Zhu. “Natasha: Faster Non-Convex Stochastic Optimization via Strongly Non-Convex Parameter”. In: *Proceedings of the 34th International Conference on Machine Learning*. ICML ’17. Full version available at http://arxiv.org/abs/1702.00763. 2017.

[7] Zeyuan Allen-Zhu and Elad Hazan. “Variance Reduction for Faster Non-Convex Optimization”. In: *Proceedings of the 33rd International Conference on Machine Learning*. ICML ’16. Full version available at http://arxiv.org/abs/1603.05643. 2016.

[8] Zeyuan Allen-Zhu and Yang Yuan. “Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives”. In: *Proceedings of the 33rd International Conference on Machine Learning*. ICML ’16. Full version available at http://arxiv.org/abs/1506.01972. 2016.

[9] Josh Alman. “Limits on the Universal Method for Matrix Multiplication”. In: *arXiv preprint arXiv:1812.08731* (2018).

[10] Josh Alman and Virginia Vassilevska Williams. “Further limitations of the known approaches for matrix multiplication”. In: *ITCS*. arXiv preprint arXiv:1712.07246, 2018.

[11] Josh Alman and Virginia Vassilevska Williams. “Limits on All Known (and Some Unknown) Approaches to Matrix Multiplication”. In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018.

[12] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson. “Local rademacher complexities”. In: *The Annals of Statistics* 33.4 (2005), pp. 1497–1537.

[13] Léon Bottou and Olivier Bousquet. “The tradeoffs of large scale learning”. In: *Advances in neural information processing systems*. 2008, pp. 161–168.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[15] Sébastien Bubeck et al. “An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. ACM. 2018, pp. 1130–1137.

[16] Kenneth L Clarkson. “Subgradient and sampling algorithms for ℓ_1 regression”. In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*. 2005, pp. 257–266.

[17] Kenneth L. Clarkson and David P. Woodruff. “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*. https://arxiv.org/pdf/1207.6365, 2013, pp. 81–90.

[18] Michael B Cohen, Yin Tat Lee, and Zhao Song. “Solving Linear Programs in the Current Matrix Multiplication Time”. In: *arXiv preprint arXiv:1810.07896* (2018).

[19] Michael B Cohen et al. “Geometric median in nearly linear time”. In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*. ACM. 2016, pp. 9–21.

[20] Don Coppersmith and Shmuel Winograd. “Matrix multiplication via arithmetic progressions”. In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing(STOC)*. ACM. 1987, pp. 1–6.

[21] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[22] David R Cox. “The regression analysis of binary sequences”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 215–242.

[23] Dominik Csiba. “Data Sampling Strategies in Stochastic Algorithms for Empirical Risk Minimization”. In: *arXiv preprint arXiv:1804.00437* (2018).

[24] Daniel Dadush and Sophie Huiberts. “A friendly smoothed analysis of the simplex method”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM. 2018, pp. 390–403.

[25] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. "A sparse johnson: Lindenstrauss transform". In: *Proceedings of the forty-second ACM symposium on Theory of computing (STOC)*. ACM. 2010, pp. 341–350.

[26] Anirban Dasgupta et al. "Sampling algorithms and coresets for ℓ_p regression". In: *SIAM Journal on Computing* 38.5 (2009), pp. 2060–2078.

[27] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives". In: *Advances in neural information processing systems*. 2014, pp. 1646–1654.

[28] Alexandre Défossez and Francis Bach. "Constant step size least-mean-square: Bias-variance trade-offs and optimal sampling distributions". In: *arXiv preprint arXiv:1412.0156* (2014).

[29] Aymeric Dieuleveut and Francis Bach. "Nonparametric stochastic approximation with large step-sizes". In: *The Annals of Statistics* 44.4 (2016), pp. 1363–1399.

[30] Vitaly Feldman et al. "Agnostic learning of monomials by halfspaces is hard". In: *SIAM Journal on Computing* 41.6 (2012), pp. 1558–1590.

[31] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[32] Roy Frostig et al. "Competing with the empirical risk minimizer in a single pass". In: *Conference on learning theory (COLT)*. 2015, pp. 728–763.

[33] Anna C Gilbert et al. "One sketch for all: fast algorithms for compressed sensing". In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM. 2007, pp. 237–246.

[34] Alon Gonen and Shai Shalev-Shwartz. "Fast rates for empirical risk minimization of strict saddle problems". In: *arXiv preprint arXiv:1701.04271* (2017).

[35] Haitham Hassanieh et al. "Nearly optimal sparse fourier transform". In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM. 2012, pp. 563–578.

[36] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. Vol. 398. John Wiley & Sons, 2013.

[37] Chi Jin et al. "On the local minima of the empirical risk". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018, pp. 4901–4910.

[38] Rie Johnson and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in neural information processing systems*. 2013, pp. 315–323.

[39] Daniel M Kane and Jelani Nelson. "Sparser johnson-lindenstrauss transforms". In: *Journal of the ACM (JACM)*. Vol. 61(1). `https://arxiv.org/pdf/1012.1577`, 2014, p. 4.

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

[41] Victor Klee and George J Minty. *How good is the simplex algorithm*. Tech. rep. WASHINGTON UNIV SEATTLE DEPT OF MATHEMATICS, 1970.

[42] Roger Koenker. “Galton, Edgeworth, Frisch, and prospects for quantile regression in econometrics”. In: *Journal of Econometrics* 95.2 (2000), pp. 347–374.

[43] Roger Koenker. *Quantile Regression*. Cambridge University Press, 2005.

[44] Roger Koenker and Kevin F Hallock. “Quantile regression”. In: *Journal of economic perspectives* 15.4 (2001), pp. 143–156.

[45] Beatrice Laurent and Pascal Massart. “Adaptive estimation of a quadratic functional by model selection”. In: *Annals of Statistics* (2000), pp. 1302–1338.

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

[47] Francois Le Gall and Florent Urrutia. “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)*. SIAM. 2018, pp. 1029–1046.

[48] Nicolas Le Roux, Mark W Schmidt, and Francis R Bach. “A Stochastic Gradient Method with an Exponential Convergence Rate for Finite Training Sets.” In: *NIPS*. 2012, pp. 2672–2680.

[49] Yin Tat Lee. “Uniform Sampling and Inverse Maintenance”. In: *Talk at Michael Cohen Memorial Symposium*. Available at: https://simons.berkeley.edu/talks/welcome-andbirds-eye-view-michaels-work., 2017.

[50] Yin Tat Lee and Aaron Sidford. “Efficient inverse maintenance and faster algorithms for linear programming”. In: *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE. 2015, pp. 230–249.

[51] Yin Tat Lee and Aaron Sidford. “Path Finding I: Solving Linear Programs with $\widetilde{O}(\sqrt{rank})$ Linear System Solves”. In: *arXiv preprint arXiv:1312.6677* (2013).

[52] Yin Tat Lee and Aaron Sidford. “Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow”. In: *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2014, pp. 424–433.

[53] Lihua Lei et al. “Non-convex finite-sum optimization via scsg methods”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 2348–2358.

[54] Yi Li, Huy L Nguyên, and David P Woodruff. “On sketching matrix norms and the top singular vector”. In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2014, pp. 1562–1581.

[55] Edo Liberty. “Simple and deterministic matrix sketching”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*. ACM. 2013, pp. 581–588.

[56] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. "A universal catalyst for first-order optimization". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3384–3392.

[57] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. "Catalyst Acceleration for First-order Convex Optimization: from Theory to Practice". In: *arXiv preprint arXiv:1712.05654* (2017).

[58] Yichao Lu et al. "Faster ridge regression via the subsampled randomized hadamard transform". In: *Advances in neural information processing systems*. 2013, pp. 369–377.

[59] Zhuang Ma, Yichao Lu, and Dean Foster. "Finding linear structure in large datasets with scalable canonical correlation analysis". In: *International Conference on Machine Learning*. 2015, pp. 169–178.

[60] Aleksander Madry. "Navigating central path with electrical flows: From flows to matchings, and back". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2013, pp. 253–262.

[61] Eric Moulines and Francis R Bach. "Non-asymptotic analysis of stochastic approximation algorithms for machine learning". In: *Advances in Neural Information Processing Systems*. 2011, pp. 451–459.

[62] Tomoya Murata and Taiji Suzuki. "Doubly accelerated stochastic variance reduced dual averaging method for regularized empirical risk minimization". In: *Advances in Neural Information Processing Systems*. 2017, pp. 608–617.

[63] Elizbar A Nadaraya. "On estimating regression". In: *Theory of Probability & Its Applications* 9.1 (1964), pp. 141–142.

[64] Jelani Nelson and Huy L Nguyên. "Lower bounds for oblivious subspace embeddings". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 883–894.

[65] Jelani Nelson, Huy L Nguyên, and David P Woodruff. "On deterministic sketching and streaming for sparse recovery and norm estimation". In: *Linear Algebra and its Applications* 441 (2014), pp. 152–167.

[66] Arkadi Nemirovski et al. "Robust stochastic approximation approach to stochastic programming". In: *SIAM Journal on optimization* 19.4 (2009), pp. 1574–1609.

[67] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2004.

[68] Yurii Nesterov. "Introductory lectures on convex programming volume i: Basic course". In: *Lecture notes* (1998).

[69] Yurii E Nesterov. "A method for solving the convex programming problem with convergence rate $O(1/k^2)$". In: *Dokl. Akad. Nauk SSSR*. Vol. 269. 1983, pp. 543–547.

[70] Yurii Nesterov and Sebastian U Stich. "Efficiency of the accelerated coordinate descent method on structured optimization problems". In: *SIAM Journal on Optimization* 27.1 (2017), pp. 110–123.

[71] Mert Pilanci and Martin J Wainwright. “Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence”. In: *SIAM Journal on Optimization* 27.1 (2017), pp. 205–245.

[72] Boris T Polyak and Anatoli B Juditsky. “Acceleration of stochastic approximation by averaging”. In: *SIAM Journal on Control and Optimization* 30.4 (1992), pp. 838–855.

[73] Eric C. Price. “Sparse recovery and Fourier sampling”. PhD thesis. Massachusetts Institute of Technology, 2013.

[74] Eric Price, Zhao Song, and David P. Woodruff. “Fast regression with an ℓ_∞ guarantee”. In: *International Colloquium on Automata, Languages, and Programming (ICALP)*. 2017.

[75] Sashank J Reddi et al. “Stochastic variance reduction for nonconvex optimization”. In: *International conference on machine learning*. 2016, pp. 314–323.

[76] Mark Schmidt, Nicolas Le Roux, and Francis Bach. “Minimizing finite sums with the stochastic average gradient”. In: *Mathematical Programming* 162.1-2 (2017), pp. 83–112.

[77] Shai Shalev-Shwartz. “Sdca without duality, regularization, and individual convexity”. In: *International Conference on Machine Learning*. 2016, pp. 747–754.

[78] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[79] Shai Shalev-Shwartz and Tong Zhang. “Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization”. In: *International Conference on Machine Learning*. 2014, pp. 64–72.

[80] Shai Shalev-Shwartz and Tong Zhang. “Stochastic dual coordinate ascent methods for regularized loss minimization”. In: *Journal of Machine Learning Research* 14.Feb (2013), pp. 567–599.

[81] Fanhua Shang et al. “Variance reduced stochastic gradient descent with sufficient decrease”. In: *arXiv preprint arXiv:1703.06807* (2017).

[82] Daniel A Spielman and Shang-Hua Teng. “Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time”. In: *Journal of the ACM (JACM)* 51.3 (2004), pp. 385–463.

[83] Volker Strassen. “Gaussian elimination is not optimal”. In: *Numerische Mathematik* 13.4 (1969), pp. 354–356.

[84] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[85] Pravin M Vaidya. “An algorithm for linear programming which requires $O(((m+n)n^2 + (m+n)^{1.5}n)L)$arithmetic operations”. In: *FOCS*. IEEE, 1987.

[86] Pravin M Vaidya. “Speeding-up linear programming using fast matrix multiplication”. In: *FOCS*. IEEE, 1989.

[87] Vladimir Vapnik. “Principles of risk minimization for learning theory”. In: *Advances in neural information processing systems*. 1992, pp. 831–838.

[88] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[89] Geoffrey S Watson. “Smooth regression analysis”. In: *Sankhyā: The Indian Journal of Statistics, Series A* (1964), pp. 359–372.

[90] Virginia Vassilevska Williams. “Multiplying matrices faster than Coppersmith-Winograd”. In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*. ACM. 2012, pp. 887–898.

[91] David P. Woodruff. “Sketching as a Tool for Numerical Linear Algebra”. In: *Foundations and Trends in Theoretical Computer Science* 10.1-2 (2014), pp. 1–157.

[92] Stephen J Wright. *Primal-dual interior-point methods*. Vol. 54. Siam, 1997.

[93] Lin Xiao and Tong Zhang. “A proximal stochastic gradient method with progressive variance reduction”. In: *SIAM Journal on Optimization* 24.4 (2014), pp. 2057–2075.

[94] Eric P Xing, Michael I Jordan, and Stuart Russell. “A generalized mean field algorithm for variational inference in exponential families”. In: *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 2002, pp. 583–591.

[95] Yinyu Ye, Michael J Todd, and Shinji Mizuno. “An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm”. In: *Mathematics of Operations Research* 19.1 (1994), pp. 53–67.

[96] Lijun Zhang, Tianbao Yang, and Rong Jin. “Empirical Risk Minimization for Stochastic Convex Optimization: $O(1/n)$-and $O(1/n^2)$-type of Risk Bounds”. In: *arXiv preprint arXiv:1702.02030* (2017).

[97] Yuchen Zhang and Lin Xiao. “Stochastic primal-dual coordinate method for regularized empirical risk minimization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 2939–2980.

[98] Shun Zheng et al. “A general distributed dual coordinate optimization framework for regularized loss minimization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4096–4117.

[99] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.