An Online and Unified Algorithm for Projection Matrix Vector Multiplication with Application to Empirical Risk Minimization

Lianke Qin
USCB

Zhao Song
Adobe Research

Lichen Zhang
MIT

Danyang Zhuo
Duke University

Abstract

Online matrix vector multiplication is a fundamental step and bottleneck in many machine learning algorithms. It is defined as follows: given a matrix at the pre-processing phase, at each iteration one receives a query vector and needs to form the matrix-vector product (approximately) before observing the next vector. In this work, we study a particular instance of such problem called the online projection matrix vector multiplication. Via a reduction, we show it suffices to solve the inverse maintenance problem. Additionally, our framework supports dimensionality reduction to speed up the computation that approximates the matrix-vector product with an optimization-friendly error guarantee. Moreover, our unified approach can handle both data-oblivious sketching and data-dependent sampling. Finally, we demonstrate the effectiveness of our framework by speeding up the empirical risk minimization solver.

1 INTRODUCTION

Online matrix-vector multiplication (OMv) (Henzinger et al., 2015; Larsen and Williams, 2017) is a fundamental problem in machine learning, e.g., gradient descent, accelerated method, and Netwon method (Nesterov, 1983, 1998; Boyd and Vandenberghe, 2004). The problem can be defined as follows: given a matrix $A \in \mathbb{R}^{n \times n}$ (with possible pre-processing). Let T denote the number of rounds, or the number of vectors we will receive. The goal of OMv is, in each round $t \in [T]$, we receive a query vector $h_t \in \mathbb{R}^n$ and we need to output the product Ah_t for each round. Note that the key difficulty of this problem is to output the matrix-vector each iteration. Otherwise, one can delay and batch the vectors into a matrix and apply fast matrix multiplication in n^ω time for $T = n$ query vectors.

A particular instance of online matrix-vector multiplication task is when the matrix A is a projection matrix. We refer this problem to online projection matrix-vector multiplication (OPMv). OPMv is a crucial sub-task in many fundamental optimization problems, such as linear programming and empirical risk minimization. Prior works on speeding up OPMv can be dated back as early as thirty years ago. The pioneer work of (Vaidya, 1989) shows that efficient linear program solver is essentially equivalent to OPMv, and can be solved in $O(n^2)$ time. Coupled with the required $\sqrt{n}$ iterations, (Vaidya, 1989) offers an LP solver that runs in $O(n^{2.5})$ time. This runtime is improved in 2019 due to (Cohen et al., 2019b), which speeds it up further by combining with fast rectangular matrix multiplication to achieve n^ω time for solving a general linear program where $\omega \approx 2.373$.

The major breakthrough of (Cohen et al., 2019b) leverages two central techniques: 1). update the projection matrix in a lazy fashion and 2). using a novel sampling approach to sparsify the vector-to-multiply so that the sparsity of the target vector is only k instead of n. For linear programming, they show that it suffices to choose $k = \sqrt{n}$. Following their work, many new approaches have emerged to realize similar speedup. In (Lee et al., 2019), they give a comparable running time for empirical risk minimization (ERM) via the use of a sketching matrix. Specifically, they apply a sketching matrix on the left of the projection, so that in each iteration, one only needs to compute RPh which is a matrix-vector product between $\sqrt{n} \times n$ matrix and length n vector. (Song and Yu, 2021) presents an alternative approach in which one can do this sketching trick on the right: at each iteration, one maintains the "left matrix" $PR^\top$ and perform the product between $PR^\top$ and Rh.

While most of these works obtain similar and comparable results in terms of running time, their approaches are drastically different, and their analysis requires various guarantees of the dimensionality reduction tool. Moreover, projection maintenance itself is hard to analyze and generalize in a unified fashion due to its sophisticated form.

Our paper demonstrates a different approach. We show that solving OPMv is equivalent to solving the inverse mainte-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

nance problem via a reduction. Specifically, we make use of Schur complement and show that by carefully arranging a large matrix, (part of) its inverse can be used to perform the projection matrix and vector product.

This equivalence result can even allow us to apply dimensionality reduction tools to the inverse maintenance problem to OPMv, such as sketching and sampling, to speed up the computation. We show that one can either add the sketching matrices directly into the inverse matrix we are to maintain, and during query time, it is enough to query a subset of rows and columns. Alternatively, when using data-dependent sampling as in (Cohen et al., 2019b), we can use the sampling matrix to sparsify the vector first, then use the inverse matrix to multiply a properly designed vector, to obtain a fast running time. Our reduction provides a concrete pipeline on solving the OPMv and we hope its detailed design can help future works on this problem.

Another important result is an ultimate unification of the data-dependent sparsification approach (Cohen et al., 2019b) into the *coordinate-wise embedding* framework, introduced in (Song and Yu, 2021). Roughly speaking, we say a random matrix R that might be oblivious or data-dependent satisfying the coordinate-wise embedding property if for fixed vectors g, h, the approximate inner product $g^\top R^\top R h$ is close to the true inner product $g^\top h$ with high probability. In other words, the matrix R behaves like a Johnson-Lindenstrauss transform (Johnson and Lindenstrauss, 1984). While it is appealing to design a data-oblivious distribution on R, we show that once R can be data-dependent, it obtains an optimal set of parameters. We believe such a data-dependent coordinate-wise embedding will find more applications in which ℓ_∞ guarantees are desirable.

Finally, we show a concrete application of our approach to empirical risk minimization. In (Lee et al., 2019), Lee, Song and Zhang show that ERM can be solved in the current matrix multiplication time, but its running time has an extra dependence on $\log^6(\log(1/\delta))$, where δ is the error parameter. Using our unified framework, we show that this factor is a consequence of sketching on the left, i.e., RPh. This causes the central path to be infeasible and subsequently, the data structure has to be restarted to balance off the variance. We remove this $\mathrm{poly}\log(\log(1/\delta))$ factor via *sketching on the right*, which is also the Song and Yu (2021) approach for solving linear program. Our result also provides a more structural understanding of the OPMv sub-task in the ERM solver, as the goal is to preserve the product between P and h instead of I and Ph.

We list our contributions as follows:

- For the online projection matrix-vector product problem, we provide a unified solution to that problem. Our approach is compatible with both data-oblivious approach (Lee et al., 2019; Song and Yu, 2021) and the data-dependent approach (Cohen et al., 2019b).
- We improve the running time of (Lee et al., 2019). We crucially shave off an unnatural $\log^6(\log(1/\delta))$ factor in their result by adopting a better sketching strategy.
- We present a novel sketching-based dynamic projection inverse data structure, which unifies the algorithm in (Cohen et al., 2019b; Lee et al., 2019; Song and Yu, 2021).

1.1 Related Work

Empirical Risk Minimization The problem of empirical risk minimization (ERM) is critical for various machine learning workloads (Nesterov, 1983; Polyak and Juditsky, 1992; Nemirovski et al., 2009; Nesterov, 2013; Brownlees et al., 2015; Zhang and Lin, 2015; Donini et al., 2018; Huang et al., 2020). One important line of work is to use first-order methods to improve ERM (Johnson and Zhang, 2013; Xiao and Zhang, 2014; Frostig et al., 2015; Mokhtari and Ribeiro, 2017; Jin et al., 2018; Wang et al., 2019).

Linear Programming Linear programming is an old topic in computer science. Simplex algorithm (Dantzig, 1947) is one of the most important algorithms in the history of linear programming, and it has an exponential running time. The Ellipsoid method reduced the running time to polinomial (Khachiyan, 1980), however, it is in practice slower than simplex method. The interior point method (Karmarkar, 1984) is a major breakthrough because it has theoretical polynomial running time and stably fast practical performance on real-world problems. Assuming d is the number of constraints and n is the number of variables, when $d = \Omega(n)$, Karmarkar's algorithm has a running time of $O^*(n^{3.5})$. The running time is further improved to $O^*(n^3)$ in (Vaidya, 1987; Renegar, 1988) and $O^*(n^{2.5})$ in (Vaidya, 1989).

Sketching Sketching is a well-known technique to improve performance or memory complexity (Clarkson and Woodruff, 2013). It has wide applications in linear algebra, such as linear regression and low-rank approximation(Clarkson and Woodruff, 2013; Nelson and Nguyên, 2013; Meng and Mahoney, 2013; Razenshteyn et al., 2016; Song et al., 2017; Haupt et al., 2017; Andoni et al., 2018; Song et al., 2019a,b; Diao et al., 2019), training over-parameterized neural network (Song et al., 2021b,c; Zandieh et al., 2021), generative adversarial networks (Xiao et al., 2018), projected gradient descent (Hanzely et al., 2018; Xu et al., 2021), kernel methods (Avron et al., 2017; Ahle et al., 2020; Chen and Yang, 2021; Song et al., 2021a), tensor decomposition Song et al. (2019c), trace estimation (Jiang et al., 2021a), John Ellipsoid computation (Cohen et al., 2019a; Song et al., 2022), semi-definite programming (Gu and Song, 2022), cutting plane method Jiang et al. (2020), federated learning (Rothchild et al., 2020), distributed problems Woodruff and Zhong (2016); Boutsidis et al. (2016), clustering (Esfandiari et al., 2021), reinforcement learning (Andreas et al., 2017; Wang et al., 2020; Shrivastava et al., 2023) and linear programming Lee et al. (2019); Jiang et al. (2021b); Song and Yu (2021).

Roadmap We first present an overview of techniques we leverage in this paper in Section 2. We present several tools in Section 3. We discuss the dynamic inverse maintenance algorithm and its application to the sketching setting in Section 4. We then show how to combine sketching with inverse maintenance in Section 5. We show how to unify the importance sampling method under the coordinate-wise embedding guarantee in Section 6. We present an improved algorithm for empirical risk minimization application in Section 7. We summarize the limitation of our paper in Section 8. We conclude our paper in Section 9.

Notations For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. We use $\mathbb{E}[\cdot]$ to denote the expectation, $\mathbf{Var}[\cdot]$ to denote the variance and $\Pr[\cdot]$ to denote the probability. For a vector x, we use $\|x\|_2$ to denote its ℓ_2 norm. We use $\mathbf{0}_n$ to denote the all-0 vector of dimension n. For any matrix A, we use $A^\top$ to denote the transpose of matrix A. For any square matrix and invertible matrix A, we use A^{-1} to denote its inverse. For any positive diagonal matrix W, we use $\sqrt{W}$ and $W^{1/2}$ to denote a diagonal matrix where the (i,i)-th entry on the diagonal is $\sqrt{W_{i,i}}$. We use $W^{-1/2}$ to denote the diagonal matrix where the (i,i)-th entry on diagonal is $1/\sqrt{W_{i,i}}$. We use I to denote the identity matrix. For a function f, we use $\widetilde{O}(f)$ to denote $f \cdot \mathrm{poly}(\log f)$.

For a matrix A, we use $\|A\|$ to denote its spectral norm. We use $\|A\|_F$ to denote its Frobenius norm.

2 TECHNICAL OVERVIEW

We first give a brief review of the projection maintenance formulation in (Cohen et al., 2019b; Lee et al., 2019; Song and Yu, 2021). Given a projection matrix $P \in \mathbb{R}^{n \times n}$ and a vector $h \in \mathbb{R}^n$,

- In (Cohen et al., 2019b), they design a sampling matrix $D \in \mathbb{R}^{n \times n}$ that depends on the vector h, at each iteration, they update the projection matrix P and compute PDh. By design, D has $\sqrt{n}$ nonzero entries in expectation.

- In (Lee et al., 2019), they preprocess a batch of sketching matrices $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \dots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n \times n}$ and pre-compute $\mathsf{R}P$. During the update, it updates the representation $\mathsf{R}P$, and during a query, it picks one sketch R_l and computes $R_l P h$ first, then computes $R_l^\top R_l P h$. Each of $R_l \in \mathbb{R}^{\sqrt{n} \times n}$.

- In (Song and Yu, 2021), they use similar sketching matrices $\mathsf{R} \in \mathbb{R}^{n \times n}$ and pre-compute $P\mathsf{R}^\top$. During a query, it picks a sketch R_l and computes $R_l h$, $PR_l^\top$, then compute $PR_l^\top R_l h$. Again, each $R_l \in \mathbb{R}^{\sqrt{n} \times n}$.

To give an ultimate unification, we need to take care of two parts: 1). update the projection matrix P and 2). compute the (approximate) product Ph.

Maintain and Update Projection P via Schur Complement. Given

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

it is usually hard to understand its form under some changes to the diagonal matrix W. Instead, we show that one can carefully design a matrix L whose blocks consist of $W^{1/2}$, W^{-1} and A. Using the well-known Schur complement, the inversion L^{-1} contains information for us to “recover” the projection matrix to maintain. By multiplying L^{-1} with a crafted but easy to compute vector v, $L^{-1}v$ will give exactly Ph, the quantity we care about.

Schur complement also gives us much flexibility to furnish the design of L. In fact, we can also include the batched sketching matrix R into the correct locations of L and exactly recover the construction $\mathsf{R}P$ in (Lee et al., 2019) and $P\mathsf{R}^\top$ in (Song and Yu, 2021). In this way, the originally complicated maintenance and update procedure has been reduced to a very generic matrix inverse maintenance data structure that certain parts can be updated and queried.

Our framework has much freedom for whether to include the sketching matrix and vector into L. This is in fact crucial for our unification: to recover the result of (Cohen et al., 2019b), we cannot include the sampling matrix D and vector h into L, since both of them are dynamically changing. On contrary, (Lee et al., 2019) requires us to maintain both R and h into L, otherwise we'll have to require to many columns from the data structure during the query.

Unifying Sketching and Sampling via Coordinate-wise Embedding. The approximate projection-vector product takes in three different forms:

- PDh (see Figure 2) for (Cohen et al., 2019b),
- $R_l^\top R_l P h$ (see Figure 3) for (Lee et al., 2019) and
- $PR_l^\top R_l h$ (see Figure 4) for (Song and Yu, 2021).

At first glance, it is not obvious how can they achieve comparable results: for example, the sampling matrix D is dependent on the vector h, but the sketching matrix R_l is independent of both P and h. Also, (Lee et al., 2019) maintains the sketch on the left but (Song and Yu, 2021) puts it on the right.

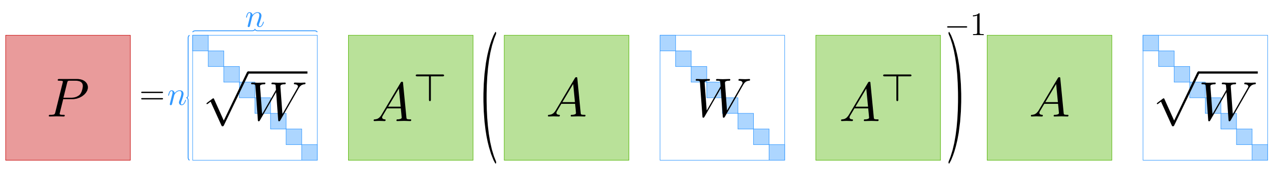

Figure 1: $P = \sqrt{W} A^\top (AWA^\top)^{-1} A \sqrt{W}$. $W \in \mathbb{R}^{n\times n}$ is a diagonal matrix. $P \in \mathbb{R}^{n\times n}$ is the projection matrix. $A \in \mathbb{R}^{d\times n}$ is the matrix with rank d in online matrix vector multiplication. We can carefully design a matrix L whose blocks consist of $W^{1/2}$, W^{-1} and A. Using the well-known Schur complement, the inversion L^{-1} contains information for us to "recover" the projection matrix to maintain.

To unify these approaches, we consider the so-called *coordinate-wise embedding*: roughly speaking, given a random matrix $R \in \mathbb{R}^{b\times n}$, we say it satisfies coordinate-wise embedding property if for any two vectors $g, h \in \mathbb{R}^n$

- In expectation, $g^\top R^\top R h$ is an unbiased estimator of $g^\top h$.
- The variance of $g^\top R^\top R h$ is small.
- With high probability, $g^\top R^\top R h$ is close to $g^\top h$ in absolute value.

This notion gives us a powerful tool to analyze their approaches: for example, we can view g as a row of the projection P, and such a guarantee tells us that $PR^\top Rh$ is close to Ph in both ℓ_2 and ℓ_∞ norm, which is exactly the property they need to show the convergence of the algorithm. Alternatively, view g to be e_i and righthand side vector to be Ph, we view the result of (Lee et al., 2019) as preserving the product between I and Ph so that it is close to Ph. This is a somewhat odd guarantee and we want to point out that this might be the main reason the central path in (Lee et al., 2019) is infeasible and their algorithm is sub-optimal — they use the wrong tool to approximate the product Ph.

One might ask: can we use this unification for the sampling scheme of (Cohen et al., 2019b)? The answer is positive. We show that, if we set $D = R^\top R$, then the sampling matrix satisfies coordinate-wise embedding. In fact, it gives the optimal parameters for this property. This should not be surprising, since D is dependent on the data, it should have stronger properties compared to the data-oblivious approach.

To compute the "sketching matrix" R in this case, we set it as a short and fat matrix consisting of the square root of the nonzero entries of D. However, the number of nonzero entries of D can only be guaranteed in expectation. This means that, while enjoying the optimality brought up by data-dependent sampling, the running time can only be guaranteed in expectation, which is the major downside of this method.

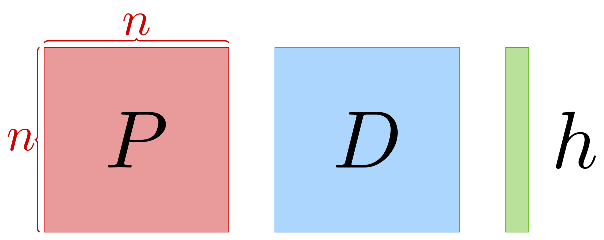

Figure 2: Approximate projection-vector product form PDh. $P = \sqrt{W} A^\top (AWA^\top)^{-1} A \sqrt{W}$ is the projection matrix. $D \in \mathbb{R}^{n\times n}$ is the sampling matrix and D has $\sqrt{n}$ nonzero entries in expectation. $h \in \mathbb{R}^n$ is the query vector. This is an illustration of (Cohen et al., 2019b).

Put things together: Faster ERM Solver. Now that we have simplified the data structure task in (Cohen et al., 2019b; Lee et al., 2019; Song and Yu, 2021) and unified the speedup tools they use, we develop a new algorithm for empirical risk minimization (ERM). More specifically, the state-of-the-art result by Lee, Song and Zhang (Lee et al., 2019) has a sub-optimal convergence rate induced by their algorithm design, or more concretely, the way they approximate Ph. They use the "left sketch", i.e., $R_l^\top R_l Ph$, by coordinate-wise embedding, this is not the right notion to approximate projection-vector product. We adapt the "right sketch" approach by computing $PR_l^\top R_l h$. By doing so, the central path is no longer infeasible and we do not need to restart the algorithm after a few iterations to control the variance.

3 PRELIMINARY

Theorem 3.1 (Rectangular matrix multiplication (Gall and Urrutia, 2018))**.** *Let the dual exponent of matrix multiplication α be the supremum among all $a \geq 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ matrix by an $n \times n^a$ matrix. Then, for any $n \geq r$, multiplying an $n \times r$ with an $r \times n$ matrix or $n \times n$ with $n \times r$ takes time*

$$n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

Furthermore, we have $\alpha > 0.31389$.

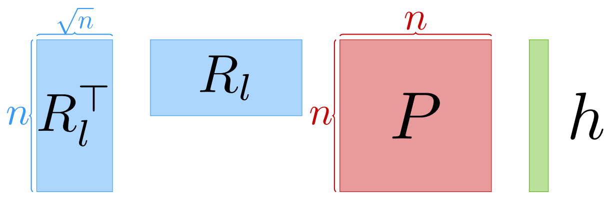

Figure 3: Approximate projection-vector product form $R_l^\top R_l P h$. $P = \sqrt{W} A^\top (AWA^\top)^{-1} A \sqrt{W}$ is the projection matrix. $R_l \in \mathbb{R}^{\sqrt{n} \times n}$ is the sketching matrix. $h \in \mathbb{R}^n$ is the query vector. This is an illustration of (Lee et al., 2019).

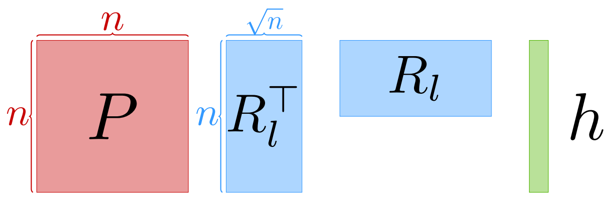

Figure 4: Approximate projection-vector product form $P R_l^\top R_l h$. $P = \sqrt{W} A^\top (AWA^\top)^{-1} A \sqrt{W}$ is the projection matrix. $R_l \in \mathbb{R}^{\sqrt{n} \times n}$ is the sketching matrix. $h \in \mathbb{R}^n$ is the query vector. This is an illustration of (Song and Yu, 2021).

Lemma 3.2 (Chernoff bound)**.** *Let $X = \sum_{i=1}^n X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^n p_i$. Then*

- $\Pr[X \geq (1+\delta)\mu] \leq \exp(-\delta^2 \mu/3)$, $\forall \delta > 0$;
- $\Pr[X \leq (1-\delta)\mu] \leq \exp(-\delta^2 \mu/2)$, $\forall 0 < \delta < 1$

Lemma 3.3 (Hoeffding bound)**.** *Let $X_1, \cdots, X_n$ denote n independent bounded variables in $[a_i, b_i]$. Let $X = \sum_{i=1}^n X_i$, then we have*

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2\exp(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}).$$

Lemma 3.4 (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,*

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

Lemma 3.5 (Woodburry matrix identity)**.** *The Woodburry matrix identity is*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

where A, U, C and V are conformable matrices: A has size $n \times n$, C has size $k \times k$, U has size $n \times k$ and V has size $k \times n$.

4 DYNAMIC INVERSE

In this section, we provide a detailed discussion of the dynamic inverse maintenance algorithm and its applications to sketching settings.

Our inverse maintenance technology starts from the well-known Schur complement:

Fact 4.1 (Schur complement)**.** *Given four matrices A, B, C, D, we have the following identity assuming that all inverses exist:*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix}$$

where

$$\begin{aligned} A_1 &= A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} \\ B_1 &= -A^{-1}B(D - CA^{-1}B)^{-1} \\ C_1 &= -(D - CA^{-1}B)^{-1}CA^{-1} \\ D_1 &= (D - CA^{-1}B)^{-1} \end{aligned}$$

Schur complement implies that if we carefully design the matrix-to-invert on the left hand side, we can make sure that the target projection matrix, even with sketching matrix and vector is at certain location of the inverse.

As the simplest example, consider matrix

$$L = \begin{bmatrix} U^{-1} & A^\top & U^{-1/2} & 0 \\ A & 0 & 0 & 0 \\ 0 & 0 & -I & 0 \\ (U^{-1/2})^\top & 0 & 0 & -I \end{bmatrix}$$

by using Schur complement, one can show that

$$L^{-1} = \begin{bmatrix} M^{-1} & M^{-1}N & 0 \\ 0 & -I & 0 \\ N^\top M^{-1} & N^\top M^{-1} N & -I \end{bmatrix}$$

in which

$$M^{-1} = \begin{bmatrix} U - UA^\top(AUA^\top)^{-1}AU & UA^\top(AUA^\top)^{-1} \\ (AUA^\top)^{-1}AU & -(AUA^\top)^{-1} \end{bmatrix},$$
$$N = \begin{bmatrix} U^{-1/2} \\ \mathbf{0}_{d \times n} \end{bmatrix}$$

If we multiply L^{-1} with a vector in the form of

$$\begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \end{bmatrix}$$

we have that

$$L^{-1}\begin{bmatrix}\mathbf{0}_{n+d}\\ v\\ v\end{bmatrix}=\begin{bmatrix}\star\\ \star\\ \sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v\end{bmatrix}$$

We can also add sketching matrix, or even the vector in the correct location of the L matrix to make sure its inverse provides what we want. For example, for left and right sketch, let $\mathsf{R}\in\mathbb{R}^{n\times n}$ denote the batched sketching matrices, then we have

$$L_{\text{left}}=\begin{bmatrix}U^{-1} & A^\top & U^{-1/2} & 0 & 0\\ A & 0 & 0 & 0 & 0\\ 0 & 0 & -I & 0 & 0\\ (U^{-1/2})^\top & 0 & 0 & -I & 0\\ 0 & 0 & 0 & \mathsf{R} & I\end{bmatrix}$$

and

$$L_{\text{right}}=\begin{bmatrix}U^{-1} & A^\top & U^{-1/2} & 0 & 0\\ A & 0 & 0 & 0 & 0\\ 0 & 0 & -I & 0 & \mathsf{R}^\top\\ (U^{-1/2})^\top & 0 & 0 & -I & \mathsf{R}^\top\\ 0 & 0 & 0 & 0 & I\end{bmatrix}$$

One can also insert the query vector h into the correct locations in the rightmost column, this will be particularly valuable for (Lee et al., 2019).

Now that we have shown the projection matrix, the sketching matrix, and the vector can be maintained in a carefully crafted matrix L, our data structure can be designed toward maintaining a matrix inversion. We also observe that due to this construction, we only need to modify certain parts of the matrix L.

To this end, we develop a generic inverse maintenance and update algorithm. Here we state an informal, and we refer the readers to Section B and Section C.

Theorem 4.2 (Informal). *Let $L\in\mathbb{R}^{m\times m}$ be an invertible matrix. There exists an algorithm with the following procedures:*

- INIT(L)*: It takes $O(m^\omega)$ time to compute the inverse L^{-1} and initialize corresponding variables.*
- UPDATE($\Delta\in\mathbb{R}^{m\times m}$)*: It takes a change matrix Δ with at most m^a nonzero entries in at most m^b nonzero columns and updates corresponding variables in time $O(m^{a+b}+m^{b\cdot\omega})$.*
- RESET($X,Y\in\mathbb{R}^{m\times m^c}$)*: Given matrices $X,Y\in\mathbb{R}^{m\times m^c}$, it updates the inverse to $(L+XY^\top)^{-1}$ and corresponding variables in time $O(m^{\omega(c,1,1)})$.*
- QUERY($I\subset[m],j\in[m]$)*: Given $I\subset[m]$ and an index $j\in[m]$, it outputs $(L^{-1})_{I,j}$ in time $O(m^{2b}+|I|\cdot m^a)$.*

The intuition of the data structure is that, if the update only affects a few columns/entries, then we can maintain an implicit representation efficiently. If the change is too much, we restart the data structure. The parameters a and b are crucial to balance the running time: if we don't maintain the vector inside L, then one can simply set $a=b$. Otherwise, we can maintain the vector in one column of L and update the vector in a slower fashion.

5 SKETCHING

Sketching is one of the most important tools to obtain speedup for central path method. Specifically, given a projection matrix $P\in\mathbb{R}^{n\times n}$ and a possibly dense vector $h\in\mathbb{R}^n$, the goal is to approximate the product Ph in $o(m^2)$ worst case time. Using sketching of size $\sqrt{n}\times n$, one either puts it on the left by maintaining $RP\in\mathbb{R}^{\sqrt{n}\times n}$ so that the product becomes $(RP)h$ which can be computed in time $O(n^{1.5})$, or puts it on the right by maintaining $PR^\top\in\mathbb{R}^{n\times\sqrt{n}}$ and compute $(PR^\top)(Rh)$ also in time $O(n^{1.5})$. Hence, it is important to incorporate sketching into the dynamic inverse maintenance data structure.

In this section, we show how to combine sketching with inverse maintenance. We first present some definitions and useful lemmas in Section 5.1. We present how we design the left sketching matrix in Section 5.2. We present how we design the right sketching matrix in Section 5.3.

5.1 Useful Tools

We first provide tools for matrix inverse in Lemma 5.2 and Lemma 5.5. Then with the matrix inverse vector multiplication result in Lemma I.2, we carefully design the left sketching matrix in Lemma 5.9 such that the matrix-vector multiplication result is

$$R_t\sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v.$$

Similarly, we present the right sketching matrix in Lemma 5.10 such that the matrix-vector multiplication result is

$$\sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}R_t^\top R_t v.$$

Definition 5.1 (M matrix). *Let $A\in\mathbb{R}^{d\times n}$ with rank d and $U\in\mathbb{R}^{n\times n}$ be a diagonal matrix with non-zero elements on the diagonal. We define the matrix $M\in\mathbb{R}^{(n+d)\times(n+d)}$ as follows:*

$$M=\begin{bmatrix}U^{-1} & A^\top\\ A & 0\end{bmatrix}$$

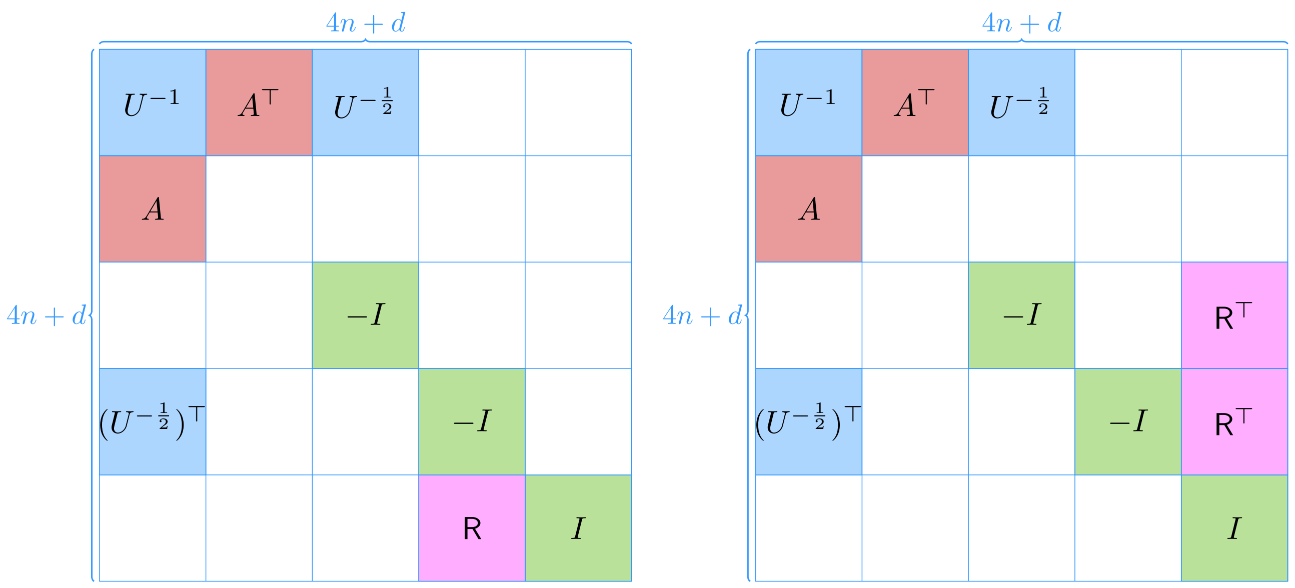

Figure 5: $L_{\mathrm{left}} \in \mathbb{R}^{(4n+d)\times(4n+d)}$ and $L_{\mathrm{right}} \in \mathbb{R}^{(4n+d)\times(4n+d)}$ matrice visualization. Here the $\mathsf{R} \in \mathbb{R}^{n\times n}$ denote the batched sketching matrices. $A \in \mathbb{R}^{d\times n}$ is the matrix in online matrix vector multiplication. $U \in \mathbb{R}^{n\times n}$ is a diagonal matrix with non-zero elements on the diagonal. $I \in \mathbb{R}^{n\times n}$ is an identity matrix. The L_{left} is the matrix corresponding to sketching on the left and the L_{right} is the matrix corresponding to sketching on the right. We defer the more detailed discussion on how to embed sketching into L to Section 5.

Lemma 5.2 (Informal version of Lemma F.1)**.** *Let $M \in \mathbb{R}^{(n+d)\times(n+d)}$ be defined as in Definition 5.1, then*

$$M^{-1} = \begin{bmatrix} U - UA^\top(AUA^\top)^{-1}AU & UA^\top(AUA^\top)^{-1} \\ (AUA^\top)^{-1}AU & -(AUA^\top)^{-1} \end{bmatrix}$$

We delay the proof to Appendix F.

Definition 5.3 (L matrix)**.** *Let $A \in \mathbb{R}^{d\times n}$ be rank d and $U \in \mathbb{R}^{n\times n}$ be a diagonal matrix with non-zero elements on the diagonal. We define the matrix $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ as follows*

$$L = \begin{bmatrix} U^{-1} & A^\top & U^{-1/2} & 0 \\ A & 0 & 0 & 0 \\ 0 & 0 & -I & 0 \\ (U^{-1/2})^\top & 0 & 0 & -I \end{bmatrix}$$

To get a better view of the inverse of L, we define the matrix N first.

Definition 5.4 (N matrix)**.** *Let $U \in \mathbb{R}^{n\times n}$ be a diagonal matrix with non-zero diagonal element. We define the matrix $N \in \mathbb{R}^{(n+d)\times n}$ as follows:*

$$N = \begin{bmatrix} U^{-1/2} \\ \mathbf{0}_{d\times n} \end{bmatrix}$$

We can express the inverse of $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ using $M \in \mathbb{R}^{(n+d)\times(n+d)}$ and $N \in \mathbb{R}^{(n+d)\times n}$.

Lemma 5.5. *Let $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ be defined in Definition 5.3. Then,*

$$L^{-1} = \begin{bmatrix} M^{-1} & M^{-1}N & 0 \\ 0 & -I & 0 \\ N^\top M^{-1} & N^\top M^{-1}N & -I \end{bmatrix}$$

where $M \in \mathbb{R}^{(n+d)\times(n+d)}$ and $N \in \mathbb{R}^{(n+d)\times n}$ are defined in Definition 5.1 and 5.4.

One important feature of the $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ matrix is multiplying its inverse with a proper vector giving the desired matrix-vector product of interest.

Lemma 5.6 (Restatement of Lemma I.2)**.** *Let L be defined as in Definition 5.3. Then we have*

$$L^{-1}\begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v \end{bmatrix}$$

Lemma 5.7 (Sketch on the left. Informal version of Lemma F.2)**.** *Let $R \in \mathbb{R}^{n\times(3n+d)}$, let $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ be the matrix defined in Definition 5.3, consider the matrix*

$$\begin{bmatrix} L & 0 \\ R & -I \end{bmatrix},$$

then we have

$$\left(\begin{bmatrix} L & 0 \\ R & -I \end{bmatrix}\right)^{-1} = \begin{bmatrix} L^{-1} & 0 \\ RL^{-1} & -I \end{bmatrix}$$

We delay the proof to Appendix F.

Lemma 5.8 (Sketch on the right. Informal version of Lemma F.3)**.** *Let $R \in \mathbb{R}^{(3n+d)\times n}$, let $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ be the matrix defined in Definition 5.3, consider the matrix*

$$\begin{bmatrix} L & R \\ 0 & -I \end{bmatrix},$$

then we have

$$\left(\begin{bmatrix} L & R \\ 0 & -I \end{bmatrix}\right)^{-1} = \begin{bmatrix} L^{-1} & L^{-1}R \\ 0 & -I \end{bmatrix}$$

We delay the proof to Appendix F.

5.2 Design the Left Sketching Matrix

As we have shown, it is possible to multiply a conforming matrix R either on the left or on the right of $L^{-1} \in \mathbb{R}^{(3n+d)\times(3n+d)}$. We now show how to design proper sketching matrices. We start with the discussion on sketching on the left.

Lemma 5.9 (Informal version of Lemma F.4)**.** *Let $\mathsf{R} \in \mathbb{R}^{n\times n}$ be a collection of sketching matrices, define $R \in \mathbb{R}^{n\times(3n+d)}$ to be the following matrix:*

$$R = \begin{bmatrix} \mathbf{0}_{n\times(2n+d)} & I_t \mathsf{R} \end{bmatrix}$$

Then we have

$$\left(\begin{bmatrix} L & \mathbf{0}_{(3n+d)\times n} \\ R & -I_n \end{bmatrix}\right)^{-1} \begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \\ \mathbf{0}_n \end{bmatrix} = \begin{bmatrix} \star \\ R_t\sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v \end{bmatrix}$$

where I_t is a diagonal matrix whose $\frac{n(t-1)}{T}$-th to $\frac{nt}{T}$-th diagonal entries are 1 *and other diagonal entries are* 0 *such that $I_t\mathsf{R} = R_t$.*

We delay the proof to Appendix F.

5.3 Design the Right Sketching Matrix

Lemma 5.10 (Informal version of Lemma F.5)**.** *Let*

$$\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \cdots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n\times n}$$

be a collection of sketching matrices. Let $T = \sqrt{n}$. Define $\mathsf{B} \in \mathbb{R}^{(3n+d)\times n}$ to be the following matrix:

$$\mathsf{B} = \begin{bmatrix} \mathbf{0}_{(n+d)\times n} \\ \mathsf{R}^\top \\ \mathsf{R}^\top \end{bmatrix}$$

Then we have

$$\left(\begin{bmatrix} L & \mathsf{B} \\ \mathbf{0}_{n\times(3n+d)} & -I_n \end{bmatrix}\right)^{-1} \begin{bmatrix} \mathbf{0}_{3n+d} \\ I_t\mathsf{R}v \end{bmatrix} = \begin{bmatrix} \star \\ \sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}R_t^\top R_t v \end{bmatrix}$$

where I_t is a diagonal matrix whose $\frac{n(t-1)}{T}$-th to $\frac{nt}{T}$-th diagonal entries are 1 *and other diagonal entries are* 0 *such that $\mathsf{R}^\top I_t \mathsf{R} = R_t^\top R_t$.*

We delay the proof to Appendix F.

6 UNIFYING IMPORTANCE SAMPLING WITH COORDINATE-WISE EMBEDDING

While (Lee et al., 2019; Song and Yu, 2021) uses randomized sketching techniques to speed up the projection-vector product, (Cohen et al., 2019b) uses an importance sampling method depending on the vector h. The advantage of this approach is it avoids preprocessing by batching sketches.

In this section, we show that the importance sampling method can be unified under the *coordinate-wise embedding* guarantee introduced in (Song and Yu, 2021). We first recall the definition:

Definition 6.1 ((α,β,δ)-coordinate wise embedding)**.** *We say a randomized matrix $R \in \mathbb{R}^{b\times n}$ satisfying (α,β,δ)-coordinate wise embedding if*

1. $\mathbb{E}_{R\sim\Pi}[g^\top R^\top R h] = g^\top h,$
2. $\mathbb{E}_{R\sim\Pi}[(g^\top R^\top R h)^2] \le (g^\top h)^2 + \frac{\alpha}{b}\|g\|_2^2\|h\|_2^2,$
3. $\Pr_{R\sim\Pi}\left[|g^\top R^\top R h - g^\top h| \ge \frac{\beta}{\sqrt{b}}\|g\|_2\|h\|_2\right] \le \delta.$

Remark 6.2. *Given a randomized matrix $R \in \mathbb{R}^{b\times n}$ satisfying (α,β,δ)-coordinate wise embedding and any orthogonal projection $P \in \mathbb{R}^{n\times n}$, above definition implies*

1. $\mathbb{E}_{R\sim\Pi}[PR^\top R h] = Ph,$
2. $\mathbb{E}_{R\sim\Pi}[(PR^\top R h)_i^2] \le (Ph)_i^2 + \frac{\alpha}{b}\|h\|_2^2,$
3. $\Pr_{R\sim\Pi}\left[|(PR^\top R h)_i - (Ph)_i| \ge \frac{\beta}{\sqrt{b}}\|h\|_2\right] \le \delta.$

since $\|P\|_2 \le 1$ implies $\|P_{i,:}\|_2 \le 1$ for all $i \in [n]$.

In (Cohen et al., 2019b), they use a diagonal sampling matrix $D \in \mathbb{R}^{n\times n}$ with roughly b non-zero entries on the diagonal. We design the matrix $R \in \mathbb{R}^{b\times n}$ as follows: let S be the set of indices of non-zeros in D, then we set $R_{i,i} = D_{i,i}$ for $i \in S$.

The matrix D is designed as follows: given $h \in \mathbb{R}^n$, we have

$$D_{i,i} = \begin{cases} \frac{1}{p_i}, & \text{with probability } p_i := b \cdot \left(\frac{h_i^2}{\|h\|_2^2} + \frac{1}{n}\right); \\ 0, & \text{otherwise.} \end{cases}$$

We prove the above diagonal sampling matrix satisfies the first two conditions of Definition 6.1.

Lemma 6.3 (Informal version of Lemma G.1)**.** *Let $D \in \mathbb{R}^{n\times n}$ be the sampling matrix defined as above. For any $g \in \mathbb{R}^n$, we have*

- $\mathbb{E}_D[g^\top Dh] = g^\top h.$
- $\mathbb{E}_D[(g^\top Dh)^2] = (g^\top h)^2 + \frac{1}{b}\|g\|_2^2\|h\|_2^2.$
- $\Pr_D[|g^\top Dh - g^\top h| \geq \frac{\log(1/\delta)}{\sqrt{b}}\|g\|_2\|h\|_2] \leq \delta.$

Remark 6.4. *Our above argument shows that the sampling matrix used in (Cohen et al., 2019b) is a $(1, \log(1/\delta), \delta)$-coordinate-wise embedding. This gives the optimal parameter for this property, due to its data-dependence nature.*

7 APPLICATION: EMPIRICAL RISK MINIMIZATION

In this section, we give an improved algorithm for empirical risk minimization (ERM) studied in (Lee et al., 2019). While they obtain an algorithm runs in the current matrix multiplication time for ERM, the speedup used in (Lee et al., 2019) comes from sketching on the left. Even though from a per iteration running time perspective, it is comparable to sketching on the right, or using the sampling-based approach, sketching on the left makes the central path infeasible. This means they have to restart their data structure after some iterations. Hence, the number of iterations required for their algorithm to converge is

$$O(\sqrt{n}\log^2 n \log(n/\delta)\log^6(\log 1/\delta)).$$

The unnatural

$$\log^6(\log(1/\delta))$$

the term is a direct consequence of the infeasibility of their sketching approach. Namely, they compute $R^\top RPh$ while we maintain $PR^\top Rh$. Even though the per iteration cost is similar, they have to pay extra log factors in number of iterations to "correct" the infeasibility caused by this sketch.

We show that if we instead *sketch on the right*, the central path will no longer be infeasible. In fact, as we have shown in the prior section, importance-based sampling is also a variant of the sketch on the right, and this approach is the key to obtaining a fast linear program solver in (Cohen et al., 2019b; Song and Yu, 2021).

Theorem 7.1 (Informal version of Theorem H.1)**.** *Given 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 $K_1, \ldots, K_m$. Assume there's no redundant constraints and $n_i = O(1)$, for $i \in [m]$. There exists an algorithm that solves*

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

up to δ precision and runs in the expected time

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

Our approach gives the optimal result in terms of the number of iterations required to converge. By using sketching on the right, we shave off the $\log^6\log(1/\delta)$ term in (Lee et al., 2019).

8 LIMITATION

There are two limitations of our approach. The first is that, when applying to matrices, the guarantees are very dependent on the spectral norm of the matrix. For the product $PR^\top Rh$, we note that the second moment and high probability bound all depend on $\|P\|$. This is one of the main reasons we motivate with projection matrices, as they not only be used in linear programs and ERMs, but $\|P\| \leq 1$ as well.

The second issue is that the guarantees are "coordinate-wise". For example, with high probability, we have

$$|(PR^\top Rh - Ph)_i| \leq \epsilon\|h\|_2.$$

This works well for frameworks seeking ℓ_∞ guarantees but is rather restrictive for other guarantees, e.g.,

$$\|PR^\top Rh\|_2.$$

9 CONCLUSION

Online matrix-vector multiplication is an important problem, and it is a basic building block for many machine learning algorithms. We study this problem when the matrix of interest is a projection matrix. We show that this problem while having a complicated form can be reduced to solving the inverse maintenance problem. Using this reduction, we design a generic inverse maintenance data structure when the projection is undergoing some multiplicative changes which can both maintain the projection and support matrix-vector product query efficiently. To implement a fast query procedure, we study the notion of coordinate-wise embedding, which can either be realized via data-oblivious dimensionality reduction or data-dependent importance sampling. Finally, we use our data structure to speed up empirical risk minimization, an important class of problems in convex optimization.

References

Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 141–160. SIAM, 2020.

Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with orlicz norm. In *International Conference on Machine Learning (ICML)*, pages 224–233. PMLR, 2018.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175. PMLR, 2017.

Haim Avron, Kenneth L Clarkson, and David P Woodruff. Faster kernel ridge regression using sketching and preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1116–1138, 2017.

Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 236–249, 2016.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. 2004.

Christian Brownlees, Emilien Joly, and Gábor Lugosi. Empirical risk minimization for heavy-tailed losses. *The Annals of Statistics*, 43(6):2507–2536, 2015.

Yifan Chen and Yun Yang. Accumulations of projections—a unified framework for random sketches in kernel ridge regression. In *International Conference on Artificial Intelligence and Statistics*, pages 2953–2961. PMLR, 2021.

Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, 2013.

Michael B Cohen, Ben Cousins, Yin Tat Lee, and Xin Yang. A near-optimal algorithm for approximating the john ellipsoid. In *Conference on Learning Theory*, pages 849–873. PMLR, 2019a.

Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *STOC*, 2019b.

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

Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David Woodruff. Optimal sketching for kronecker product regression and low rank approximation. *Advances in neural information processing systems*, 32, 2019.

Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. *Advances in Neural Information Processing Systems*, 31, 2018.

Hossein Esfandiari, Vahab Mirrokni, and Peilin Zhong. Almost linear time density level set estimation via dbscan. In *AAAI*, 2021.

Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pages 2540–2548. PMLR, 2015.

François 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*, pages 1029–1046. SIAM, 2018.

Yuzhou Gu and Zhao Song. A faster small treewidth sdp solver. *arXiv preprint arXiv:2211.06033*, 2022.

Filip Hanzely, Konstantin Mishchenko, and Peter Richtárik. Sega: Variance reduction via gradient sketching. *Advances in Neural Information Processing Systems*, 31, 2018.

Jarvis Haupt, Xingguo Li, and David P Woodruff. Near optimal sketching of low-rank tensor regression. In *ICML*, 2017.

Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, 2015.

Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. *Advances in neural information processing systems*, 33: 19365–19376, 2020.

Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games and its applications. In *STOC*, 2020.

Shuli Jiang, Hai Pham, David Woodruff, and Richard Zhang. Optimal sketching for trace estimation. *Advances in Neural Information Processing Systems*, 34, 2021a.

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. Faster dynamic matrix inverse for faster lps. In *STOC*. arXiv preprint arXiv:2004.07470, 2021b.

Chi Jin, Lydia T Liu, Rong Ge, and Michael I Jordan. On the local minima of the empirical risk. *Advances in neural information processing systems*, 31, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

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

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

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

Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2182–2189. SIAM, 2017.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*, 2019.

Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pages 91–100, 2013.

Aryan Mokhtari and Alejandro Ribeiro. First-order adaptive sample size methods to reduce complexity of empirical risk minimization. *Advances in Neural Information Processing Systems*, 30, 2017.

Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

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

Y Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Springer Science & Business Media, 2013.

Yu Nesterov. Introductory lectures on convex programming, 1998.

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

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

Ilya Razenshteyn, Zhao Song, and David P. Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, page 250–263, 2016.

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

Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

Anshumali Shrivastava, Zhao Song, and Zhaozhuo Xu. A tale of two efficient value iteration algorithms for solving linear mdps with large action space. In *AISTATS*, 2023.

Zhao Song and Zheng Yu. Oblivious sketching-based central path method for solving linear programming problems. In *38th International Conference on Machine Learning (ICML)*, 2021.

Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*, 2017.

Zhao Song, David Woodruff, and Peilin Zhong. Average case column subset selection for entrywise ℓ_1-norm loss. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:10111–10121, 2019a.

Zhao Song, David Woodruff, and Peilin Zhong. Towards a zero-one law for column subset selection. *Advances in Neural Information Processing Systems*, 32:6123–6134, 2019b.

Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *SODA*. arXiv preprint arXiv:1704.08246, 2019c.

Zhao Song, David Woodruff, Zheng Yu, and Lichen Zhang. Fast sketching of polynomial kernels of polynomial degree. In *International Conference on Machine Learning*, pages 9812–9823. PMLR, 2021a.

Zhao Song, Shuo Yang, and Ruizhe Zhang. Does preprocessing help training over-parameterized neural networks? *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021b.

Zhao Song, Lichen Zhang, and Ruizhe Zhang. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint arXiv:2112.07628*, 2021c.

Zhao Song, Xin Yang, Yuanyuan Yang, and Tianyi Zhou. Faster algorithm for structured john ellipsoid computation. *arXiv preprint arXiv:2211.14407*, 2022.

Pravin M. Vaidya. An algorithm for linear programming which requires o(((m+n)nˆ2 + (m+n)ˆ1.5 n)l) arithmetic operations. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 29–38. ACM, 1987.

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

Po-Wei Wang, Ching-pei Lee, and Chih-Jen Lin. The common-directions method for regularized empirical risk minimization. *The Journal of Machine Learning Research*, 20(1):2072–2120, 2019.

Ruosong Wang, Peilin Zhong, Simon S Du, Russ R Salakhutdinov, and Lin F Yang. Planning with general objective functions: Going beyond total rewards. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

David P Woodruff and Peilin Zhong. Distributed low rank approximation of implicit functions of a matrix. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 847–858. IEEE, 2016.

Chang Xiao, Peilin Zhong, and Changxi Zheng. Bourgan: generative networks with metric embeddings. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, pages 2275–2286, 2018.

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

Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip datastructures. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Amir Zandieh, Insu Han, Haim Avron, Neta Shoham, Chaewon Kim, and Jinwoo Shin. Scaling neural tangent kernels via sketching and random features. *Advances in Neural Information Processing Systems*, 34, 2021.

Yuchen Zhang and Xiao Lin. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *International Conference on Machine Learning*, pages 353–361. PMLR, 2015.

Appendix

Roadmap. We first introduce some notations and preliminary definition and theorem in Section A. We present the projection maintenance via inverse maintenance without vector and the sketch is on the left in Section B. We present the projection maintenance via inverse maintenance without vector and the sketch is on the right in Section C. We present the projection maintenance via inverse maintenance with vector and the sketch is on the left in Section D. We present the projection maintenance via inverse maintenance with vector and the sketch is on the right in Section E. We provide proofs for sketching matrix design in Section F. We analyze the error of applying sketching in Section G. We show our results on faster empirical risk minimization in Section H. We provide some discussion in Section I.

A PRELIMINARY

In this section, we present some notations and preliminary definition and theorems.

Notations. For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. We use $\mathbb{E}[\cdot]$ to denote the expectation, $\mathbf{Var}[\cdot]$ to denote the variance and $\Pr[\cdot]$ to denote the probability. For a vector x, we use $\|x\|_2$ to denote its ℓ_2 norm. We use $\mathbf{0}_n$ to denote the all-0 vector of dimension n. For any matrix A, we use $A^\top$ to denote the transpose of matrix A. For any square matrix and invertible matrix A, we use A^{-1} to denote its inverse. For any positive diagonal matrix W, we use $\sqrt{W}$ and $W^{1/2}$ to denote a diagonal matrix where the (i,i)-th entry on diagonal is $\sqrt{W_{i,i}}$. We use $W^{-1/2}$ to denote the diagonal matrix where the (i,i)-th entry on diagonal is $1/\sqrt{W_{i,i}}$. We use I to denote the identity matrix.

Definition A.1. *We define g_i as:*

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

Remark A.2. *The update time in (Cohen et al., 2019b) is $O(rg_r n^{2+o(1)}) = O(\sum_{i=1}^r g_i n^{2+o(1)})$ for any $r \geq n^a$. If $r = n^a$, update time is $O(n^{2+o(1)})$. If $r = n$, update time is $O(n^{\omega+o(1)})$*

B PROJECTION MAINTENANCE VIA INVERSE MAINTENANCE: ONE LAYER AND LEFT SKETCH WITHOUT VECTOR

We present a framework where we design a matrix M that contains the sketched projection $\mathsf{R}^\top \sqrt{U} A^\top (AUA^\top)^{-1} A \sqrt{U}$. When we need to perform the approximate matrix vector product, we have to query $\sqrt{n}$ rows and n columns, yields to a slower running time than (Lee et al., 2019).

We present initialization, update, reset and query functions in Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4. The INITIALIZE operation construct a left sketching matrix M from Lemma 5.9 and compute the inverse $N = M^{-1}$. The UPDATE operation updates the inverse matrix such that $Q = (I + Y^\top M^{-1} X)^{-1}$. The RESET operation reset the value of N such that $N = M^{-1}$. QUERY operation takes $I \subset [n]$ and an index $j \in [n]$ as input and computes v such that

$$v = (M^{-1})_{I,j} - (M^{-1} X (I + Y^\top M^{-1} X)^{-1} Y^\top M^{-1} e_j)_I.$$

UPDATEONESTEP operation takes an update matrix $W \in \mathbb{R}^{n \times n}$ and decides to call either UPDATE or RESET to maintain the inverse of the matrix. QUERYONESTEP calls QUERY operation for n times and obtain the projected sketching result y and z such that:

$$y = R_l^\top R_l \widetilde{P} h$$
$$z = R_l^\top R_l (I - \widetilde{P}) h$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

B.1 Main Result

In this section, we show how to implement the projection matrix maintenance task as an inverse maintenance problem.

Theorem B.1 (Sketching on left without vector). *Given a matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ where $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \cdots & R_T^\top \end{bmatrix}$. Let $T = \sqrt{n}$. Let $m = 4n + d$.*

Let $A \in \mathbb{R}^{d \times n}$ of rank d and let $U \in \mathbb{R}^{n \times n}$ be a diagonal matrix with non-zero diagonal entries, $v \in \mathbb{R}^n$ and $\mathsf{R} \in \mathbb{R}^{n \times n}$ be a sketch matrix. There exists a data structure PROJECTIONMAINTENANCELEFT *(Algorithm 1 and 2) which supports the following operations:*

- **public:**INIT$(a \in (0,1), b \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n \times n}, A \in \mathbb{R}^{d \times n}, h \in \mathbb{R}^n, \epsilon_{mp} \in (0, 1/4))$*: Given thresholds $a, b \in (0,1)$ with $b \le a$, a vector $u_0 \in \mathbb{R}^n$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$, a matrix $A \in \mathbb{R}^{d \times n}$, a vector $h \in \mathbb{R}^n$ and a tolerance parameter $\epsilon_{mp} \in (0, 1/4)$ as input,* INIT *(Algorithm 1) operation runs in $O(n^\omega)$ time.*

- **public:**UPDATEONESTEP$(W \in \mathbb{R}^{n \times n})$. *Given a positive block diagonal psd matrix $W \in \mathbb{R}^{n \times n}$, the output of* UPDATEONESTEP*(Algorithm 3) updates a diagonal block matrix $\widetilde{V}$ such that*

$$(1 - \epsilon_{mp}) \|w_i\|_F \preceq \|\widetilde{v}_i\|_F \preceq (1 + \epsilon_{mp}) \|w_i\|_F,$$

  where $w_i, \widetilde{v}_i$ denote the i-th block of $W, \widetilde{V}$ respectively.

  With the definition of g_r in Definition A.1, the time complexity of UPDATEONESTEP *is:*

$$\begin{cases} O(n^{a+b} + n^{b \cdot \omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \ge n^a \end{cases}$$

- **public:**QUERYONESTEP$(h \in \mathbb{R}^n)$. *Given a query vector $h \in \mathbb{R}^n$, the output of* QUERYONESTEP*(Algorithm 3) satisfies:*

$$y = R_l^\top R_l \widetilde{P} h$$
$$z = R_l^\top R_l (I - \widetilde{P}) h$$

  where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP. *And the time complexity of* QUERYONESTEP *is $O(n^{3/2} + n^{2b+1} + n^{3/2+a})$.*

- **private:**UPDATE$(u^{\text{new}} \in \mathbb{R}^n)$*: Given a vector $u^{\text{new}} \in \mathbb{R}^n$ as input, the* UPDATE *(Algorithm 1) operation runs $O(n^{a+b} + n^{b \cdot \omega})$ time.*

- **private:**QUERY$(I \subset [n], j \in [n])$*: Given $I \subset [n]$ and an index $j \in [n]$ as input, the* QUERY *(Algorithm 2) operation runs in $O(n^{2b} + |I| \cdot n^a)$ time.*

- **private:**RESET$(X^{\text{new}}, Y^{\text{new}})$*: Give matrices $X^{\text{new}}, Y^{\text{new}} \in \mathbb{R}^{n \times n^c}$,* RESET *(Algorithm 2) operations runs in $O(n^{\omega(c,1,1)})$ time.*

Remark B.2. *Note that the above theorem is not strong enough to reproduce the result of (Lee et al., 2019). Specifically, note the query takes time $O(n^{3/2+a})$, by picking $a = \min(\alpha, 1/3)$ where $\alpha \approx 0.31$, this gives an overall running time of*

$$O(n^\omega + n^{2+1/3}).$$

This is due to the fact that we need to query $\sqrt{n}$ rows and n columns. Later, we will see how to store h into M so that we only need to query one column, thus recover the result of (Lee et al., 2019).

Proof. The correctness follows from combining Lemma B.3, Lemma B.4 and Lemma B.5.

The running time follows from combining Lemma B.6, Lemma B.7, Lemma B.8 and Lemma B.9. ☐

Algorithm 1 1-layer projection maintenance by inverse maintenance with Sketching on the left. Note that we store the inverse of M, but only update it during the procedure RESET. For UPDATE, we assume the change happens for at most n^a entries in at most n^b columns.

```
1:  data structure ProjectionMaintenanceLeft                                   ▷ Theorem B.1
2:  members
3:      $U \in \mathbb{R}^{n \times n}$
4:      $A \in \mathbb{R}^{d \times n}$
5:      $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \dots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n \times n}$
6:      $M \in \mathbb{R}^{(4n+d) \times (4n+d)}$
7:      $a \in (0,1)$                                                          ▷ Threshold 1
8:      $b \in (0,1)$                                                          ▷ Threshold 2
9:      $\epsilon_{mp} \in (0, 1/4)$                                           ▷ Tolerance
10:     $X, Y \in \mathbb{R}^{(4n+d) \times n^b}$               ▷ Each column of $Y$ has only one nonzero entry
11:     $Q \in \mathbb{R}^{n^b \times n^b}$
12:     $N \in \mathbb{R}^{(4n+d) \times (4n+d)}$                              ▷ Inverse of $M$
13:     $v, \widetilde{v} \in \mathbb{R}^n$
14:     $l \in \mathbb{N}$
15: end members
16: private:
17: procedure Init($a, b \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n \times n}, A \in \mathbb{R}^{d \times n}, \epsilon_{mp}$)   ▷ Lemma B.6
18:     $a \leftarrow a, b \leftarrow b, A \leftarrow A, \epsilon_{mp} \leftarrow \epsilon_{mp}$
19:     $U \leftarrow \mathrm{diag}(u_0)$
20:     $\mathsf{R} \leftarrow \mathsf{R}$
21:     $M \leftarrow \begin{bmatrix} U^{-1} & A^\top & U^{-1/2} & 0 & 0 \\ A & 0 & 0 & 0 & 0 \\ 0 & 0 & -I & 0 & 0 \\ (U^{-1/2})^\top & 0 & 0 & -I & 0 \\ 0 & 0 & 0 & \mathsf{R} & I \end{bmatrix}$   ▷ Left sketching matrix construction from Lemma 5.9
22:     $X, Y \leftarrow \mathbf{0}_{(4n+d) \times n^b}$
23:     $N \leftarrow M^{-1}$                                                  ▷ Takes $O(n^\omega)$ time
24:     $Q \leftarrow I_{n^b}$
25:     $l \leftarrow 0$
26: end procedure
27: private:
28: procedure Update($u^{\mathrm{new}} \in \mathbb{R}^n$)                      ▷ Lemma B.3 and Lemma B.7
29:                           ▷ To run this procedure, we require that $u^{\mathrm{new}}$ has at most $n^b$ non-zero entries
30:     if $\|u^{\mathrm{new}}\|_0 > n^b$ then
31:         return error
32:     end if
33:     $\Delta \leftarrow \begin{bmatrix} (U^{\mathrm{new}})^{-1} & 0 & (U^{\mathrm{new}})^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ((U^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
34:     Let $\Delta = XY^\top$ where $X \in \mathbb{R}^{(4n+d) \times n^b}$ and $Y \in \mathbb{R}^{(4n+d) \times n^b}$   ▷ $X$ consists of nonzero entries, $Y$ is a column
    selection matrix
35:                                                  ▷ Each row of $Y$ has only 1 nonzero entry
36:     $X \leftarrow X, Y \leftarrow Y$
37:     $S \leftarrow I + Y^\top N X$     ▷ Compute $Y^\top N$ takes $O(n^b)$ time and $Y^\top N X$ takes $O(n^{a+b})$ time
38:     $Q \leftarrow S^{-1}$                                                  ▷ Takes $O(n^{b \cdot \omega})$ time
39: end procedure
```

Algorithm 2 Query and Reset

```
1:  data structure ProjectionMaintenanceLeft                         ▷ Theorem B.1
2:  private:
3:  procedure Query($I \subset [n], j \in [n]$)                       ▷ Lemma B.5 and Lemma B.8
4:                                          ▷ Compute query using matrix Woodbury formula
5:                          ▷ Return $(M^{-1})_{I,j} - (M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$
6:      $v_1 \leftarrow Y^\top \cdot N \cdot e_j$                        ▷ Takes $O(n^b)$ time, $v_1 \in \mathbb{R}^{n^b}$
7:      $v_2 \leftarrow Qv_1$                                        ▷ Takes $O(n^{2b})$ time
8:      $L \leftarrow (NX)_I$                          ▷ Takes $O(|I| \cdot n^a)$ time, $L \in \mathbb{R}^{|I| \times n^b}$
9:      $v_3 \leftarrow Lv_2$                                        ▷ Takes $O(|I| \cdot n^b)$ time
10:     $v \leftarrow N_{I,j} - v_3$
11:     return $v$
12: end procedure
13: private:
14: procedure Reset($X^{\text{new}} \in \mathbb{R}^{(4n+d)\times k}, Y^{\text{new}} \in \mathbb{R}^{(4n+d)\times k}$)   ▷ Lemma B.4 and Lemma B.9
15:                                       ▷ To run this procedure, we require that $k \geq n^a$
16:                                  ▷ Compute $M^{-1}$ explicitly by matrix Woodbury formula
17:                         ▷ $(M+XY^\top)^{-1} = M^{-1} - M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}$
18:                                                              ▷ Let $k = n^c$
19:     $L_1 \leftarrow (Y^{\text{new}})^\top N$                       ▷ Takes $O(n^{\omega(c,1,1)})$ time
20:     $Q \leftarrow (I + (Y^{\text{new}})^\top M^{-1} X^{\text{new}})^{-1}$   ▷ Takes $O(n^{\omega(1,1,c)})$ time
21:     $L_2 \leftarrow QL_1$                                  ▷ Takes $O(n^{\omega(c,c,1)})$ time
22:     $L_3 \leftarrow NX^{\text{new}} L_2$                           ▷ Takes $O(n^{\omega(1,1,c)})$ time
23:     $L \leftarrow N - L_3$
24:     $N \leftarrow L$
25:     $M \leftarrow M + X^{\text{new}}(Y^{\text{new}})^\top$
26:     $X \leftarrow 0, Y \leftarrow 0$
27:     $Q \leftarrow I$
28: end procedure
29: end data structure
```

B.2 Correctness

In this section, we first present the correctness proof for Update in Lemma B.3. Then we present the correctness proof for Reset in Lemma B.4. We present the correctness proof for Query in Lemma B.5.

Lemma B.3 (Update correctness)**.** *The output of* Update(u^{new}) *in Algorithm 1 satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. This follows directly from the invariant that $N = M^{-1}$. Note that by storing the quantity $(I + Y^\top M^{-1}X)^{-1}$, we can compute the query quickly, as we will show later. □

Lemma B.4 (Reset correctness)**.** *The output of* Reset($X^{\text{new}}, Y^{\text{new}}$) *in Algorithm 2 satisfies:*

$$N = M^{-1}$$

Proof.

$$\begin{aligned} N^{\text{new}} &= N - NXQY^\top N \\ &= M^{-1} - M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1} \\ &= (M + XY^\top)^{-1} \\ &= (M^{\text{new}})^{-1} \end{aligned}$$

where the first step follows from $N = M^{-1}$, the second step follows the matrix Woodbury formula, and the third step follows from Line 25 in Algorithm 1.

Algorithm 3 Restatement of Algorithm 2 and 3 in (Lee et al., 2019).

```
1: data structure ProjectionMaintenanceLeft    ▷ Theorem B.1
2: procedure UpdateOneStep($W \in \mathbb{R}^{n \times n}$)    ▷ Lemma B.10
3:     $y_i = v_i^{-1/2} w_i v_i^{-1/2} - 1, \forall i \in [n]$
4:     $r \leftarrow$ the number of indices $i$ such that $\|y_i\|_F \geq \epsilon_{mp}$
5:     if $r < n^a$ then
6:         $v^{\mathrm{new}} \leftarrow v$
7:         $l \leftarrow l + 1$
8:     else
9:         Let $\pi : [n] \longrightarrow [n]$ be a sorting permutation such that $\|y_{\pi(i)}\|_F \geq \|y_{\pi(i+1)}\|_F$
10:        while $1.5 \cdot r < n$ and $\|y_{\pi(\lceil 1.5 \cdot r \rceil)}\|_F \geq (1 - 1/\log(n))\|y_{\pi(r)}\|_F$ do
11:            $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
12:        end while
13:        $v^{\mathrm{new}}_{\pi(i)} \leftarrow \begin{cases} w_{\pi(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\pi(i)} & i \in \{r+1, \cdots, n\} \end{cases}$
14:        $\Delta \leftarrow \begin{bmatrix} (V^{\mathrm{new}})^{-1} & 0 & (V^{\mathrm{new}})^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ((V^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
15:        Let $\Delta = X^{\mathrm{new}} (Y^{\mathrm{new}})^\top$ where $X^{\mathrm{new}} \in \mathbb{R}^{(4n+d) \times k}$ and $Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d) \times k}$    ▷ $X^{\mathrm{new}}$ consists of nonzero entries, $Y^{\mathrm{new}}$ is a column selection matrix
16:        Reset($X^{\mathrm{new}}, Y^{\mathrm{new}}$)
17:        $l \leftarrow 1$
18:    end if
19:    $v \leftarrow v^{\mathrm{new}}$
20:    $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } (1 - \epsilon_{mp}) v_i \preceq w_i \preceq (1 + \epsilon_{mp}) v_i \\ w_i & \text{otherwise} \end{cases}$
21:    if $r < n^a$ then    ▷ This is put small update in UpdateOneStep
22:        Update($\widetilde{v}$)    ▷ Update the matrix with $\begin{bmatrix} (\widetilde{V})^{-1} & 0 & (\widetilde{V})^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ (\widetilde{V})^{-1/2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
23:    end if
24: end procedure
25: procedure QueryOneStep($h \in \mathbb{R}^n$)    ▷ Lemma B.12
26:        ▷ We query the rows corresponding to $R_l$
27:    $I \leftarrow$ rows corresponding to $R_l$
28:    $x \leftarrow 0_{\sqrt{n}}$
29:    for $j = 1 \to n$ do
30:        $x \leftarrow x + \mathrm{Query}(I, j) \cdot h_j$
31:    end for
32:    $y \leftarrow R_l^\top x$    ▷ $y$ is $R_l^\top R_l P h$
33:    $z \leftarrow R_l^\top R_l h - y$    ▷ $z$ is $R_l^\top R_l (I - P) h$
34:    return $(y, z)$
35: end procedure
36: end data structure
```

Algorithm 4 Restatement of Algorithm 5 in (Lee et al., 2019)

```
1: procedure CentralPathStep($\overline{x}, \overline{s}, t, \lambda, \alpha$)
2:     for $i = 1 \to n$ do
3:         $\mu_i^t \leftarrow \overline{s}_i/t + \nabla\phi_i(\overline{x}_i)$
4:         $\gamma_i^t \leftarrow \|\mu_i^t\|_{\nabla^2\phi_i(\overline{x}_i)^{-1}}$
5:         $c_i^t \leftarrow \frac{\exp(\lambda\gamma_i^t)/\gamma_i^t}{(\sum_{i=1}^n \exp(2\lambda\gamma_i^t))^{1/2}}$ if $\gamma_i^t \geq 96\sqrt{\alpha}$ and $c_i^t \leftarrow 0$ otherwise
6:         $h_i \leftarrow -\alpha \cdot c_i^t \cdot \mu_i^t$
7:     end for
8:     $\overline{W} \leftarrow (\nabla^2\phi(\overline{x}))^{-1}$
9:     return $h, \overline{W}$
10: end procedure
11:
12: procedure OneStep(mp, $x_{\mathrm{init}}, s_{\mathrm{init}}, t, \lambda, \alpha$)
13:     $h, \overline{W} \leftarrow$ CentralPathStep($x_{\mathrm{init}}, s_{\mathrm{init}}, t, \lambda, \alpha$)
14:     mp.UpdateOneStep($\overline{W}, h$)
15:     $(\overline{x}, \overline{s}) \leftarrow$ mp.QueryOneStep($h$)
16:     return $(\overline{x}, \overline{s})$
17: end procedure
```

Note at the end of Reset, N and M are updated with the new value. This completes the proof. □

Lemma B.5 (Query correctness). *The output of* Query$(I \subset [n], j \in [n])$ *in Algorithm 2 satisfies:*

$$v = (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$$

Proof. We have

$$\begin{aligned} v &= N_{I,j} - (NX)_I QY^\top N e_j \\ &= (M^{-1})_{I,j} - (M^{-1}XQY^\top M^{-1}e_j)_I \\ &= (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I \end{aligned}$$

where the first step follows from $N = M^{-1}$, and the second step follows from $Q = (I+Y^\top NX)^{-1} = (I+Y^\top M^{-1}X)^{-1}$. This completes the proof. □

B.3 Running Time

In this section, we first present the Init running time in Lemma B.6. Then we present the Update running time in Lemma B.7. We present the Query running time in Lemma B.8. We present the Reset running time in Lemma B.9.

B.3.1 Initialization time

Lemma B.6 (Initialization Time). *Taking a threshold $a \in (0, 1)$, a positive block diagonal matrix $U_0 \in \mathbb{R}^{n\times n}$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n\times n}$, a matrix $A \in \mathbb{R}^{d\times n}$ a vector $h \in \mathbb{R}^n$ and a tolerance parameter $\epsilon_{mp} \in (0, 1/4)$ as input,* Init *(Algorithm 1) operation takes $O(n^\omega)$ time to complete.*

Proof. The running time of Init (Algorithm 1) operation consists of the following component:

- $N \leftarrow M^{-1}$ takes $O(n^\omega)$ to compute the matrix inverse of $M \in \mathbb{R}^{(4n+d)\times(4n+d)}$.

Therefore, we complete the proof. □

B.3.2 Update time

Lemma B.7 (Update Time)**.** *Taking a positive block diagonal matrix $U^{\mathrm{new}} \in \mathbb{R}^{n \times n}$ as input, the* UPDATE *(Algorithm 1) operation takes $O(n^{a+b} + n^{b \cdot \omega})$ time to complete.*

Proof. The running time of UPDATE operation consists of the following components:

- $Y^\top N$ takes $O(n^b)$ time to compute the matrix multiplication between a sparse matrix $Y^\top \in \mathbb{R}^{n^b \times (4n+d)}$ where each column of Y only has one non-zero entry and matrix $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$.
- $(Y^\top N) \cdot X$ takes $O(n^{a+b}))$ time to compute the matrix multiplication between $Y^\top N \in \mathbb{R}^{n^b \times (4n+d)}$ and $X \in \mathbb{R}^{(4n+d)\times n^b}$.
- $Q \leftarrow S^{-1}$ takes $o(n^{b\omega})$ time to compute the matrix inverse of $S \in \mathbb{R}^{n^b \times n^b}$.

Therefore, we have:

$$\begin{aligned} & O(n^b) + O(n^{a+b}) + O(n^{b\omega}) \\ = & O(n^{a+b} + n^{b\omega}) \end{aligned}$$

This completes the proof. □

B.3.3 Query time

Lemma B.8 (Query Time)**.** *Taking $I \subset [n]$ and an index $j \in [n]$ as input, the* QUERY *(Algorithm 2) operation takes $O(n^{2b} + |I| \cdot n^a)$ time to complete.*

Proof. The running time of QUERY operation consists of the following components:

- $v_1 \leftarrow Y^\top N e_j$ takes $O(n^b)$ time to compute $Y^\top N e_j$ where Y only contains one non-zero entry per column and e_j only has non-zero entry on jth element.
- $v_2 \leftarrow Q v_1$ takes $O(n^{2b})$ time to compute the matrix vector multiplication between $Q \in \mathbb{R}^{n^b \times n^b}$ and $v_1 \in \mathbb{R}^{n^b}$.
- $L \leftarrow (NX)_I$ takes $O(|I| \cdot n^a)$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$ and $X \in \mathbb{R}^{(4n+d)\times n^b}$ with n^a nonzero entries, for $|I|$ rows.
- $v_3 \leftarrow L v_2$ takes $O(|I| \cdot n^b)$ time to compute the matrix vector multiplication between $L \in \mathbb{R}^{|I| \times n^b}$ and $v_2 \in \mathbb{R}^{n^b}$.

Therefore, we have:

$$\begin{aligned} & O(n^b) + O(n^{2b}) + O(|I| \cdot n^a) + O(|I| \cdot n^b) \\ = & O(n^{2b} + |I| \cdot n^a) \end{aligned}$$

This completes our proof. □

B.3.4 Reset time

Lemma B.9 (Reset Time)**.** *Let $X^{\mathrm{new}}, Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times n^c}$ be the inputs to* RESET *(Algorithm 2), then* RESET *takes $O(n^{\omega(c,1,1)})$ time to complete.*

Proof. For the simplicity of notation, we use X and Y to denote X^{new} and Y^{new}.

The running time of RESET operation consists of the following components:

- $L_1 \leftarrow Y^\top N$ takes $O(n^{\omega(c,1,1)})$ time to compute the multiplication between a $n^c \times (4n+d)$ matrix and a $(4n+d) \times (4n+d)$ matrix.

- $Q \leftarrow (I + Y^\top M^{-1} X)^{-1}$, it takes $O(n^{\omega(c,1,1)})$ time to compute the product $Y^\top M^{-1} X$ and $O(n^{c\omega})$ time to compute the inverse.
- $L_2 \leftarrow QL_1$ takes $O(n^{\omega(c,c,1)})$ time to compute the matrix multiplication between $Q \in \mathbb{R}^{n^c \times n^c}$ and $L_1 \in \mathbb{R}^{n^c \times (4n+d)}$.
- $L_3 \leftarrow NXL_2$ takes $O(n^{\omega(1,1,c)})$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$ and $X \in \mathbb{R}^{(4n+d)\times n^c}$ and $O(n^{\omega(1,c,1)})$ time to compute the matrix multiplication between $NX \in \mathbb{R}^{(4n+d)\times n^c}$ and $L_2 \in \mathbb{R}^{n^c \times (4n+d)}$. Therefore, the total time for $L_3 \leftarrow NXL_2$ is $O(n^{\omega(1,1,c)})$.

Therefore, we have:

$$\begin{aligned} & O(n^{\omega(c,1,1)}) + O(n^{\omega(c,c,1)}) + O(n^{\omega(1,1,c)}) \\ = & O(n^{\omega(c,1,1)}) \end{aligned}$$

This completes the proof. □

B.4 Update One Step

Lemma B.10 (UpdateOneStep correctness)**.** *Given an update matrix $W \in \mathbb{R}^{n\times n}$, the output of* UPDATEONESTEP *satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. When $r < n^a$, by Lemma B.3, we have $Q = (I + Y^\top M^{-1} X)^{-1}$.

When $r \geq n^a$, RESET is called. X and Y are reset as zero matrice and Q is reset as identity matrix. Therefore, we have:

$$\begin{aligned} Q = & ~ I \\ = & ~ (I + Y^\top M^{-1} X)^{-1} \end{aligned}$$

where the first step follows that Q is reset as identity matrix, and the second step follows that $Y^\top M^{-1} X = 0$.

This completes our proof.

□

Lemma B.11 (UpdateOneStep Time)**.** *Taking a vector $w \in \mathbb{R}^n$ as input, the* UPDATEONESTEP *(Algorithm 3) operation takes*

$$\begin{cases} O(n^{a+b} + n^{b\cdot\omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \geq n^a \end{cases}$$

time to complete.

Proof. When $r < n^a$, the time is dominant by UPDATE operation. By Lemma B.7, we know the time complexity is $O(n^{a+b} + n^{b\cdot\omega})$.

When $r \geq n^a$, the time is dominant by RESET operation. By Lemma B.4, we know the time complexity is $O(n^{\omega(c,1,1)})$ where $r = n^c$. By the definition of g_r in Definition A.1 and Theorem 3.1, we know the time complexity is $O(r g_r n^{2+o(1)})$.

This completes the proof. □

B.5 Query One Step

Lemma B.12 (QueryOneStep correctness)**.** *Given a query vector $h \in \mathbb{R}^n$, the output of* QUERYONESTEP *satisfies:*

$$\begin{aligned} y = & ~ R_l^\top R_l \widetilde{P} h \\ z = & ~ R_l^\top R_l (I - \widetilde{P}) h \end{aligned}$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

Proof. We have:

$$\begin{aligned} x &= \sum_{j=1}^{n} \textsc{Query}(I,j) \cdot h_j \\ &= \sum_{j=1}^{n} ((M^{-1})_{I,j} - (M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I) \cdot h_j \\ &= (M+XY^\top)_I^{-1} \cdot h \\ &= R_l \sqrt{\widetilde{U}} A^\top (A\widetilde{U}A^\top)^{-1} A \sqrt{\widetilde{U}} h \\ &= R_l \widetilde{P} h \end{aligned}$$

where the first step follows the execution of QUERYONESTEP, the second step follows from the output of QUERY in Lemma B.5, the third step comes from matrix Woodbury formula, the fourth step comes from Lemma 5.9, and the fifth step follows that $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A\widetilde{U}A^\top)^{-1} A \sqrt{\widetilde{U}}$.

Therefore, we have:

$$\begin{aligned} y &= R_l^\top x = R_l^\top R_l \widetilde{P} h \\ z &= R_l^\top R_l h - y = R_l^\top R_l (I - \widetilde{P}) h \end{aligned}$$

This completes our proof. □

Lemma B.13 (QueryOneStep Time). *Given a query vector $h \in \mathbb{R}^n$ as input, the* QUERYONESTEP *(Algorithm 3) operation takes $O(n^{3/2} + n^{2b+1} + n^{3/2+a})$ time to complete.*

Proof. The QUERYONESTEP operation contains three steps:

- It takes $O(n^{2b+1} + n^{3/2+a})$ time to call QUERY operation for n times.
- It takes $O(n^{3/2})$ time to compute the multiplication between $R_l^\top \in \mathbb{R}^{n \times \sqrt{n}}$ and $x \in \mathbb{R}^{\sqrt{n}}$.
- It takes $O(n^{3/2})$ time to compute the multiplication between $R_l \in \mathbb{R}^{\sqrt{n} \times n}$ and $h \in \mathbb{R}^n$ and $O(n^{3/2})$ time to compute the multiplication between $R_l^\top \in \mathbb{R}^{n \times \sqrt{n}}$ and $R_l x \in \mathbb{R}^{\sqrt{n}}$.

Therefore, we have the overall time complexity of QUERYONESTEP operation:

$$\begin{aligned} & O(n^{2b+1} + n^{3/2+a}) + O(n^{3/2}) + O(n^{3/2}) + O(n^{3/2}) \\ = & O(n^{3/2} + n^{2b+1} + n^{3/2+a}) \end{aligned}$$

This completes the proof. □

Remark B.14. *In the query phase, we have to query a $\sqrt{n} \times n$ submatrix, which takes $O(n^{1.5+a})$ time per iteration. Sum over $\sqrt{n}$ iterations, we need to pay $O(n^{2+a})$ time in total. After balancing, we observe that $a = 1/3$, and this algorithm is slower than (Lee et al., 2019).*

C PROJECTION MAINTENANCE VIA INVERSE MAINTENANCE: ONE LAYER AND RIGHT SKETCH WITHOUT VECTOR

We show how to design M so that its inverse contains the right sketched projection $\sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U}\mathsf{R}^\top$.

We present the corresponding initialization, update, reset and query functions in Algorithm 5, Algorithm 6, Algorithm 7 and Algorithm 8. The INITIALIZE operation construct a right sketching matrix M from Lemma 5.10 and compute the inverse $N = M^{-1}$. The UPDATE operation updates the inverse matrix such that $Q = (I + Y^\top M^{-1} X)^{-1}$. The RESET operation reset the value of N such that $N = M^{-1}$. QUERY operation takes $I \subset [n]$ and an index $j \in [n]$ as input and computes v such that

$$v = (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I.$$

UPDATEONESTEP operation takes an update vector $w \in \mathbb{R}^n$ and decides to call either UPDATE or RESET to maintain the inverse of the matrix. QUERYONESTEP calls QUERY operation for n times and obtain the projected sketching result y and z such that:

$$y = \widetilde{P} R_l^\top R_l h$$
$$z = (I - \widetilde{P}) R_l^\top R_l h$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

C.1 Main Result

In this section, we show how to implement the projection matrix maintenance task as an inverse maintenance with the sketch on the right problem.

Theorem C.1. *Given a matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ where $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \cdots & R_T^\top \end{bmatrix}$. Let $T = \sqrt{n}$. Let $m = 4n + d$.*

Let $A \in \mathbb{R}^{d \times n}$ of rank d and let $U \in \mathbb{R}^{n \times n}$ be a diagonal matrix with non-zero diagonal entries, $v \in \mathbb{R}^n$ and $\mathsf{R} \in \mathbb{R}^{n \times n}$ be a sketch matrix. There exists a data structure PROJECTIONMAINTENANCERIGHT *(Algorithm 5 and 6) which supports the following operations:*

- **public:**INIT$(a \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n \times n}, A \in \mathbb{R}^{d \times n})$*: Given a threshold $a \in (0,1)$, a vector $u_0 \in \mathbb{R}^n$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ and a matrix $A \in \mathbb{R}^{d \times n}$ as input,* INIT *(Algorithm 5) operation runs in $O(n^\omega)$ time.*
- **public:**UPDATEONESTEP$(w \in \mathbb{R}^n)$*. Given an update vector $w \in \mathbb{R}^n$, the output of* UPDATEONESTEP*(Algorithm 7) satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

  With the definition of g_r in Definition A.1, the time complexity of UPDATEONESTEP *is:*

$$\begin{cases} O(n^{a+b} + n^{b \cdot \omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \geq n^a \end{cases}$$

- **public:**QUERYONESTEP$(h \in \mathbb{R}^n)$*. Given a query vector $h \in \mathbb{R}^n$, the output of* QUERYONESTEP*(Algorithm 7) satisfies:*

$$y = \widetilde{P} R_l^\top R_l h$$
$$z = (I - \widetilde{P}) R_l^\top R_l h$$

  where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP*. And the time complexity of* QUERYONESTEP *is $O(n^{3/2} + n^{2b+1/2} + n^{3/2+a})$.*
- **private:**UPDATE$(u^{\text{new}} \in \mathbb{R}^n)$*: Given a vector $u^{\text{new}} \in \mathbb{R}^n$ as input, the* UPDATE *(Algorithm 5) operation runs $O(n^{a\omega})$ time.*
- **private:**QUERY$(I \subset [n], j \in [n])$*: Given $I \subset [n]$ and an index $j \in [n]$ as input, the* QUERY *(Algorithm 6) operation runs in $O(n^{2b} + |I| \cdot n^a)$ time.*
- **private:**RESET$(X^{\text{new}} \in \mathbb{R}^{(4n+d) \times n^c}, Y^{\text{new}} \in \mathbb{R}^{(4n+d) \times n^c})$*:* RESET *(Algorithm 6) operations runs in $O(n^{\omega(c,1,1)})$ time.*

Proof. The correctness follows from combining Lemma C.2, Lemma C.3 and Lemma C.4.

The running time follows from combining Lemma C.5, Lemma C.6, Lemma C.7 and Lemma C.8. □

Algorithm 5 1-layer projection maintenance by inverse maintenance with Sketching on the right. Note that we store the inverse of M, but only update it during the procedure RESET

```
1:  data structure PROJECTIONMAINTENANCERIGHT                        ▷ Theorem C.1
2:  members
3:      $u \in \mathbb{R}^n$
4:      $U \in \mathbb{R}^{n \times n}$
5:      $A \in \mathbb{R}^{d \times n}$
6:      $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \dots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n \times n}$
7:      $M \in \mathbb{R}^{(4n+d) \times (4n+d)}$
8:      $a \in (0,1)$                                                    ▷ Threshold 1
9:      $X, Y \in \mathbb{R}^{(4n+d) \times 3n^a}$          ▷ Each column of $Y$ has only one nonzero entry
10:     $Q \in \mathbb{R}^{3n^a \times 3n^a}$
11:     $N \in \mathbb{R}^{(4n+d) \times (4n+d)}$                             ▷ Inverse of $M$
12: end members
13:
14: procedure INIT($a \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n \times n}, A \in \mathbb{R}^{d \times n}$)      ▷ Lemma C.5
15:     $a \leftarrow a, u \leftarrow u_0, A \leftarrow A$
16:     $U \leftarrow \mathrm{diag}(u)$
17:     $\mathsf{R} \leftarrow \mathsf{R}$
18:     $M \leftarrow \begin{bmatrix} U^{-1} & A^\top & U^{-1/2} & 0 & 0 \\ A & 0 & 0 & 0 & 0 \\ 0 & 0 & -I & 0 & \mathsf{R}^\top \\ (U^{-1/2})^\top & 0 & 0 & -I & \mathsf{R}^\top \\ 0 & 0 & 0 & 0 & I \end{bmatrix}$    ▷ Right sketching matrix construction from Lemma 5.10
19:     $X, Y \leftarrow \mathbf{0}_{(4n+d) \times 3n^a}$
20:     $N \leftarrow M^{-1}$                                        ▷ Takes $O(n^\omega)$ time
21:     $Q \leftarrow \mathbf{0}_{3n^a \times 3n^a}$
22: end procedure
23:
24: procedure UPDATE($u^{\mathrm{new}} \in \mathbb{R}^n$)                  ▷ Lemma C.2 and Lemma C.6
25:                                             ▷ $u^{\mathrm{new}}$ has at most $n^a$ non-zero entries
26:     $\Delta \leftarrow \begin{bmatrix} (U^{\mathrm{new}})^{-1} & 0 & (U^{\mathrm{new}})^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ((U^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
27:     Let $\Delta = XY^\top$ where $X \in \mathbb{R}^{(4n+d) \times 3n^a}$ and $Y \in \mathbb{R}^{(4n+d) \times 3n^a}$   ▷ $X$ consists of nonzero entries, $Y$ is a column selection matrix
28:                                                  ▷ Each row of $Y$ has only 1 nonzero entry
29:     $X \leftarrow X, Y \leftarrow Y$
30:     $S \leftarrow I + Y^\top N X$      ▷ Compute $Y^\top N$ takes $O(n^a)$ time and $Y^\top N X$ takes $O(n^{2a})$ time
31:     $Q \leftarrow S^{-1}$                                        ▷ Takes $O(n^{a\omega})$ time
32: end procedure
```

C.2 Correctness

In this section, we first present the correctness proof for UPDATE in Lemma C.2. Then we present the correctness proof for RESET in Lemma C.3. We present the correctness proof for QUERY in Lemma C.4.

Lemma C.2 (Update correctness)**.** *The output of* UPDATE(u^{new}) *in Algorithm 5 satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. This follows directly from the invariant that $N = M^{-1}$. Note that by storing the quantity $(I + Y^\top M^{-1} X)^{-1}$, we can compute the query quickly, as we will show later. □

Algorithm 6 Query and Reset

```
1:  data structure ProjectionMaintenanceRight                    ▷ Theorem C.1
2:  procedure Query($I \subset [n], j \in [n]$)                  ▷ Lemma C.4 and Lemma C.7
3:                                                               ▷ Compute query using matrix Woodbury formula
4:                                                               ▷ Return $(M^{-1})_{I,j} - (M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$
5:      $v_1 \leftarrow Y^\top \cdot N \cdot e_j$                ▷ Takes $O(n^b)$ time, $v_1 \in \mathbb{R}^{n^b}$
6:      $v_2 \leftarrow Qv_1$                                    ▷ Takes $O(n^{2b})$ time
7:      $L \leftarrow (NX)_I$                                    ▷ Takes $O(|I| \cdot n^a)$ time, $L \in \mathbb{R}^{|I| \times n^b}$
8:      $v_3 \leftarrow Lv_2$                                    ▷ Takes $O(|I| \cdot n^b)$ time
9:      $v \leftarrow N_{I,j} - v_3$
10:     return $v$
11: end procedure
12:
13: procedure Reset($X^{\mathrm{new}} \in \mathbb{R}^{(4n+d) \times k}, Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d) \times k}$)   ▷ Lemma C.3 and Lemma C.8
14:                                                              ▷ To run this procedure, we require that $k \geq n^a$
15:                                                              ▷ Compute $M^{-1}$ explicitly by matrix Woodbury formula
16:                                                              ▷ $(M+XY^\top)^{-1} = M^{-1} - M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}$
17:                                                              ▷ Let $k = n^c$
18:     $L_1 \leftarrow (Y^{\mathrm{new}})^\top N$               ▷ Takes $O(n^{\omega(c,1,1)})$ time
19:     $Q \leftarrow (I + (Y^{\mathrm{new}})^\top M^{-1} X^{\mathrm{new}})^{-1}$   ▷ Takes $O(n^{\omega(1,1,c)})$ time
20:     $L_2 \leftarrow QL_1$                                    ▷ Takes $O(n^{\omega(c,c,1)})$ time
21:     $L_3 \leftarrow NX^{\mathrm{new}}L_2$                    ▷ Takes $O(n^{\omega(1,1,c)})$ time
22:     $L \leftarrow N - L_3$
23:     $N \leftarrow L$
24:     $M \leftarrow M + X^{\mathrm{new}}(Y^{\mathrm{new}})^\top$
25:     $X \leftarrow 0, Y \leftarrow 0$
26:     $Q \leftarrow I$
27: end procedure
28: end data structure
```

Lemma C.3 (Reset correctness)**.** *The output of* RESET$(X^{\mathrm{new}}, Y^{\mathrm{new}})$ *in Algorithm 6 satisfies:*

$$N = M^{-1}$$

Proof.

$$\begin{aligned} N^{\mathrm{new}} &= N - NXQY^\top N \\ &= M^{-1} - M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1} \\ &= (M+XY^\top)^{-1} \\ &= (M^{\mathrm{new}})^{-1} \end{aligned}$$

where the first step follows from $N = M^{-1}$, the second step follows the matrix Woodbury formula, and the third step follows from the UPDATE in Algorithm 5.

Note at the end of RESET, N and M are updated with the new value. This completes the proof. □

Lemma C.4 (Query correctness)**.** *The output of* QUERY$(I \subset [n], j \in [n])$ *in Algorithm 6 satisfies:*

$$v = (M^{-1})_{I,j} - (M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$$

Proof. We have

$$\begin{aligned} v &= N_{I,j} - (NX)_I QY^\top Ne_j \\ &= (M^{-1})_{I,j} - (M^{-1}XQY^\top M^{-1}e_j)_I \end{aligned}$$

Algorithm 7 Restatement of Algorithm 9 and 10 in (Song and Yu, 2021).

```
1: data structure ProjectionMaintenanceRight                                  ▷ Theorem C.1
2: procedure UpdateOneStep($W \in \mathbb{R}^{n \times n}$)                          ▷ Lemma C.9
3:     $y_i = v_i^{-1/2} w_i v_i^{-1/2} - 1, \forall i \in [n]$
4:     $r \leftarrow$ the number of indices $i$ such that $|y_i| \geq \epsilon_{mp}/2$
5:     if $r < n^a$ then
6:         $v^{\mathrm{new}} \leftarrow v$
7:         $l \leftarrow l + 1$
8:     else
9:         Let $\pi : [n] \longrightarrow [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
10:        while $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r \rceil)}| \geq (1 - 1/\log(n))|y_{\pi(r)}|$ do
11:            $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
12:        end while
13:        $v_{\pi(i)}^{\mathrm{new}} \leftarrow \begin{cases} w_{\pi(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\pi(i)} & i \in \{r+1, \cdots, n\} \end{cases}$
14:        $\Delta \leftarrow \begin{bmatrix} (V^{\mathrm{new}})^{-1} & 0 & (V^{\mathrm{new}})^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ((V^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
15:        Let $\Delta = X^{\mathrm{new}} (Y^{\mathrm{new}})^\top$ where $X^{\mathrm{new}} \in \mathbb{R}^{(4n+d) \times k}$ and $Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d) \times k}$      ▷ $X^{\mathrm{new}}$ consists of nonzero entries, $Y^{\mathrm{new}}$ is a column selection matrix
16:        Reset($X^{\mathrm{new}}, Y^{\mathrm{new}}$)
17:        $l \leftarrow 1$
18:    end if
19:    $v \leftarrow v^{\mathrm{new}}$
20:    $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } |\ln w_i - \ln v_i| < \epsilon_{\mathrm{mp}}/2 \\ w_i & \text{otherwise} \end{cases}$
21:    if $r < n^a$ then                                   ▷ This is put small update in UpdateOneStep
22:        Update($\widetilde{v}$)      ▷ Update the matrix with $\begin{bmatrix} (\widetilde{V})^{-1} & 0 & (\widetilde{V})^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ (\widetilde{V})^{-1/2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
23:    end if
24:    return $\widetilde{v}$
25: end procedure
26: procedure QueryOneStep($h \in \mathbb{R}^n$)                                    ▷ Lemma C.11
27:                                      ▷ We query the columns corresponding to $R_l$
28:    $I \leftarrow \{3n + d + 1, \dots, 4n + d\}$
29:    $x \leftarrow R_l h$
30:    $y \leftarrow 0_n$
31:    $J \leftarrow$ columns corresponding to $R_l^\top$                                ▷ $|J| = O(\sqrt{n})$
32:    for $j \in J$ do
33:        $y \leftarrow y + \mathrm{Query}(I, j) \cdot x_j$
34:    end for
35:                                                                      ▷ $y$ is $P R_l^\top R_l h$
36:    $z \leftarrow R_l^\top R_l h - y$                                             ▷ $z$ is $(I - P) R_l^\top R_l h$
37:    return $(y, z)$
38: end procedure
39: end data structure
```

Algorithm 8 Restatement of Algorithm 7 in (Song and Yu, 2021)

```
1:  procedure OneStep($\mathrm{mp}, x, s, \delta_\mu, b_{\mathrm{sketch}}, \epsilon$)
2:      $w \leftarrow \frac{x}{s}$, $\widetilde{v} \leftarrow \mathrm{mp}.$UpdateOneStep$(w)$
3:      $\overline{x} \leftarrow x\sqrt{\frac{\widetilde{v}}{w}}$, $\overline{s} \leftarrow s \cdot \sqrt{\frac{w}{\widetilde{v}}}$
4:      repeat
5:          $p_x, p_s \leftarrow \mathrm{mp}.$QueryOneStep$(\frac{1}{\sqrt{XS}}\delta_\mu)$
6:          $\widetilde{\delta}_s \leftarrow \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}} p_s$
7:          $\widetilde{\delta}_x \leftarrow \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}} p_x$
8:      until $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{1}{100\log n}$ and $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{1}{100\log n}$
9:      return $(x + \widetilde{\delta}_x, s + \widetilde{\delta}_s)$
10: end procedure
```

$$= (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$$

where the first step follows from $N = M^{-1}$, and the second step follows from $Q = (I+Y^\top NX)^{-1} = (I+Y^\top M^{-1}X)^{-1}$.

This completes the proof. □

C.3 Running Time

In this section, we first present the Init running time in Lemma C.5. Then we present the Update running time in Lemma C.6. We present the Query running time in Lemma C.7. We present the Reset running time in Lemma C.8.

C.3.1 Initialization time

Lemma C.5 (Initialization Time)**.** *Taking a threshold $a \in (0, 1)$, a vector $u_0 \in \mathbb{R}^n$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n\times n}$ and a matrix $A \in \mathbb{R}^{d\times n}$ as input,* Init *(Algorithm 5) operation takes $O(n^\omega)$ time to complete.*

Proof. The running time of Init (Algorithm 5) operation consists of the following component:

- $N \leftarrow M^{-1}$ takes $O(n^\omega)$ to compute the matrix inverse of $M \in \mathbb{R}^{(4n+d)\times(4n+d)}$.

Therefore, we complete the proof. □

C.3.2 Update time

Lemma C.6 (Update Time)**.** *Taking a vector $u^{\mathrm{new}} \in \mathbb{R}^n$ as input, the* Update *(Algorithm 5) operation takes $O(n^{a\omega})$ time to complete.*

Proof. The running time of Update operation consists of the following components:

- $Y^\top N$ takes $O(n^a)$ time to compute the matrix multiplication between a sparse matrix $Y^\top \in \mathbb{R}^{3n^a\times(4n+d)}$ where each column of Y only has one non-zero entry and matrix $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$.
- $(Y^\top N) \cdot X$ takes $O(n^{2a}))$ time to compute the matrix multiplication between $Y^\top N \in \mathbb{R}^{3n^a\times(4n+d)}$ and $X \in \mathbb{R}^{(4n+d)\times 3n^a}$.
- $Q \leftarrow S^{-1}$ takes $o(n^{a\omega})$ time to compute the matrix inverse of $S \in \mathbb{R}^{3n^a\times 3n^a}$.

Therefore, we have:

$$\begin{aligned} & O(n^a) + O(n^{2a}) + O(n^{a\omega}) \\ = & O(n^{a\omega}) \end{aligned}$$

This completes the proof. □

C.3.3 Query time

Lemma C.7 (Query Time)**.** *Taking $I \subset [n]$ and an index $j \in [n]$ as input, the* QUERY *(Algorithm 6) operation takes $O(n^{2b} + |I| \cdot n^a)$ time to complete.*

Proof. The running time of QUERY operation consists of the following components:

- $v_1 \leftarrow Y^\top N e_j$ takes $O(n^b)$ time to compute $Y^\top N e_j$ where Y only contains one non-zero entry per column and e_j only has non-zero entry on jth element.
- $v_2 \leftarrow Q v_1$ takes $O(n^{2b})$ time to compute the matrix vector multiplication between $Q \in \mathbb{R}^{n^b \times n^b}$ and $v_1 \in \mathbb{R}^{n^b}$.
- $L \leftarrow (NX)_I$ takes $O(|I| \cdot n^a)$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$ and $X \in \mathbb{R}^{(4n+d)\times n^b}$ with n^a nonzero entries, for $|I|$ rows.
- $v_3 \leftarrow L v_2$ takes $O(|I| \cdot n^b)$ time to compute the matrix vector multiplication between $L \in \mathbb{R}^{|I| \times n^b}$ and $v_2 \in \mathbb{R}^{n^b}$.

Therefore, we have:

$$\begin{aligned} & O(n^b) + O(n^{2b}) + O(|I| \cdot n^a) + O(|I| \cdot n^b) \\ = & O(n^{2b} + |I| \cdot n^a) \end{aligned}$$

This completes our proof. □

C.3.4 Reset time

Lemma C.8 (Reset Time)**.** *Let $X^{\mathrm{new}}, Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times n^c}$ be the inputs to* RESET *(Algorithm 6), then* RESET *takes $O(n^{\omega(c,1,1)})$ time to complete.*

Proof. For the simplicity of notation, we use X and Y to denote X^{new} and Y^{new}.

The running time of RESET operation consists of the following components:

- $L_1 \leftarrow Y^\top N$ takes $O(n^{\omega(c,1,1)})$ time to compute the multiplication between a $n^c \times (4n+d)$ matrix and a $(4n+d) \times (4n+d)$ matrix.
- $Q \leftarrow (I + Y^\top M^{-1} X)^{-1}$, it takes $O(n^{\omega(c,1,1)})$ time to compute the product $Y^\top M^{-1} X$ and $O(n^{c\omega})$ time to compute the inverse.
- $L_2 \leftarrow Q L_1$ takes $O(n^{\omega(c,c,1)})$ time to compute the matrix multiplication between $Q \in \mathbb{R}^{n^c \times n^c}$ and $L_1 \in \mathbb{R}^{n^c \times (4n+d)}$.
- $L_3 \leftarrow N X L_2$ takes $O(n^{\omega(1,1,c)})$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$ and $X \in \mathbb{R}^{(4n+d)\times n^c}$ and $O(n^{\omega(1,c,1)})$ time to compute the matrix multiplication between $NX \in \mathbb{R}^{(4n+d)\times n^c}$ and $L_2 \in \mathbb{R}^{n^c \times (4n+d)}$. Therefore, the total time for $L_3 \leftarrow N X L_2$ is $O(n^{\omega(1,1,c)})$.

Therefore, we have:

$$\begin{aligned} & O(n^{\omega(c,1,1)}) + O(n^{\omega(c,c,1)}) + O(n^{\omega(1,1,c)}) \\ = & O(n^{\omega(c,1,1)}) \end{aligned}$$

This completes the proof. □

C.4 Update One Step

Lemma C.9 (UpdateOneStep correctness)**.** *Given an update vector $w \in \mathbb{R}^n$, the output of* UPDATEONESTEP *satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. When $r < n^a$, by Lemma C.2, we have $Q = (I + Y^\top M^{-1} X)^{-1}$.

When $r \geq n^a$, RESET is called. X and Y are reset as zero matrice and Q is reset as identity matrix. Therefore, we have:

$$\begin{aligned} Q &= I \\ &= (I + Y^\top M^{-1} X)^{-1} \end{aligned}$$

where the first step follows that Q is reset as identity matrix, and the second step follows that $Y^\top M^{-1} X = 0$.

This completes our proof.

□

Lemma C.10 (UpdateOneStep Time)**.** *Taking a vector $w \in \mathbb{R}^n$ as input, the* UPDATEONESTEP *(Algorithm 7) operation takes*

$$\begin{cases} O(n^{a+b} + n^{b \cdot \omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \geq n^a \end{cases}$$

time to complete.

Proof. When $r < n^a$, the time is dominant by UPDATE operation. By Lemma C.6, we know the time complexity is $O(n^{a+b} + n^{b \cdot \omega})$.

When $r \geq n^a$, the time is dominant by RESET operation. By Lemma C.3, we know the time complexity is $O(n^{\omega(c,1,1)})$ where $r = n^c$. By the definition of g_r in Definition A.1 and Theorem 3.1, we know the time complexity is $O(r g_r n^{2+o(1)})$.

This completes the proof. □

C.5 Query One Step

Lemma C.11 (QueryOneStep correctness)**.** *Given a query vector $h \in \mathbb{R}^n$, the output of* QUERYONESTEP *satisfies:*

$$\begin{aligned} y &= \widetilde{P} R_l^\top R_l h \\ z &= (I - \widetilde{P}) R_l^\top R_l h \end{aligned}$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

Proof. We have:

$$\begin{aligned} y &= \sum_{j=1}^n \mathrm{QUERY}(I, j) \cdot (R_l h)_j \\ &= \sum_{j=1}^n ((M^{-1})_{I,j} - (M^{-1} X (I + Y^\top M^{-1} X)^{-1} Y^\top M^{-1} e_j)_I) \cdot (R_l h)_j \\ &= (M + XY^\top)_I^{-1} \cdot R_l h \\ &= \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}} R_l^\top R_l h \\ &= \widetilde{P} R_l^\top R_l h \end{aligned}$$

where the first step follows the execution of QUERYONESTEP, the second step follows from the output of QUERY in Lemma C.4, the third step comes from matrix Woodbury formula, the fourth step comes from Lemma 5.10, and the fifth step follows that $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$.

Therefore, we have:

$$z = R_l^\top R_l h - y = (I - \widetilde{P}) R_l^\top R_l h$$

This completes our proof. □

Lemma C.12 (QueryOneStep Time). *Given a query vector $h \in \mathbb{R}^n$ as input, the* QUERYONESTEP *(Algorithm 7) operation takes $O(n^{3/2} + n^{2b+1/2} + \cdot n^{3/2+a})$ time to complete.*

Proof. The QUERYONESTEP operation contains three steps:

- It takes $O(n^{3/2})$ time to compute the matrix multiplication $x \leftarrow R_l h$.
- It takes $O(n^{2b+1/2} + |I| \cdot n^{1/2+a})$ time to call QUERY for $|J| = O(\sqrt{n})$ times.
- It takes $O(n^{3/2})$ time to compute the multiplication between $R_l \in \mathbb{R}^{\sqrt{n} \times n}$ and $h \in \mathbb{R}^n$ and $O(n^{3/2})$ time to compute the multiplication between $R_l^\top \in \mathbb{R}^{n \times \sqrt{n}}$ and $R_l h \in \mathbb{R}^{\sqrt{n}}$.

Therefore, we have the overall time complexity of QUERYONESTEP operation:

$$\begin{aligned} & O(n^{3/2}) + O(n^{2b+1/2} + |I| \cdot n^{1/2+a}) + O(n^{3/2}) + O(n^{3/2}) \\ = & O(n^{3/2} + n^{2b+1/2} + |I| \cdot n^{1/2+a}) \\ = & O(n^{3/2} + n^{2b+1/2} + \cdot n^{3/2+a}) \end{aligned}$$

where the last step follows that $|I| = n$. This completes the proof. □

Remark C.13. *Unlike sketching-on-the-left, here we need to query a submatrix of size $n \times \sqrt{n}$. This yields a similar running time as sketching-on-the-left which is slower than (Cohen et al., 2019b; Song and Yu, 2021).*

D SKETCH DATA STRUCTURE WITH VECTOR, ON LEFT

In this section, we show that we can directly store the approximate projection vector product inside M, i.e., M^{-1} contains $\mathsf{R}^\top \sqrt{U} A^\top (AUA^\top)^{-1} A \sqrt{U} h$.

We present the corresponding initialization, update, reset and query functions in Algorithm 9, Algorithm 10, Algorithm 11 and Algorithm 12. Similar to Section B, The INITIALIZE operation construct a right sketching matrix M from Lemma 5.9 with the query vector h embedded inside, and compute the inverse $N = M^{-1}$. The UPDATE operation updates the inverse matrix such that $Q = (I + Y^\top M^{-1} X)^{-1}$. The RESET operation reset the value of N such that $N = M^{-1}$. QUERY operation takes $I \subset [n]$ and an index $j \in [n]$ as input and computes v such that

$$v = (M^{-1})_{I,j} - (M^{-1} X (I + Y^\top M^{-1} X)^{-1} Y^\top M^{-1} e_j)_I.$$

UPDATEONESTEP operation takes an update vector $w \in \mathbb{R}^n$ and decides to call either UPDATE or RESET to maintain the inverse of the matrix. QUERYONESTEP calls QUERY operation for n times and obtain the projected sketching result y and z such that:

$$y = R_l^\top R_l \widetilde{P} h$$
$$z = R_l^\top R_l (I - \widetilde{P}) h$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

Theorem D.1 (Sketching on left with vector)**.** *Given a matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ where $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \cdots & R_T^\top \end{bmatrix}$. Let $T = \sqrt{n}$. Let $m = 4n + d$.*

Let $A \in \mathbb{R}^{d \times n}$ of rank d and let $U \in \mathbb{R}^{n \times n}$ be a diagonal matrix with non-zero diagonal entries, $v \in \mathbb{R}^n$ and $\mathsf{R} \in \mathbb{R}^{n \times n}$ be a sketch matrix. There exists a data structure PROJECTIONMAINTENANCELEFTWITHVECTOR *(Algorithm 9 and 10) which supports the following operations:*

- INIT$(a \in (0,1), b \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n \times n}, A \in \mathbb{R}^{d \times n})$*: Given thresholds $a, b \in (0,1)$ with $b \leq a$, a vector $u_0 \in \mathbb{R}^n$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ and a matrix $A \in \mathbb{R}^{d \times n}$ as input,* INIT *(Algorithm 9) operation runs in $O(n^\omega)$ time.*
- UPDATE$(u^{\mathrm{new}} \in \mathbb{R}^n, h^{\mathrm{new}} \in \mathbb{R}^{\mathrm{new}})$*: Given a vector $u^{\mathrm{new}} \in \mathbb{R}^n$ as input, the* UPDATE *(Algorithm 9) operation runs $O(n^{a+b} + n^{b \cdot \omega})$ time.*
- QUERY$(I \subset [n], j \in [n])$*: Given $I \subset [n]$ and an index $j \in [n]$ as input, the* QUERY *(Algorithm 10) operation runs in $O(n^{2b} + |I| \cdot n^a)$ time.*
- UPDATEONESTEP$(W \in \mathbb{R}^{n \times n}, h \in \mathbb{R}^n)$*. Given an positive block diagonal matrix $W \in \mathbb{R}^n$,* UPDATEONESTEP*(Algorithm 11) updates a block diagonal matrix $\widetilde{V} \in \mathbb{R}^{n \times n}$*

  $$(1 - \epsilon_{mp}) \|w_i\|_F \preceq \|\widetilde{v}_i\|_F \preceq (1 + \epsilon_{mp}) \|w_i\|_F,$$

  where $w_i, \widetilde{v}_i$ denote the i-th block of $W, \widetilde{V}$ respectively. With the definition of g_r in Definition A.1, the time complexity of UPDATEONESTEP *is:*

  $$\begin{cases} O(n^{a+b} + n^{b \cdot \omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \geq n^a \end{cases}$$

- QUERYONESTEP$(h \in \mathbb{R}^n)$*. Given a query vector $h \in \mathbb{R}^n$, the output of* QUERYONESTEP*(Algorithm 11) satisfies:*

  $$y = R_l^\top R_l \widetilde{P} h$$
  $$z = R_l^\top R_l (I - \widetilde{P}) h$$

  where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP*. And the time complexity of* QUERYONESTEP *is $O(n^{1.5} + n^{2b})$.*

- RESET$(X^{\mathrm{new}}, Y^{\mathrm{new}})$*: Give matrices* $X^{\mathrm{new}}, Y^{\mathrm{new}} \in \mathbb{R}^{n \times n^c}$*,* RESET *(Algorithm 10) operations runs in* $O(n^{\omega(c,1,1)})$ *time.*

Proof. The proof follows Lemma D.2, Lemma D.3, Lemma D.4, Lemma D.5, Lemma D.6, Lemma D.7, Lemma D.8, Lemma D.9 and Lemma D.10.

□

D.1 Correctness

In this section, we first present the correctness proof for UPDATE in Lemma D.2. We present the correctness proof for QUERY in Lemma D.3. Then we present the correctness proof for RESET in Lemma D.4.

Lemma D.2 (Update correctness)**.** *The output of* UPDATE$(u^{\mathrm{new}} \in \mathbb{R}^n, h \in \mathbb{R}^n)$ *in Algorithm 9 satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. This follows directly from the invariant that $N = M^{-1}$. Note that by storing the quantity $(I + Y^\top M^{-1} X)^{-1}$, we can compute the query quickly, as we will show later. □

Lemma D.3 (Query correctness)**.** *The output of* QUERY$(I \subset [n], j \in [n])$ *in Algorithm 10 satisfies:*

$$v = (M^{-1})_{I,j} - (M^{-1} X (I + Y^\top M^{-1} X)^{-1} Y^\top M^{-1} e_j)_I$$

Proof. We have

$$\begin{aligned} v &= N_{I,j} - (NX)_I Q Y^\top N e_j \\ &= (M^{-1})_{I,j} - (M^{-1} X Q Y^\top M^{-1} e_j)_I \\ &= (M^{-1})_{I,j} - (M^{-1} X (I + Y^\top M^{-1} X)^{-1} Y^\top M^{-1} e_j)_I \end{aligned}$$

where the first step follows from $N = M^{-1}$, and the second step follows from $Q = (I + Y^\top N X)^{-1} = (I + Y^\top M^{-1} X)^{-1}$.

This completes the proof. □

Lemma D.4 (Reset correctness)**.** *The output of* RESET$(X^{\mathrm{new}}, Y^{\mathrm{new}})$ *in Algorithm 10 satisfies:*

$$N = M^{-1}$$

Proof.

$$\begin{aligned} N^{\mathrm{new}} &= N - N X Q Y^\top N \\ &= M^{-1} - M^{-1} X (I + Y^\top M^{-1} X)^{-1} Y^\top M^{-1} \\ &= (M + X Y^\top)^{-1} \\ &= (M^{\mathrm{new}})^{-1} \end{aligned}$$

where the first step follows from $N = M^{-1}$, the second step follows the matrix Woodbury formula, and the third step follows from UPDATE in Algorithm 5.

Note at the end of RESET, N and M are updated with the new value. This completes the proof. □

D.2 Running Time

In this section, we first present the INIT running time in Lemma D.5. Then we present the UPDATE running time in Lemma D.6. We present the QUERY running time in Lemma D.7. We present the RESET running time in Lemma D.8.

We remark that we will store the dense vector h into matrix M, hence, the threshold parameter a becomes 1. This yields a slower update time of $O(n^{1+b})$ and a query time of $O(n^{1.5})$. This is enough to reproduce the result in (Lee et al., 2019).

Algorithm 9 1-layer projection maintenance by inverse maintenance with Sketching on the left. Note that we store the inverse of M, but only update it during the procedure RESET. For UPDATE, we assume the change happens for at most n^a entries in at most n^b columns.

```
1:  data structure PROJECTIONMAINTENANCELEFTWITHVECTOR    ▷ Theorem D.1
2:  members
3:      $U \in \mathbb{R}^{n \times n}$
4:      $A \in \mathbb{R}^{d \times n}$
5:      $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \dots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n \times n}$
6:      $M \in \mathbb{R}^{(4n+d)\times(4n+d)}$
7:      $a \in (0,1)$    ▷ Threshold 1
8:      $b \in (0,1)$    ▷ Threshold 2
9:      $X, Y \in \mathbb{R}^{(4n+d)\times n^b}$    ▷ Each column of $Y$ has only one nonzero entry
10:     $Q \in \mathbb{R}^{n^b \times n^b}$
11:     $N \in \mathbb{R}^{(4n+d)\times(4n+d)}$    ▷ Inverse of $M$
12: end members
13:
14: procedure INIT($a, b \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n\times n}, A \in \mathbb{R}^{d\times n}, h \in \mathbb{R}^n$)    ▷ Lemma D.5
15:     $a \leftarrow a, b \leftarrow b, A \leftarrow A$
16:     $U \leftarrow \mathrm{diag}(u)$
17:     $\mathsf{R} \leftarrow \mathsf{R}$
18:     $M \leftarrow \begin{bmatrix} U^{-1} & A^\top & U^{-1/2} & 0 & 0 & 0 \\ A & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -I & 0 & 0 & h \\ (U^{-1/2})^\top & 0 & 0 & -I & 0 & h \\ 0 & 0 & 0 & \mathsf{R} & I & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$    ▷ Left sketching matrix construction from Lemma 5.9
19:     $X, Y \leftarrow \mathbf{0}_{(4n+d)\times n^b}$
20:     $N \leftarrow M^{-1}$    ▷ Takes $O(n^\omega)$ time
21:     $Q \leftarrow \mathbf{0}_{n^b \times n^b}$
22: end procedure
23:
24: procedure UPDATE($u^{\mathrm{new}} \in \mathbb{R}^n, h^{\mathrm{new}} \in \mathbb{R}^n$)    ▷ Lemma D.2 and Lemma D.6
25:         ▷ To run this procedure, we require that $u^{\mathrm{new}}$ has at most $n^b$ non-zero entries
26:     if $\|u^{\mathrm{new}}\|_0 > n^b$ then
27:         return error
28:     end if
29:     $\Delta \leftarrow \begin{bmatrix} (U^{\mathrm{new}})^{-1} & 0 & (U^{\mathrm{new}})^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h^{\mathrm{new}} \\ ((U^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 & h^{\mathrm{new}} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
30:     Let $\Delta = XY^\top$ where $X \in \mathbb{R}^{(4n+d+1)\times n^b}$ and $Y \in \mathbb{R}^{(4n+d+1)\times n^b}$    ▷ $X$ consists of nonzero entries, $Y$ is a column selection matrix.
31:         ▷ Each row of $Y$ has only 1 nonzero entry
32:     $X \leftarrow X, Y \leftarrow Y$
33:     $S \leftarrow I + Y^\top N X$    ▷ Compute $Y^\top N$ takes $O(n^b)$ time and $Y^\top N X$ takes $O(n^{a+b})$ time
34:     $Q \leftarrow S^{-1}$    ▷ Takes $O(n^{b\cdot\omega})$ time
35: end procedure
```

D.2.1 Initialization time

Lemma D.5 (Initialization Time). *Taking a threshold $a \in (0,1)$, a vector $u_0 \in \mathbb{R}^n$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n\times n}$ and a matrix $A \in \mathbb{R}^{d\times n}$ as input,* INIT *(Algorithm 9) operation takes $O(n^\omega)$ time to complete.*

Algorithm 10 Query and Reset

```
1: data structure ProjectionMaintenanceLeftWithVector    ▷ Theorem D.1
2: procedure Query($I \subset [n], j \in [n]$)    ▷ Lemma D.3 and Lemma D.7
3:                                                 ▷ Compute query using matrix Woodbury formula
4:                                                 ▷ Return $(M^{-1})_{I,j} - (M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$
5:     $v_1 \leftarrow Y^\top \cdot N \cdot e_j$    ▷ Takes $O(n^b)$ time, $v_1 \in \mathbb{R}^{n^b}$
6:     $v_2 \leftarrow Qv_1$    ▷ Takes $O(n^{2b})$ time
7:     $L \leftarrow (NX)_I$    ▷ Takes $O(|I| \cdot n^a)$ time, $L \in \mathbb{R}^{|I| \times n^b}$
8:     $v_3 \leftarrow Lv_2$    ▷ Takes $O(|I| \cdot n^b)$ time
9:     $v \leftarrow N_{I,j} - v_3$
10:    return $v$
11: end procedure
12:
13: procedure Reset($X^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times k}, Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times k}$)    ▷ Lemma D.4 and Lemma D.8
14:                                                ▷ To run this procedure, we require that $k \geq n^a$
15:                                                ▷ Compute $M^{-1}$ explicitly by matrix Woodbury formula
16:                                                ▷ $(M+XY^\top)^{-1} = M^{-1} - M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}$
17:                                                ▷ Let $k = n^c$
18:    $L_1 \leftarrow (Y^{\mathrm{new}})^\top N$    ▷ Takes $O(n^{\omega(c,1,1)})$ time
19:    $Q \leftarrow (I + (Y^{\mathrm{new}})^\top M^{-1} X^{\mathrm{new}})^{-1}$    ▷ Takes $O(n^{\omega(1,1,c)})$ time
20:    $L_2 \leftarrow QL_1$    ▷ Takes $O(n^{\omega(c,c,1)})$ time
21:    $L_3 \leftarrow NX^{\mathrm{new}}L_2$    ▷ Takes $O(n^{\omega(1,1,c)})$ time
22:    $L \leftarrow N - L_3$
23:    $N \leftarrow L$
24:    $M \leftarrow M + X^{\mathrm{new}}(Y^{\mathrm{new}})^\top$
25:    $X \leftarrow 0, Y \leftarrow 0$
26: end procedure
27: end data structure
```

Proof. The running time of Init (Algorithm 5) operation consists of the following component:

- $N \leftarrow M^{-1}$ takes $O(n^\omega)$ to compute the matrix inverse of $M \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$.

Therefore, we complete the proof. ☐

D.2.2 Update time

Lemma D.6 (Update Time)**.** *Taking vectors $u^{\mathrm{new}} \in \mathbb{R}^n$ and $h \in \mathbb{R}^n$ as input, the* Update *(Algorithm 9) operation takes $O(n^{a+b} + n^{b\cdot\omega})$ time to complete.*

Proof. The running time of Update operation consists of the following components:

- $Y^\top N$ takes $O(n^a)$ time to compute the matrix multiplication between a sparse matrix $Y^\top \in \mathbb{R}^{3n^a \times (4n+d+1)}$ where each column of Y only has one non-zero entry and matrix $N \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$.
- $(Y^\top N) \cdot X$ takes $O(n^{2a}))$ time to compute the matrix multiplication between $Y^\top N \in \mathbb{R}^{3n^a \times (4n+d+1)}$ and $X \in \mathbb{R}^{(4n+d+1)\times 3n^a}$.
- $Q \leftarrow S^{-1}$ takes $o(n^{a\omega})$ time to compute the matrix inverse of $S \in \mathbb{R}^{3n^a \times 3n^a}$.

Therefore, we have:

$$\begin{aligned} & O(n^a) + O(n^{2a}) + O(n^{a\omega}) \\ = & O(n^{a\omega}) \end{aligned}$$

This completes the proof. ☐

D.2.3 Query time

Lemma D.7 (Query Time)**.** *Taking $I \subset [n]$ and an index $j \in [n]$ as input, the* QUERY *(Algorithm 10) operation takes $O(n^{2b} + |I| \cdot n^a)$ time to complete.*

Proof. The running time of QUERY operation consists of the following components:

- $v_1 \leftarrow Y^\top N e_j$ takes $O(n^a)$ time to compute $Y^\top N e_j$ where Y only contains one non-zero entry per column and e_j only has non-zero entry on jth element.
- $v_2 \leftarrow Q v_1$ takes $O(n^{2a})$ time to compute the matrix vector multiplication between $Q \in \mathbb{R}^{3n^a \times 3n^a}$ and $v_1 \in \mathbb{R}^{3n^a}$.
- $L \leftarrow (NX)_I$ takes $O(|I| \cdot n^a)$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$ and $X \in \mathbb{R}^{(4n+d+1)\times 3n^a}$ for $|I|$ rows.
- $v_3 \leftarrow L v_2$ takes $O(|I| \cdot n^a)$ time to compute the matrix vector multiplication between $L \in \mathbb{R}^{|I| \times 3n^a}$ and $v_2 \in \mathbb{R}^{3n^a}$.

Therefore, we have:

$$\begin{aligned} & O(n^a) + O(n^{2a}) + O(|I| \cdot n^a) + O(|I| \cdot n^a) \\ = & O(n^{2a} + |I| \cdot n^a) \end{aligned}$$

This completes our proof. □

D.2.4 Reset time

Lemma D.8 (Reset Time)**.** *Let $X^{\text{new}}, Y^{\text{new}} \in \mathbb{R}^{(4n+d)\times n^c}$ be the inputs to* RESET *(Algorithm 10), then* RESET *takes $O(n^{\omega(c,1,1)})$ time to complete.*

Proof. The running time of RESET operation consists of the following components:

- $L_1 \leftarrow Y^\top N$ takes $O(n^{\omega(c,1,1)})$ time to compute the multiplication between a $n^c \times (4n+d+1)$ matrix and a $(4n+d+1) \times (4n+d+1)$ matrix.
- $L_2 \leftarrow Q L_1$ takes $O(n^{\omega(c,c,1)})$ time to compute the matrix multiplication between $Q \in \mathbb{R}^{n^c \times n^c}$ and $L_1 \in \mathbb{R}^{n^c \times (4n+d+1)}$.
- $L_3 \leftarrow N X L_2$ takes $O(n^{\omega(1,1,c)})$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$ and $X \in \mathbb{R}^{(4n+d+1)\times n^c}$ and $O(n^{\omega(1,1,c)})$ time to compute the matrix multiplication between $NX \in \mathbb{R}^{(4n+d+1)\times n^c}$ and $L_2 \in \mathbb{R}^{n^c \times (4n+d+1)}$. Therefore, the total time for $L_3 \leftarrow N X L_2$ is $O(n^{\omega(1,1,c)})$.

Therefore, we have:

$$\begin{aligned} & O(n^{\omega(c,1,1)}) + O(n^{\omega(c,c,1)}) + O(n^{\omega(1,1,c)}) \\ = & O(n^{\omega(c,1,1)}) \end{aligned}$$

This completes the proof. □

D.3 Implementing OneStep

Note that the time of QUERYONESTEP is $O(n^{1.5} + n^{2b})$ since we only need to query one column with $\sqrt{n}$ rows.

UPDATEONESTEP makes use of UPDATE. Hence the overall running time is $O(n^{1+a})$ if $r < n^a$ and $O(r g_r n^{2+o(1)})$ if $r \geq n^a$.

Lemma D.9 (QueryOneStep Time)**.** *Given a query vector $h \in \mathbb{R}^n$ as input, the* QUERYONESTEP *(Algorithm 11) operation takes $O(n^{1.5} + n^{2b})$ time to complete.*

Proof. The QUERYONESTEP operation contains three steps:

- It takes $O(n^{2b} + |I| \cdot n^a)$ time to call QUERY operation for 1 times.
- It takes $O(n^{3/2})$ time to compute the multiplication between $R_l^\top \in \mathbb{R}^{n \times \sqrt{n}}$ and $x \in \mathbb{R}^{\sqrt{n}}$.
- It takes $O(n^{3/2})$ time to compute the multiplication between $R_l \in \mathbb{R}^{\sqrt{n} \times n}$ and $h \in \mathbb{R}^n$ and $O(n^{3/2})$ time to compute the multiplication between $R_l^\top \in \mathbb{R}^{n \times \sqrt{n}}$ and $R_l x \in \mathbb{R}^{\sqrt{n}}$.

Therefore, we have the overall time complexity of QUERYONESTEP operation:

$$\begin{aligned} & O(n^{2b} + |I| \cdot n^a) + O(n^{3/2}) + O(n^{3/2}) + O(n^{3/2}) \\ = & O(n^{3/2} + n^{2b}) \end{aligned}$$

This completes the proof. □

Lemma D.10 (UpdateOneStep Time). *Given an update matrix $W \in \mathbb{R}^{n \times n}$ as input, the* UPDATEONESTEP *(Algorithm 11) operation takes*

$$\begin{cases} O(n^{a+b} + n^{b \cdot \omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \geq n^a \end{cases}$$

time to complete.

Proof. When $r < n^a$, the time is dominant by UPDATE operation. By Lemma D.6, we know the time complexity is $O(n^{a+b} + n^{b \cdot \omega})$.

When $r \geq n^a$, the time is dominant by RESET operation. By Lemma D.4, we know the time complexity is $O(n^{\omega(c,1,1)})$ where $r = n^c$. By the definition of g_r in Definition A.1 and Theorem 3.1, we know the time complexity is $O(r g_r n^{2+o(1)})$.

This completes the proof. □

Remark D.11. *We observe the reason we can recover (Lee et al., 2019) is due to the fact that vector h is also stored, therefore we only need to query one column of M^{-1} instead of n columns, which is too slow. We will observe similar phenomenon in the right sketch case.*

E SKETCH DATA STRUCTURE WITH VECTOR, ON RIGHT

Similar to Appendix D, we show how to design M so that M^{-1} contains $\sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}\mathsf{R}^\top\mathsf{R}h$. When updating the righthand side vector $\mathsf{R}h$, we note that it is sufficient to store $I_l\mathsf{R}h$ where I_l is the matrix with only diagonal corresponding to indices in $[l]$ is 1 and all others 0. This selects the sketching matrix we need to query R_l.

We present the corresponding initialization, update, reset and query functions in Algorithm 13, Algorithm 14, Algorithm 15 and Algorithm 15. Similar to Section C, The INITIALIZE operation construct a right sketching matrix M from Lemma 5.10 with the query vector h embedded inside, and compute the inverse $N = M^{-1}$. The UPDATE operation updates the inverse matrix such that $Q = (I + Y^\top M^{-1} X)^{-1}$. The RESET operation reset the value of N such that $N = M^{-1}$. QUERY operation takes $I \subset [n]$ and an index $j \in [n]$ as input and computes v such that

$$v = (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I.$$

UPDATEONESTEP operation takes an update vector $w \in \mathbb{R}^n$ and decides to call either UPDATE or RESET to maintain the inverse of the matrix. QUERYONESTEP calls QUERY operation for n times and obtain the projected sketching result y and z such that:

$$\begin{aligned} y &= \widetilde{P} R_l^\top R_l h \\ z &= (I - \widetilde{P}) R_l^\top R_l h \end{aligned}$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}}A^\top(A\widetilde{U}A^\top)^{-1}A\sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

Algorithm 11 Restatement of Algorithm 2 and 3 in (Lee et al., 2019).

```
1: data structure ProjectionMaintenanceLeftWithVector                ▷ Theorem D.1
2: procedure UpdateOneStep($W \in \mathbb{R}^{n\times n}, h \in \mathbb{R}^n$)    ▷ Lemma D.10
3:     $y_i = v_i^{-1/2} w_i v_i^{-1/2} - 1, \forall i \in [n]$
4:     $r \leftarrow$ the number of indices $i$ such that $\|y_i\|_F \geq \epsilon_{mp}$
5:     if $r < n^a$ then
6:         $v^{\mathrm{new}} \leftarrow v$
7:         $l \leftarrow l + 1$
8:     else
9:         Let $\pi : [n] \longrightarrow [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
10:        while $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r\rceil)}| \geq (1 - 1/\log(n))|y_{\pi(r)}|$ do
11:            $r \leftarrow \min(\lceil 1.5 \cdot r\rceil, n)$
12:        end while
13:        $v^{\mathrm{new}}_{\pi(i)} \leftarrow \begin{cases} w_{\pi(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\pi(i)} & i \in \{r+1, \cdots, n\} \end{cases}$
14:        $\Delta \leftarrow \begin{bmatrix} (V^{\mathrm{new}})^{-1} & 0 & (V^{\mathrm{new}})^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h \\ ((V^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 & h \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
15:        Let $\Delta = X^{\mathrm{new}}(Y^{\mathrm{new}})^\top$ where $X^{\mathrm{new}} \in \mathbb{R}^{(4n+d+1)\times k}$ and $Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d+1)\times k}$   ▷ $X^{\mathrm{new}}$ consists of nonzero
           entries, $Y^{\mathrm{new}}$ is a column selection matrix. $h$ can change $n$ entries.
16:        Reset($X^{\mathrm{new}}, Y^{\mathrm{new}}$)
17:        $l \leftarrow 1$
18:    end if
19:    $v \leftarrow v^{\mathrm{new}}$
20:    $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } (1-\epsilon_{mp})v_i \preceq w_i \preceq (1+\epsilon_{mp})v_i \\ w_i & \text{otherwise} \end{cases}$
21:    if $r < n^a$ then                                ▷ This is put small update in UpdateOneStep
22:        Update($\widetilde{v}$)   ▷ Update the matrix with $\begin{bmatrix} (\widetilde{V})^{-1} & 0 & (\widetilde{V})^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h \\ (\widetilde{V})^{-1/2} & 0 & 0 & 0 & 0 & h \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
23:    end if
24: end procedure
25: procedure QueryOneStep($h \in \mathbb{R}^n$)                          ▷ Lemma D.9
26:                                         ▷ We query the rows corresponding to $R_t$
27:    $I \leftarrow$ rows corresponding to $R_l$
28:    $j \leftarrow 4n + d + 1$
29:    $x \leftarrow$ Query($I, j$)                                   ▷ $x$ is $R_l P h$
30:    $y \leftarrow R_l^\top x$                                      ▷ $y$ is $R_l^\top R_l P h$
31:    $z \leftarrow R_l^\top R_l h - y$                              ▷ $z$ is $R_l^\top R_l (I - P)h$
32:    return $(y, z)$
33: end procedure
34: end data structure
```

Theorem E.1 (Sketching on right with vector). *Given a matrix* $\mathsf{R} \in \mathbb{R}^{n\times n}$ *where* $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \cdots & R_T^\top \end{bmatrix}$. *Let* $T = \sqrt{n}$. *Let* $m = 4n + d$.

Let $A \in \mathbb{R}^{d\times n}$ *of rank* d *and let* $U \in \mathbb{R}^{n\times n}$ *be a diagonal matrix with non-zero diagonal entries,* $v \in \mathbb{R}^n$ *and* $\mathsf{R} \in \mathbb{R}^{n\times n}$ *be a sketch matrix. There exists a data structure* ProjectionMaintenanceRight *(Algorithm 13 and 14) which supports*

Algorithm 12 Restatement of Algorithm 5 in (Lee et al., 2019)

```
1: procedure CENTRALPATHSTEP($\overline{x}, \overline{s}, t, \lambda, \alpha$)
2:     for $i = 1 \to n$ do
3:         $\mu_i^t \leftarrow \overline{s}_i/t + \nabla\phi_i(\overline{x}_i)$
4:         $\gamma_i^t \leftarrow \|\mu_i^t\|_{\nabla^2\phi_i(\overline{x}_i)^{-1}}$
5:         $c_i^t \leftarrow \frac{\exp(\lambda\gamma_i^t)/\gamma_i^t}{(\sum_{i=1}^n \exp(2\lambda\gamma_i^t))^{1/2}}$ if $\gamma_i^t \geq 96\sqrt{\alpha}$ and $c_i^t \leftarrow 0$ otherwise
6:         $h_i \leftarrow -\alpha \cdot c_i^t \cdot \mu_i^t$
7:     end for
8:     $\overline{W} \leftarrow (\nabla^2\phi(\overline{x}))^{-1}$
9:     return $h, \overline{W}$
10: end procedure
11:
12: procedure ONESTEP(mp, $x_{\mathrm{init}}, s_{\mathrm{init}}, t, \lambda, \alpha$)
13:     $h, \overline{W} \leftarrow$ CENTRALPATHSTEP($x_{\mathrm{init}}, s_{\mathrm{init}}, t, \lambda, \alpha$)
14:     mp.UPDATEONESTEP($\overline{W}, h$)
15:     $(\overline{x}, \overline{s}) \leftarrow$ mp.QUERYONESTEP($h$)
16:     return $(\overline{x}, \overline{s})$
17: end procedure
```

the following operations:

- INIT$(a \in (0,1), b \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n\times n}, A \in \mathbb{R}^{d\times n})$*: Given thresholds* $a, b \in (0,1)$ *with* $b \leq a$*, a vector* $u_0 \in \mathbb{R}^n$*, a sketching matrix* $\mathsf{R} \in \mathbb{R}^{n\times n}$ *and a matrix* $A \in \mathbb{R}^{d\times n}$ *as input,* INIT *(Algorithm 13) operation runs in* $O(n^\omega)$ *time.*
- UPDATE$(u^{\mathrm{new}} \in \mathbb{R}^n, h^{\mathrm{new}} \in \mathbb{R}^{\mathrm{new}})$*: Given a vector* $u^{\mathrm{new}} \in \mathbb{R}^n$ *as input, the* UPDATE *(Algorithm 13) operation runs* $O(n^{a+b} + n^{b\cdot\omega})$ *time.*
- QUERY$(I \subset [n], j \in [n])$*: Given* $I \subset [n]$ *and an index* $j \in [n]$ *as input, the* QUERY *(Algorithm 14) operation runs in* $O(n^{2b} + |I| \cdot n^a)$ *time.*
- RESET$(X^{\mathrm{new}}, Y^{\mathrm{new}})$*: Give matrices* $X^{\mathrm{new}}, Y^{\mathrm{new}} \in \mathbb{R}^{n\times n^c}$*,* RESET *(Algorithm 14) operations runs in* $O(n^{\omega(c,1,1)})$ *time.*

E.1 Correctness

In this section, we first present the correctness proof for UPDATE in Lemma E.2. We present the correctness proof for QUERY in Lemma E.3. Then we present the correctness proof for RESET in Lemma E.4.

Lemma E.2 (Update correctness)**.** *The output of* UPDATE$(u^{\mathrm{new}} \in \mathbb{R}^n, h \in \mathbb{R}^n)$ *in Algorithm 13 satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. This follows directly from the invariant that $N = M^{-1}$. Note that by storing the quantity $(I + Y^\top M^{-1}X)^{-1}$, we can compute the query quickly, as we will show later. ☐

Lemma E.3 (Query correctness)**.** *The output of* QUERY$(I \subset [n], j \in [n])$ *in Algorithm 14 satisfies:*

$$v = (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$$

Proof. We have

$$\begin{aligned} v &= N_{I,j} - (NX)_I Q Y^\top N e_j \\ &= (M^{-1})_{I,j} - (M^{-1}XQY^\top M^{-1}e_j)_I \\ &= (M^{-1})_{I,j} - (M^{-1}X(I + Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I \end{aligned}$$

Algorithm 13 1-layer projection maintenance by inverse maintenance with Sketching on the right. Note that we store the inverse of M, but only update it during the procedure RESET. For UPDATE, we assume the change happens for at most n^a entries in at most n^b columns.

```
1: data structure ProjectionMaintenanceRightWithVector    ▷ Theorem E.1
2: members
3:     $U \in \mathbb{R}^{n \times n}$
4:     $A \in \mathbb{R}^{d \times n}$
5:     $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \dots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n \times n}$
6:     $M \in \mathbb{R}^{(4n+d) \times (4n+d)}$
7:     $a \in (0,1)$    ▷ Threshold 1
8:     $b \in (0,1)$    ▷ Threshold 2
9:     $X, Y \in \mathbb{R}^{(4n+d) \times n^b}$    ▷ Each column of $Y$ has only one nonzero entry
10:    $Q \in \mathbb{R}^{n^b \times n^b}$
11:    $N \in \mathbb{R}^{(4n+d) \times (4n+d)}$    ▷ Inverse of $M$
12: end members
13:
14: procedure Init($a, b \in (0,1), u_0 \in \mathbb{R}^n, \mathsf{R} \in \mathbb{R}^{n \times n}, A \in \mathbb{R}^{d \times n}, h \in \mathbb{R}^n$)    ▷ Lemma E.5
15:     $a \leftarrow a, b \leftarrow b, A \leftarrow A$
16:     $U \leftarrow \mathrm{diag}(u_0)$
17:     $\mathsf{R} \leftarrow \mathsf{R}$
18:     $M \leftarrow \begin{bmatrix} U^{-1} & A^\top & U^{-1/2} & 0 & 0 & 0 \\ A & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -I & 0 & \mathsf{R}^\top & h \\ (U^{-1/2})^\top & 0 & 0 & -I & \mathsf{R}^\top & h \\ 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$    ▷ Left sketching matrix construction from Lemma 5.9
19:     $X, Y \leftarrow \mathbf{0}_{(4n+d) \times n^b}$
20:     $N \leftarrow M^{-1}$    ▷ Takes $O(n^\omega)$ time
21:     $Q \leftarrow \mathbf{0}_{n^b \times n^b}$
22: end procedure
23:
24: procedure Update($u^{\mathrm{new}} \in \mathbb{R}^n, h^{\mathrm{new}} \in \mathbb{R}^n$)    ▷ Lemma E.2 and Lemma E.6
25:     ▷ To run this procedure, we require that $u^{\mathrm{new}}$ has at most $n^b$ non-zero entries
26:     if $\|u^{\mathrm{new}}\|_0 > n^b$ then
27:         return error
28:     end if
29:     $\Delta \leftarrow \begin{bmatrix} (U^{\mathrm{new}})^{-1} & 0 & (U^{\mathrm{new}})^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h^{\mathrm{new}} \\ ((U^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 & h^{\mathrm{new}} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
30:     Let $\Delta = XY^\top$ where $X \in \mathbb{R}^{(4n+d+1) \times n^b}$ and $Y \in \mathbb{R}^{(4n+d+1) \times n^b}$    ▷ $X$ consists of nonzero entries, $Y$ is a column selection matrix.
31:     ▷ Each row of $Y$ has only 1 nonzero entry
32:     $X \leftarrow X, Y \leftarrow Y$
33:     $S \leftarrow I + Y^\top N X$    ▷ Compute $Y^\top N$ takes $O(n^b)$ time and $Y^\top N X$ takes $O(n^{a+b})$ time
34:     $Q \leftarrow S^{-1}$    ▷ Takes $O(n^{b \cdot \omega})$ time
35: end procedure
```

where the first step follows from $N = M^{-1}$, and the second step follows from $Q = (I + Y^\top N X)^{-1} = (I + Y^\top M^{-1} X)^{-1}$.

This completes the proof. □

Algorithm 14 Query and Reset

```
1: data structure ProjectionMaintenanceRightWithVector                          ▷ Theorem E.1
2: procedure Query($I \subset [n], j \in [n]$)                              ▷ Lemma E.3 and Lemma E.7
3:                                                   ▷ Compute query using matrix Woodbury formula
4:                             ▷ Return $(M^{-1})_{I,j} - (M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}e_j)_I$
5:     $v_1 \leftarrow Y^\top \cdot N \cdot e_j$                          ▷ Takes $O(n^b)$ time, $v_1 \in \mathbb{R}^{n^b}$
6:     $v_2 \leftarrow Qv_1$                                                     ▷ Takes $O(n^{2b})$ time
7:     $L \leftarrow (NX)_I$                            ▷ Takes $O(|I| \cdot n^a)$ time, $L \in \mathbb{R}^{|I| \times n^b}$
8:     $v_3 \leftarrow Lv_2$                                               ▷ Takes $O(|I| \cdot n^b)$ time
9:     $v \leftarrow N_{I,j} - v_3$
10:    return $v$
11: end procedure
12:
13: procedure Reset($X^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times k}, Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times k}$)      ▷ Lemma E.4 and Lemma E.8
14:                                                  ▷ To run this procedure, we require that $k \geq n^a$
15:                                          ▷ Compute $M^{-1}$ explicitly by matrix Woodbury formula
16:                         ▷ $(M+XY^\top)^{-1} = M^{-1} - M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1}$
17:                                                                                ▷ Let $k = n^c$
18:    $L_1 \leftarrow (Y^{\mathrm{new}})^\top N$                                 ▷ Takes $O(n^{\omega(c,1,1)})$ time
19:    $Q \leftarrow (I + (Y^{\mathrm{new}})^\top M^{-1} X^{\mathrm{new}})^{-1}$              ▷ Takes $O(n^{\omega(1,1,c)})$ time
20:    $L_2 \leftarrow QL_1$                                                  ▷ Takes $O(n^{\omega(c,c,1)})$ time
21:    $L_3 \leftarrow NX^{\mathrm{new}}L_2$                                       ▷ Takes $O(n^{\omega(1,1,c)})$ time
22:    $L \leftarrow N - L_3$
23:    $N \leftarrow L$
24:    $M \leftarrow M + X^{\mathrm{new}}(Y^{\mathrm{new}})^\top$
25:    $X \leftarrow 0, Y \leftarrow 0$
26: end procedure
27: end data structure
```

Lemma E.4 (Reset correctness)**.** *The output of* Reset$(X^{\mathrm{new}}, Y^{\mathrm{new}})$ *in Algorithm 14 satisfies:*

$$N = M^{-1}$$

Proof.

$$\begin{aligned} N^{\mathrm{new}} &= N - NXQY^\top N \\ &= M^{-1} - M^{-1}X(I+Y^\top M^{-1}X)^{-1}Y^\top M^{-1} \\ &= (M+XY^\top)^{-1} \\ &= (M^{\mathrm{new}})^{-1} \end{aligned}$$

where the first step follows from $N = M^{-1}$, the second step follows the matrix Woodbury formula, and the third step follows from Update in Algorithm 13.

Note at the end of Reset, N and M are updated with the new value. This completes the proof. □

E.2 Running Time

In this section, we first present the Init running time in Lemma E.5. Then we present the Update running time in Lemma E.6. We present the Query running time in Lemma E.7. We present the Reset running time in Lemma E.8.

We remark that we will store the dense vector h into matrix M, hence, the threshold parameter a becomes 1. This yields a slower update time of $O(n^{1+b})$ and a query time of $O(n^{1.5})$. This is still enough to reproduce the result in (Lee et al., 2019).

Algorithm 15 This part should achieve the same power as Algorithm 9 and 10 in (Lee et al., 2019).

```
1: data structure ProjectionMaintenanceRightWithVector                ▷ Theorem E.1
2: procedure UpdateOneStep($w \in \mathbb{R}^n, h \in \mathbb{R}^n$)        ▷ Lemma E.11 and Lemma E.12
3:     $y_i = \ln w_i - \ln v_i, \forall i \in [n]$
4:     $r \leftarrow$ the number of indices $i$ such that $|y_i| \geq \epsilon_{mp}/2$
5:     if $r < n^a$ then
6:         $v^{\mathrm{new}} \leftarrow v$
7:         $l \leftarrow l + 1$
8:     else
9:         Let $\pi : [n] \longrightarrow [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
10:        while $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r \rceil)}| \geq (1 - 1/\log(n))|y_{\pi(r)}|$ do
11:            $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
12:        end while
13:        $v^{\mathrm{new}}_{\pi(i)} \leftarrow \begin{cases} w_{\pi(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\pi(i)} & i \in \{r+1, \cdots, n\} \end{cases}$
14:        $\widetilde{h} \leftarrow \begin{bmatrix} R_1 h \\ \mathbf{0}_{n-\sqrt{n}} \end{bmatrix}$
15:        $\Delta \leftarrow \begin{bmatrix} (V^{\mathrm{new}})^{-1} & 0 & (V^{\mathrm{new}})^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \widetilde{h} \\ ((V^{\mathrm{new}})^{-1/2})^\top & 0 & 0 & 0 & 0 & \widetilde{h} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
16:        Let $\Delta = X^{\mathrm{new}}(Y^{\mathrm{new}})^\top$ where $X^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times k}$ and $Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times k}$    ▷ $X^{\mathrm{new}}$ consists of nonzero
           entries, $Y^{\mathrm{new}}$ is a column selection matrix
17:        Reset($X^{\mathrm{new}}, Y^{\mathrm{new}}$)
18:        $l \leftarrow 1$
19:    end if
20:    $v \leftarrow v^{\mathrm{new}}$
21:    $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } |\ln w_i - \ln v_i| < \epsilon_{\mathrm{mp}}/2 \\ w_i & \text{otherwise} \end{cases}$
22:    if $r < n^a$ then                                  ▷ This is put small update in UpdateOneStep
23:        $\widetilde{h} \leftarrow I_l \mathsf{R} h$                            ▷ Select out $R_l h$, make all other entries 0
24:        Update($\widetilde{v}, \widetilde{h}$)    ▷ Update the matrix with $\begin{bmatrix} (\widetilde{V})^{-1} & 0 & (\widetilde{V})^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \widetilde{h} \\ (\widetilde{V})^{-1/2} & 0 & 0 & 0 & 0 & \widetilde{h} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
25:    end if
26:    return $\widetilde{v}$
27: end procedure
28: procedure QueryOneStep($h \in \mathbb{R}^n$)                    ▷ Lemma E.9 and Lemma E.10
29:    $I \leftarrow [n]$
30:    $y \leftarrow$ Query($I, 4n + d + 1$)                          ▷ $y$ is $PR_l^\top R_l h$
31:    $z \leftarrow R_l^\top R_l h - y$                                ▷ $z$ is $(I - P)R_l^\top R_l h$
32:    return $(y, z)$
33: end procedure
34: end data structure
```

Algorithm 16 Restatement of Algorithm 7 in (Song and Yu, 2021)

```
1:  procedure OneStep(mp, $x, s, \delta_\mu, b_{\mathrm{sketch}}, \epsilon$)
2:      $w \leftarrow \frac{x}{s}$, $\widetilde{v} \leftarrow$ mp.UpdateOneStep($w$)
3:      $\overline{x} \leftarrow x\sqrt{\frac{\widetilde{v}}{w}}$, $\overline{s} \leftarrow s \cdot \sqrt{\frac{w}{\widetilde{v}}}$
4:      repeat
5:          $p_x, p_s \leftarrow$ mp.QueryOneStep($\frac{1}{\sqrt{\overline{X}\overline{S}}}\delta_\mu$)
6:          $\widetilde{\delta}_s \leftarrow \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}} p_s$
7:          $\widetilde{\delta}_x \leftarrow \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}} p_x$
8:      until $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{1}{100\log n}$ and $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{1}{100\log n}$
9:      return $(x + \widetilde{\delta}_x, s + \widetilde{\delta}_s)$
10: end procedure
```

E.2.1 Initialization time

Lemma E.5 (Initialization Time)**.** *Taking a threshold $a \in (0,1)$, a vector $u_0 \in \mathbb{R}^n$, a sketching matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$ and a matrix $A \in \mathbb{R}^{d \times n}$ as input,* Init *(Algorithm 13) operation takes $O(n^\omega)$ time to complete.*

Proof. The running time of Init (Algorithm 13) operation consists of the following component:

- $N \leftarrow M^{-1}$ takes $O(n^\omega)$ to compute the matrix inverse of $M \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$.

Therefore, we complete the proof. $\square$

E.2.2 Update time

Lemma E.6 (Update Time)**.** *Taking vectors $u^{\mathrm{new}} \in \mathbb{R}^n$ and $h \in \mathbb{R}^n$ as input, the* Update *(Algorithm 13) operation takes $O(n^{a+b} + n^{b\cdot\omega})$ time to complete.*

Proof. The running time of Update operation consists of the following components:

- $Y^\top N$ takes $O(n^a)$ time to compute the matrix multiplication between a sparse matrix $Y^\top \in \mathbb{R}^{3n^a \times (4n+d+1)}$ where each column of Y only has one non-zero entry and matrix $N \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$.
- $(Y^\top N) \cdot X$ takes $O(n^{2a}))$ time to compute the matrix multiplication between $Y^\top N \in \mathbb{R}^{3n^a \times (4n+d+1)}$ and $X \in \mathbb{R}^{(4n+d+1)\times 3n^a}$.
- $Q \leftarrow S^{-1}$ takes $o(n^{a\omega})$ time to compute the matrix inverse of $S \in \mathbb{R}^{3n^a \times 3n^a}$.

Therefore, we have:

$$\begin{aligned} & O(n^a) + O(n^{2a}) + O(n^{a\omega}) \\ = & O(n^{a\omega}) \end{aligned}$$

This completes the proof. $\square$

E.2.3 Query time

Lemma E.7 (Query Time)**.** *Taking $I \subset [n]$ and an index $j \in [n]$ as input, the* Query *(Algorithm 14) operation takes $O(n^{2b} + |I| \cdot n^a)$ time to complete.*

Proof. The running time of Query operation consists of the following components:

- $v_1 \leftarrow Y^\top N e_j$ takes $O(n^a)$ time to compute $Y^\top N e_j$ where Y only contains one non-zero entry per column and e_j only has non-zero entry on jth element.
- $v_2 \leftarrow Q v_1$ takes $O(n^{2a})$ time to compute the matrix vector multiplication between $Q \in \mathbb{R}^{3n^a \times 3n^a}$ and $v_1 \in \mathbb{R}^{3n^a}$.
- $L \leftarrow (NX)_I$ takes $O(|I| \cdot n^a)$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$ and $X \in \mathbb{R}^{(4n+d+1)\times 3n^a}$ for $|I|$ rows.
- $v_3 \leftarrow L v_2$ takes $O(|I| \cdot n^a)$ time to compute the matrix vector multiplication between $L \in \mathbb{R}^{|I| \times 3n^a}$ and $v_2 \in \mathbb{R}^{3n^a}$.

Therefore, we have:

$$\begin{aligned} & O(n^a) + O(n^{2a}) + O(|I| \cdot n^a) + O(|I| \cdot n^a) \\ = & O(n^{2a} + |I| \cdot n^a) \end{aligned}$$

This completes our proof. □

E.2.4 Reset time

Lemma E.8 (Reset Time)**.** *Let $X^{\mathrm{new}}, Y^{\mathrm{new}} \in \mathbb{R}^{(4n+d)\times n^c}$ be the inputs to* RESET *(Algorithm 14), then* RESET *takes $O(n^{\omega(c,1,1)})$ time to complete.*

Proof. The running time of RESET operation consists of the following components:

- $L_1 \leftarrow Y^\top N$ takes $O(n^{\omega(c,1,1)})$ time to compute the multiplication between a $n^c \times (4n+d+1)$ matrix and a $(4n+d+1)\times(4n+d+1)$ matrix.
- $L_2 \leftarrow Q L_1$ takes $O(n^{\omega(c,c,1)})$ time to compute the matrix multiplication between $Q \in \mathbb{R}^{n^c \times n^c}$ and $L_1 \in \mathbb{R}^{n^c \times (4n+d+1)}$.
- $L_3 \leftarrow N X L_2$ takes $O(n^{\omega(1,1,c)})$ time to compute the matrix multiplication between $N \in \mathbb{R}^{(4n+d+1)\times(4n+d+1)}$ and $X \in \mathbb{R}^{(4n+d+1)\times n^c}$ and $O(n^{\omega(1,1,c)})$ time to compute the matrix multiplication between $NX \in \mathbb{R}^{(4n+d+1)\times n^c}$ and $L_2 \in \mathbb{R}^{n^c \times (4n+d+1)}$. Therefore, the total time for $L_3 \leftarrow NXL_2$ is $O(n^{\omega(1,1,c)})$.

Therefore, we have:

$$\begin{aligned} & O(n^{\omega(c,1,1)}) + O(n^{\omega(c,c,1)}) + O(n^{\omega(1,1,c)}) \\ = & O(n^{\omega(c,1,1)}) \end{aligned}$$

This completes the proof. □

E.3 Implementing OneStep

Lemma E.9 (QueryOneStep correctness)**.** *Given a query vector $h \in \mathbb{R}^n$, the output of* QUERYONESTEP *satisfies:*

$$\begin{aligned} y &= \widetilde{P} R_l^\top R_l h \\ z &= (I - \widetilde{P}) R_l^\top R_l h \end{aligned}$$

where the projection matrix $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A \widetilde{U} A^\top)^{-1} A \sqrt{\widetilde{U}}$ where $\widetilde{U}$ is output from the last UPDATEONESTEP.

Proof. We have:

$$\begin{aligned} y &= \textsc{Query}(I, 4n+d+1) \\ &= \widetilde{P} R_l^\top R_l h \end{aligned}$$

where the first step follows the QUERYONESTEP algorithm, the second step follows from the output of QUERY in Lemma E.3 and $\widetilde{P} = \sqrt{\widetilde{U}} A^\top (A\widetilde{U}A^\top)^{-1} A \sqrt{\widetilde{U}}$.

Therefore, we have:

$$z = R_l^\top R_l h - y = (I - \widetilde{P}) R_l^\top R_l h$$

This completes our proof. □

Lemma E.10 (QueryOneStep Time)**.** *Given a query vector $h \in \mathbb{R}^n$ as input, the* QUERYONESTEP *(Algorithm 15) operation takes $O(n^{1.5} + n^{2b} + n^{1+a})$ time to complete.*

Proof. The QUERYONESTEP operation contains three steps:

- It takes $O(n^{2b} + |I| \cdot n^a)$ time to call QUERY operation for 1 time.
- It takes $O(n^{3/2})$ time to compute the multiplication between $R_l \in \mathbb{R}^{\sqrt{n} \times n}$ and $h \in \mathbb{R}^n$ and $O(n^{3/2})$ time to compute the multiplication between $R_l^\top \in \mathbb{R}^{n \times \sqrt{n}}$ and $R_l h \in \mathbb{R}^{\sqrt{n}}$.

Therefore, we have the overall time complexity of QUERYONESTEP operation:

$$\begin{aligned} & O(n^{2b} + |I| \cdot n^a) + O(n^{3/2}) + O(n^{3/2}) \\ = & O(n^{3/2} + n^{2b} + n^{1+a}) \end{aligned}$$

where the first step follows that $|I| = n$ and $a < 1$. This completes the proof. □

Lemma E.11 (UpdateOneStep correctness)**.** *Given an update vector $w \in \mathbb{R}^n$, the output of* UPDATEONESTEP *satisfies:*

$$Q = (I + Y^\top M^{-1} X)^{-1}$$

Proof. When $r < n^a$, by Lemma E.2, we have $Q = (I + Y^\top M^{-1} X)^{-1}$.

When $r \geq n^a$, RESET is called. X and Y are reset as zero matrice and Q is reset as identity matrix. Therefore, we have:

$$\begin{aligned} Q &= I \\ &= (I + Y^\top M^{-1} X)^{-1} \end{aligned}$$

where the first step follows that Q is reset as identity matrix, and the second step follows that $Y^\top M^{-1} X = 0$.

This completes our proof.

□

Lemma E.12 (UpdateOneStep Time)**.** *Taking a vector $w \in \mathbb{R}^n$ as input, the* UPDATEONESTEP *(Algorithm 15) operation takes*

$$\begin{cases} O(n^{a+b} + n^{b \cdot \omega}) & \text{if } r < n^a \\ O(r g_r n^{2+o(1)}) & \text{if } r \geq n^a \end{cases}$$

time to complete.

Proof. When $r < n^a$, the time is dominant by UPDATE operation. By Lemma E.6, we know the time complexity is $O(n^{a+b} + n^{b \cdot \omega})$.

When $r \geq n^a$, the time is dominant by RESET operation. By Lemma E.4, we know the time complexity is $O(n^{\omega(c,1,1)})$ where $r = n^c$. By the definition of g_r in Definition A.1 and Theorem 3.1, we know the time complexity is $O(r g_r n^{2+o(1)})$.

This completes the proof. □

Remark E.13. *Even though we need to query n rows instead of $\sqrt{n}$ rows, it is still enough to recover the result in (Cohen et al., 2019b; Song and Yu, 2021).*

The advantage is UPDATEONESTEP *is faster, in the left sketch case, we need to pay $O(n^{1+a})$ time to update the dense vector h. In contrary, by sketching-on-the-right, we only need to update a sparse vector with $O(\sqrt{n})$ entries, hence, the update time is $O(n^{0.5+a})$. This term is a small term, and will be hidden by other terms.*

F MISSING PROOFS FOR SECTION 5

F.1 Summary of Our Results

	INIT	UPDATE	QUERY	RESET	Solve (Lee et al., 2019)	Solve (Cohen et al., 2019b; Song and Yu, 2021)
Theorem B.1	n^ω	$n^{a+b}+n^{b\cdot\omega}$	$n^{2b}+\lvert I\rvert\cdot n^a$	$n^{\omega(c,1,1)}$	$n^{2+1/3}$	
Theorem C.1	n^ω	$n^{a+b}+n^{b\cdot\omega}$	$n^{2b}+\lvert I\rvert\cdot n^a$	$n^{\omega(c,1,1)}$		$n^{2+1/3}$
Theorem D.1	n^ω	$n^{a+b}+n^{b\cdot\omega}$	$n^{2b}+\lvert I\rvert\cdot n^a$	$n^{\omega(c,1,1)}$	$n^{2+1/6}$	
Theorem E.1	n^ω	$n^{a+b}+n^{b\cdot\omega}$	$n^{2b}+\lvert I\rvert\cdot n^a$	$n^{\omega(c,1,1)}$		$n^{2+1/6}$

Table 1: Comparison between our results.

	Solve	$\lvert I\rvert$	# queries per iter
Theorem B.1	(Lee et al., 2019)	$\sqrt{n}$	n
Theorem C.1	(Cohen et al., 2019b; Song and Yu, 2021)	n	$\sqrt{n}$
Theorem D.1	(Lee et al., 2019)	$\sqrt{n}$	1
Theorem E.1	(Cohen et al., 2019b; Song and Yu, 2021)	n	1

Table 2: For the first one, we need to balance the time $(n^{2a}+n^{0.5+a})n$ with $n^{2-a/2}$. This leads to the optimal choice of $a=1/3$. This gives running time to be $n^{2+1/3}$. For the second one, we need to balance the time $(n^{2a}+n^{1+a})n^{0.5}$ with $n^{2-a/2}$. This leads to optimal choice of $a=1/3$. This gives running time to be $n^{2+1/3}$.

Lemma F.1 (Formal version of Lemma 5.2)**.** *Let $M\in\mathbb{R}^{(n+d)\times(n+d)}$ be defined as in Definition 5.1, then*

$$M^{-1}=\begin{bmatrix} U-UA^\top(AUA^\top)^{-1}AU & UA^\top(AUA^\top)^{-1} \\ (AUA^\top)^{-1}AU & -(AUA^\top)^{-1}\end{bmatrix}$$

Proof. We have

$$\begin{aligned} M^{-1} &= \left(\begin{bmatrix} U^{-1} & A^\top \\ A & 0\end{bmatrix}\right)^{-1} \\ &= \begin{bmatrix} U+UA^\top(0-AUA^\top)^{-1}AU & -UA^\top(0-AUA^\top)^{-1} \\ -(0-AUA^\top)^{-1}AU & (0-AUA^\top)^{-1}\end{bmatrix} \\ &= \begin{bmatrix} U-UA^\top(AUA^\top)^{-1}AU & UA^\top(AUA^\top)^{-1} \\ (AUA^\top)^{-1}AU & -(AUA^\top)^{-1}\end{bmatrix}\end{aligned}$$

where the first step follows from Fact 4.1 and the second step comes from simplying the terms. □

Lemma F.2 (Sketch on the left. Formal version of Lemma 5.7)**.** *Let $R\in\mathbb{R}^{n\times(3n+d)}$, let $L\in\mathbb{R}^{(3n+d)\times(3n+d)}$ be the matrix defined in Definition 5.3, consider the matrix*

$$\begin{bmatrix} L & 0 \\ R & -I\end{bmatrix},$$

then we have

$$\left(\begin{bmatrix} L & 0 \\ R & -I\end{bmatrix}\right)^{-1}=\begin{bmatrix} L^{-1} & 0 \\ RL^{-1} & -I\end{bmatrix}$$

Proof. We have

$$\begin{aligned}\left(\begin{bmatrix} L & 0 \\ R & -I \end{bmatrix}\right)^{-1} &= \begin{bmatrix} L^{-1} & 0 \\ -(-I)^{-1}RL^{-1} & (-I)^{-1} \end{bmatrix} \\ &= \begin{bmatrix} L^{-1} & 0 \\ RL^{-1} & -I \end{bmatrix}\end{aligned}$$

where the first step comes from Fact 4.1 and the second step comes from simplying the terms. □

Lemma F.3 (Sketch on the right, Formal version of Lemma 5.8)**.** *Let $R \in \mathbb{R}^{(3n+d)\times n}$, let $L \in \mathbb{R}^{(3n+d)\times(3n+d)}$ be the matrix defined in Definition 5.3, consider the matrix*

$$\begin{bmatrix} L & R \\ 0 & -I \end{bmatrix},$$

then we have

$$\left(\begin{bmatrix} L & R \\ 0 & -I \end{bmatrix}\right)^{-1} = \begin{bmatrix} L^{-1} & L^{-1}R \\ 0 & -I \end{bmatrix}$$

Proof. We have

$$\begin{aligned}\left(\begin{bmatrix} L & R \\ 0 & -I \end{bmatrix}\right)^{-1} &= \begin{bmatrix} L^{-1} & -L^{-1}R(-I)^{-1} \\ 0 & (-I)^{-1} \end{bmatrix} \\ &= \begin{bmatrix} L^{-1} & L^{-1}R \\ 0 & -I \end{bmatrix}\end{aligned}$$

where the first step comes from Fact 4.1 and the second step comes from simplying the terms. □

Lemma F.4 (Formal version of Lemma 5.9)**.** *Let $\mathsf{R} \in \mathbb{R}^{n\times n}$ be a collection of sketching matrices, define $R \in \mathbb{R}^{n\times(3n+d)}$ to be the following matrix:*

$$R = \begin{bmatrix} \mathbf{0}_{n\times(2n+d)} & \mathsf{I_t R} \end{bmatrix}$$

Then we have

$$\left(\begin{bmatrix} L & \mathbf{0}_{(3n+d)\times n} \\ R & -I_n \end{bmatrix}\right)^{-1} \begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \\ \mathbf{0}_n \end{bmatrix} = \begin{bmatrix} \star \\ R_t\sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v \end{bmatrix}$$

where I_t is a diagonal matrix whose $\frac{n(t-1)}{T}$th to $\frac{nt}{T}$th diagonal entries are 1 and other diagonal entries are 0 such that $I_t\mathsf{R} = R_t$.

Proof. By Lemma 5.7, we know

$$\left(\begin{bmatrix} L & \mathbf{0}_{(3n+d)\times n} \\ R & -I_n \end{bmatrix}\right)^{-1} = \begin{bmatrix} L^{-1} & \mathbf{0}_{(3n+d)\times n} \\ RL^{-1} & -I_n \end{bmatrix}.$$

Compute the matrix vector product gives us

$$\begin{bmatrix} L^{-1} & \mathbf{0}_{(3n+d)\times n} \\ RL^{-1} & -I_n \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \\ \mathbf{0}_n \end{bmatrix} = \begin{bmatrix} \star \\ RL^{-1} \begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \end{bmatrix} \end{bmatrix}$$

We have

$$RL^{-1}\begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \end{bmatrix} = R\begin{bmatrix} \star \\ \star \\ \sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0}_{n\times(2n+d)} & I_t \mathsf{R} \end{bmatrix} \begin{bmatrix} \star \\ \star \\ \sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U} v \end{bmatrix}$$
$$= R_t \sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U} v$$

where the first step follows that $L^{-1}\begin{bmatrix} \mathbf{0}_{n+d} \\ v \\ v \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U} v \end{bmatrix}$ by Lemma I.2, the second step follows from the definition of R, and the final step comes from the matrix multiplication and $I_t\mathsf{R} = R_t$.

This completes the proof. □

Lemma F.5 (Formal version of Lemma 5.10)**.** *Let* $\mathsf{R} = \begin{bmatrix} R_1^\top & R_2^\top & \cdots & R_T^\top \end{bmatrix} \in \mathbb{R}^{n\times n}$ *be a collection of sketching matrices. Let* $T = \sqrt{n}$*. Define* $\mathsf{B} \in \mathbb{R}^{(3n+d)\times n}$ *to be the following matrix:*

$$\mathsf{B} = \begin{bmatrix} \mathbf{0}_{(n+d)\times n} \\ \mathsf{R}^\top \\ \mathsf{R}^\top \end{bmatrix}$$

Then we have

$$\left(\begin{bmatrix} L & \mathsf{B} \\ \mathbf{0}_{n\times(3n+d)} & -I_n \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{0}_{3n+d} \\ I_t \mathsf{R} v \end{bmatrix} = \begin{bmatrix} \star \\ \sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U} R_t^\top R_t v \end{bmatrix}$$

where I_t *is a diagonal matrix whose* $\frac{n(t-1)}{T}$*th to* $\frac{nt}{T}$*th diagonal entries are* 1 *and other diagonal entries are* 0 *such that* $\mathsf{R}^\top I_t \mathsf{R} = R_t^\top R_t$*.*

Proof. By Lemma 5.8, we know

$$\left(\begin{bmatrix} L & \mathsf{B} \\ \mathbf{0}_{n\times(3n+d)} & -I_n \end{bmatrix} \right)^{-1} = \begin{bmatrix} L^{-1} & L^{-1}\mathsf{B} \\ \mathbf{0}_{n\times(3n+d)} & -I_n \end{bmatrix}.$$

Compute the matrix vector product gives us

$$\begin{bmatrix} L^{-1} & L^{-1}\mathsf{B} \\ \mathbf{0}_{n\times(3n+d)} & -I_n \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3n+d} \\ I_t \mathsf{R} v \end{bmatrix} = \begin{bmatrix} \star \\ L^{-1} R I_t \mathsf{R} v \end{bmatrix}$$

We have

$$L^{-1}\mathsf{B} I_t \mathsf{R} v = L^{-1} \begin{bmatrix} \mathbf{0}_{n+d} \\ \mathsf{R}^\top I_t \mathsf{R} v \\ \mathsf{R}^\top I_t \mathsf{R} v \end{bmatrix}$$
$$= \begin{bmatrix} \star \\ \star \\ \sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U} \mathsf{R}^\top I_t \mathsf{R} v \end{bmatrix}$$
$$= \begin{bmatrix} \star \\ \star \\ \sqrt{U}A^\top (AUA^\top)^{-1} A\sqrt{U} R_t^\top R_t v \end{bmatrix}$$

where the first step follows from the definition of B, the second step follows from Lemma I.2, and the final step follows that $\mathsf{R}^\top I_t \mathsf{R} = R_t^\top R_t$.

This completes the proof. □

G IMPORTANCE SAMPLING AS COORDINATE-WISE EMBEDDING

Lemma G.1 (Formal version of Lemma 6.3)**.** *Let $D \in \mathbb{R}^{n \times n}$ be the sampling matrix defined as above. For any $g \in \mathbb{R}^n$, we have*

- $\mathbb{E}_D[g^\top Dh] = g^\top h$.
- $\mathbb{E}_D[(g^\top Dh)^2] = (g^\top h)^2 + \frac{1}{b}\|g\|_2^2\|h\|_2^2$.
- $\Pr_D[|g^\top Dh - g^\top h| \geq \frac{\log(1/\delta)}{\sqrt{b}}\|g\|_2\|h\|_2] \leq \delta$.

Proof. In expectation, we have

$$\begin{aligned}\mathbb{E}[D_{i,i}] &= p_i \cdot \frac{1}{p_i} + (1-p_i)\cdot 0\\ &= 1\end{aligned}$$

hence, the matrix in expectation is identity.

For variance, we have

$$\begin{aligned}\mathbb{E}[(g^\top Dh)^2] =&~ \mathbb{E}[(\sum_{i=1}^n g_i D_{i,i} h_i)^2]\\ =&~ \mathbb{E}[\sum_{i=1}^n (g_i D_{i,i} h_i)^2 + \sum_{i\neq j} 2 g_i D_{i,i} h_i g_j D_{j,j} h_j]\\ =&~ \underbrace{\sum_{i=1}^n \mathbb{E}[(g_i D_{i,i} h_i)^2]}_{A} + \underbrace{2\sum_{i\neq j} \mathbb{E}[g_i D_{i,i} h_i g_j D_{j,j} h_j]}_{B}.\end{aligned}$$

We bound A (diagonal term) and B (off-diagonal term) separately.

For A, we have

$$\begin{aligned}A =&~ \mathbb{E}[\sum_{i=1}^n g_i^2 D_{i,i}^2 h_i^2]\\ =&~ \sum_{i=1}^n g_i^2\,\mathbb{E}[D_{i,i}^2] h_i^2\\ =&~ \sum_{i=1}^n \frac{1}{p_i} g_i^2 h_i^2\\ =&~ \frac{1}{b}\sum_{i=1}^n \frac{1}{\frac{h_i^2}{\|h\|_2^2} + \frac{1}{n}} g_i^2 h_i^2\\ =&~ \frac{1}{b}\sum_{i=1}^n \frac{n\|h\|_2^2}{nh_i^2 + \|h\|_2^2} g_i^2 h_i^2\\ \leq&~ \frac{1}{b}\sum_{i=1}^n \frac{n\|h\|_2^2}{nh_i^2} g_i^2 h_i^2\\ =&~ \frac{1}{b}\|h\|_2^2\|g\|_2^2\end{aligned}$$

For B, we have

$$B = \mathbb{E}[\sum_{i\neq j} g_i D_{i,i} h_i g_j D_{j,j} h_j]$$

$$\begin{aligned}
&= \sum_{i \neq j} g_i h_i g_j h_j \, \mathbb{E}[D_{i,i} D_{j,j}] \\
&= \sum_{i \neq j} g_i h_i g_j h_j \\
&= \sum_{i \in [n]} g_i h_i \sum_{j \in [n] \backslash \{i\}} g_j h_j \\
&= \sum_{i \in [n]} g_i h_i (\sum_{j \in [n]} g_j h_j - g_i h_i) \\
&= \sum_{i \in [n]} g_i h_i (g^\top h - g_i h_i) \\
&= (g^\top h)^2 - \sum_{i \in [n]} g_i^2 h_i^2 \\
&\leq (g^\top h)^2.
\end{aligned}$$

Put it together, we have

$$\mathbb{E}[(g^\top D h)^2] \leq (g^\top h)^2 + \frac{1}{b} \|g\|_2^2 \|h\|_2^2.$$

For probability, let Y_i denote $g_i D_{i,i} h_i - g_i h_i$, note that $\mathbb{E}[Y_i] = 0$ and the variance of Y_i is

$$\begin{aligned}
\mathbf{Var}[Y_i] =& ~ \mathbb{E}[Y_i^2] \\
&= g_i^2 \, \mathbb{E}[(D_{i,i} - 1)^2] h_i^2 \\
&= g_i^2 h_i^2 (\mathbb{E}[D_{i,i}^2] - 2 \, \mathbb{E}[D_{i,i}] + 1) \\
&= g_i^2 h_i^2 (\frac{1}{p_i} - 1) \\
&= g_i^2 h_i^2 \frac{1}{b} \frac{1}{\frac{h_i^2}{\|h\|_2^2} + \frac{1}{n}} \\
&= g_i^2 h_i^2 \frac{1}{b} \frac{n \|h\|_2^2}{n h_i^2 + \|h\|_2^2} \\
&\leq \frac{1}{b} g_i^2 \|h\|_2^2.
\end{aligned}$$

We also need an absolute value bound:

$$\begin{aligned}
|Y_i| &= |g_i (D_{i,i} - 1) h_i| \\
&\leq |g_i h_i|.
\end{aligned}$$

By Bernstein inequality, we have

$$\begin{aligned}
\Pr[|\sum_{i=1}^n Y_i| \geq \frac{\beta}{\sqrt{b}} \|g\|_2 \|h\|_2] \leq & ~ \exp\left(-\frac{\frac{\beta^2}{b} \|g\|_2^2 \|h\|_2^2}{\frac{1}{b} \|g\|_2^2 \|h\|_2^2 + \max_{i \in [n]} \frac{\beta}{\sqrt{b}} |g_i h_i| \|g\|_2 \|h\|_2 / 3}\right) \\
\leq & ~ \exp\left(-\frac{\frac{\beta^2}{b} \|g\|_2^2 \|h\|_2^2}{\frac{\beta}{b} \|g\|_2^2 \|h\|_2^2}\right) \\
= & ~ \exp(-\beta),
\end{aligned}$$

picking $\beta = \log(1/\delta)$, we obtain the desired result. $\square$

H FASTER EMPIRICAL RISK MINIMIZATION

In this section, we provide an improved algorithm for empirical risk minimization (ERM) via sketching-on-the-right.

H.1 Main Result

Theorem H.1 (Formal version of Theorem 7.1)**.** *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 $i \in [m]$, 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

- *Diameter of the set: For any $x \in \prod_{i=1}^m K_i$, we have that $\|x\|_2 \leq R$.*
- *Lipschitz constant of the program: $\|c\|_2 \leq L$.*

Then, the algorithm MAIN finds a vector x such that

$$c^\top x \leq \min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x + LR\cdot\delta,$$
$$\|Ax-b\|_1 \leq 3\delta\cdot(R\sum_{i,j}|A_{i,j}| + \|b\|_1),$$
$$x \in \prod_{i=1}^m K_i.$$

in time

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

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

Remark H.2. *We get a comparable result as in (Lee et al., 2019), but our algorithm is more light-weighted, since we do not need to restart the algorithm after $\sqrt{n}$ iterations. By sketching on the right, we don't need to worry about large variance caused by sketching on the left.*

H.2 Preliminary

We consider the following optimization problem:

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

where $\prod_{i=1}^m K_i$ is the direct product of m low-dimensional convex sets K_i. Let x_i be the i-th block of x corresponding to K_i, we solve this problem using interior point method, specifically, we consider the following path parametrization:

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

where $\phi_i : K_i \to \mathbb{R}$ is a self-concordant barrier function. This paramerization is the well-known central path.

We assume $n_i = O(1)$ and $n := \sum_{i=1}^m n_i$. The running time will be stated in terms of n.

Self-concordance barrier is defined as follows:

Definition H.3. *We call a function ϕ a ν self-concordant barrier for K if $\mathrm{dom}\phi = K$ and for any $x \in \mathrm{dom}\phi$ and for any $u \in \mathbb{R}^n$*

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

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

Remark H.4. *It is known that $\nu \geq 1$ for any self-concordant barrier function.*

Algorithm 17 Robust IPM algorithm

```
1: procedure RobustIPM($A, b, c, \phi, \delta$)
2:     $\lambda \leftarrow 2^{16} \cdot \log(m), \alpha \leftarrow 2^{-20}\lambda^{-2}, \kappa \leftarrow 2^{-10}\alpha$
3:     $\delta \leftarrow \min(\frac{1}{\lambda}, \delta)$
4:     $\nu = \sum_{i=1}^{m} \nu_i$ where $\nu_i$ are the self-concordant parameters of $\phi_i$.
5:     Modify the convex problem and obtain an initial $x$ and $s$ according to Lemma H.7
6:     $t \leftarrow 1$
7:     while $t > \frac{\delta^2}{4\nu}$ do
8:         Find $\bar{x}$ and $\bar{s}$ such that $\|\bar{x}_i - x_i\|_{x_i} < \alpha$ and $\|\bar{s}_i - s_i\|_{\bar{x}_i}^* < t\alpha$ for all $i$
9:         Find $\widetilde{V}_i$ such that $(1-\alpha)(\nabla^2\phi_i(\bar{x}_i))^{-1} \preceq \widetilde{V}_i \preceq (1+\alpha)(\nabla^2\phi_i(\bar{x}_i))^{-1}$ for all $i$
10:        Compute $h = -\alpha \cdot c_i^t(\bar{x}, \bar{s})\mu_i^t(\bar{x}, \bar{s})$ where:
               $c_i^t(\bar{x}, \bar{s}) = \begin{cases} \frac{\exp(\lambda\gamma_i^t(\bar{x},\bar{s}))/\gamma_i^t(\bar{x},\bar{s})}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t\bar{x},\bar{s})))^{1/2}} & \text{if } \gamma_i^t(\bar{x},\bar{s}) \geq 96\sqrt{\alpha} \\ 0 & \text{otherwise} \end{cases}$
11:        and $\mu_i^t(\bar{x}, \bar{s}) = \bar{s}_i/t + \nabla\phi_i(\bar{x}_i)$ and $\gamma_i^t(\bar{x}, \bar{s}) = \|\mu_i^t(\bar{x}, \bar{s})\|_{\nabla^2\phi_i(\bar{x}_i)^{-1}}$
12:        Let $\widetilde{P} \leftarrow \widetilde{V}^{1/2}A^\top(A\widetilde{V}A^\top)^{-1}A\widetilde{V}^{1/2}$
13:        Compute $\delta_x \leftarrow \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h$ and $\delta_s \leftarrow t \cdot \widetilde{V}^{-1/2}\widetilde{P}\widetilde{V}^{1/2}h$
14:        Move $x \leftarrow x + \delta_x, s \leftarrow s + \delta_s$
15:        $t^{\text{new}} \leftarrow (1 - \frac{\kappa}{\sqrt{\nu}})t$.
16:    end while
17:    Return an approximation solution of the convex problem according to Lemma H.7
18: end procedure
```

H.3 An overview of Robust Central Path

In (Lee et al., 2019), they show how to solve ERM using a so-called *Robust Central Path* framework. Specifically, to follow the path $x(t)$, the optimality condition of Eq. (1):

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

where $\nabla\phi(x) = (\nabla\phi_1(x_1), \dots, \nabla\phi_m(x_m))$. Consider the incurred error in the above formulation: for some $\mu > 0$:

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

where μ is the gap between the original central path and our approximate central path. At each iteration, we will decrease t by a certain amount, and we need to reduce the norm of the gap μ. We follow the Newton step to move μ to $\mu + h$:

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

where $\nabla^2\phi(x)$ is a block diagonal matrix in which each block is $\nabla^2\phi_i(x_i)$. Use W to denote $(\nabla^2\phi(x))^{-1}$, we have the following:

$$\delta_y = -t \cdot (AWA^\top)^{-1}AWh,$$

$$\delta_s = t \cdot A^\top (AWA^\top)^{-1} AWh,$$
$$\delta_x = Wh - WA^\top (AWA^\top)^{-1} AWh.$$

Let $P = W^{1/2} A^\top (AWA^\top)^{-1} AW^{1/2}$ denote the projection matrix, we can rewrite δ_s and δ_x as

$$\delta_s = t \cdot W^{-1/2} P W^{1/2} h,$$
$$\delta_x = W^{1/2} (I - P) W^{1/2} h.$$

To speed up the computation of δ_s and δ_x, we define the following approximate version:

$$\widetilde{\delta}_s = t \cdot W^{-1/2} P R^\top R W^{1/2} h,$$
$$\widetilde{\delta}_x = W^{1/2} (I - P) R^\top R W^{1/2} h,$$

where $R \in \mathbb{R}^{b_{\text{sketch}} \times n}$ is some random sketching matrix. $\widetilde{\delta}_s$ and $\widetilde{\delta}_x$ can be viewed as the solution to the following linear system

$$\begin{aligned} \frac{1}{t} \cdot \widetilde{\delta}_s + \nabla^2 \phi(x) \cdot \widetilde{\delta}_x &= \widetilde{h}, \\ A \widetilde{\delta}_x &= 0, \\ A^\top \widetilde{\delta}_y + \widetilde{\delta}_s &= 0, \end{aligned} \tag{3}$$

where $\widetilde{h} = W^{-1/2} R^\top R W^{1/2} h$. This makes sketching-on-the-right feasible, which means we do not need to restart the algorithm.

The following lemma directly implies our discussion.

Lemma H.5. (x. *and* $\bar{x}$ *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*

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

$\epsilon_x = O(\alpha \log^{2.5}(n/\delta) \cdot \frac{n^{1/4}}{\sqrt{b}})$, *$b$ is the size of sketching matrix.*

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

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

For iteration t, the definition should be

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

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

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

because we have $x_i^{(k_i)} = \bar{x}_i^{(k_i)}$ (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

$$(x_i^{(k)} - \bar{x}_i^{(k)})^\top \cdot (\widetilde{V}_i^{(k)})^{-1} \cdot (x_i^{(k)} - \bar{x}_i^{(k)}) = (\sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)})^\top \cdot (\widetilde{V}_i^{(k)})^{-1} \cdot (\sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)})$$

$$= \| \sum_{t=k_i}^{k} (I - \widetilde{P}^{(t)})((R^{(t)})^\top R^{(t)} - I)(\widetilde{V}^{(t)})^{1/2} h^{(t)})_i \|_2$$

For the following discussion, we fix a block $i \in [m]$ and consider block i and a coordinate $j \in$ block i. We define random vector $X_t \in \mathbb{R}^{n_i}$ as follows:

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

For notation simplicity, we ignore i in subsequent discussion and only use j as the index.

Let $(X_t)_j$ denote the j-th coordinate of X_t, for each $j \in [n_i]$. Since we pick $R^{(t)}$ to pick a coordinate-wise embedding (Def. 6.1), we have for each t,

$$\mathbb{E}[X_t] = 0,$$

for second moment, note

$$\begin{aligned} \mathbb{E}[(X_t)_j^2] &\leq \frac{1}{b} \| e_j - \widetilde{P}_j^{(t)} \|_2^2 \cdot \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2^2 \\ &\leq \frac{4}{b} \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2^2 \end{aligned}$$

and with probability $1 - \delta$,

$$\begin{aligned} |(X_t)_j| &\leq \| e_j - \widetilde{P}_j^{(t)} \|_2 \cdot \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2 \frac{\beta}{\sqrt{b}} \\ &\leq \frac{2\beta}{\sqrt{b}} \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2 \\ &:= M. \end{aligned}$$

Now, we apply Bernstein inequality (Lemma 3.4),

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

Choosing $\tau = 10^3 \frac{\sqrt{T}}{\sqrt{b}} \beta^{5/3} \cdot \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2$,

$$\begin{aligned} & \Pr[\sum_t (X_t)_j > 10^3 \frac{\sqrt{T}}{\sqrt{b}} \beta^{5/3} \cdot \| \widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2] \\ \leq\ & \exp(- \frac{10^6 \frac{T}{b} \beta^{10/3} \cdot \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2^2 / 2}{\frac{4\alpha T}{b} \| (\widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2^2 + 10^3 \frac{\sqrt{T}}{b} \beta^{8/3} \| \widetilde{V}^{(t)})^{1/2} h^{(t)} \|_2^2 / 3}) \\ \leq\ & \exp(-100 \beta^{2/3}) \end{aligned}$$

Now, taking a union, we have

$$\| \sum_{t=k_i}^{k} (I - \widetilde{P}^{(t)})((I - (R^{(t)})^\top R^{(t)})(\widetilde{V}^{(t)})^{1/2} h^{(t)})_i \|_2 = O(\frac{\sqrt{T}}{\sqrt{b}} \beta^{5/3} \| (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i \|_2)$$

$$\begin{aligned}&\leq O(\frac{\sqrt{T}}{\sqrt{b}}\beta^{5/3}\alpha)\\&=O(\frac{\sqrt{T}}{\sqrt{b}}\log^{2.5}(n/\delta)\alpha)\end{aligned}$$

where we use that $\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 \leq \|((\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 = O(\alpha), n_i = O(1)$ and $\beta = \log^{1.5}(n/\delta)$.

This completes the proof. $\square$

A similar proof can be derived regarding $\overline{s}$ and s.

We wrap up this section with several technical lemmas that solves ERM.

Lemma H.6 (Theorem 4.1.7, Lemma 4.2.4 in (Nesterov, 1998)). *Let ϕ be any ν-self-concordant barrier. Then, for any $x, y \in \mathrm{dom}\phi$, we have*

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

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

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

Lemma H.7 (Lemma D.2 in (Lee et al., 2019)). *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 $i \in [m]$, 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:*

- *Diameter of the set: For any $x \in \prod_{i=1}^m K_i$, we have that $\|x\|_2 \leq R$.*
- *Lipschitz constant of the program: $\|c\|_2 \leq L$.*

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

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

satisfies the following:

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

$$c^\top\bar{x}_{1:n} \leq \min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x + LR\cdot\delta,$$
$$\|A\bar{x}_{1:n} - b\|_1 \leq 3\delta\cdot(R\sum_{i,j}|A_{i,j}| + \|b\|_1)$$
$$\bar{x}_{1:n} \in \prod_{i=1}^m K_i$$

Lemma H.8 (Lemma D.3 in (Lee et al., 2019)). *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 $x^* = \arg\min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x$ and $\nu = \sum_{i=1}^m \nu_i$.

I GENERALIZATION TO ASYMMETRIC PROJECTION

We present the original matrix inverse lemma in Section I.1. We present a matrix inverse lemma with extra diagonal matrice in Section I.2.

I.1 The original reduction

Fact I.1. *Given four matrice A, B, C, D, we have the inverse of matrix:*

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}$$

Proof. We need to prove the following equation holds:

$$\begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} = I \tag{4}$$

For the upper left block in Eq. (4) we have:

$$\begin{aligned} & (A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1})A - (A^{-1}B(D - CA^{-1}B)^{-1})C \\ = & I + A^{-1}B(D - CA^{-1}B)^{-1}C - A^{-1}B(D - CA^{-1}B)^{-1}C \\ = & I \end{aligned}$$

For the upper right block in Eq. (4) we have:

$$\begin{aligned} & (A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1})B - (A^{-1}B(D - CA^{-1}B)^{-1})D \\ = & A^{-1}B + A^{-1}B((D - CA^{-1}B)^{-1}(CA^{-1}B - D)) \\ = & A^{-1}B - A^{-1}B \\ = & 0 \end{aligned}$$

For the lower left block in Eq. (4) we have:

$$\begin{aligned} & -(D - CA^{-1}B)^{-1}CA^{-1}A + (D - CA^{-1}B)^{-1}C \\ = & -(D - CA^{-1}B)^{-1}C + (D - CA^{-1}B)^{-1}C \\ = & 0 \end{aligned}$$

For the lower right block in Eq. (4) we have:

$$\begin{aligned} & -(D - CA^{-1}B)^{-1}CA^{-1}B + (D - CA^{-1}B)^{-1}D \\ = & (D - CA^{-1}B)^{-1}(D - CA^{-1}B) \\ = & I \end{aligned}$$

Therefore, we know the Eq. (4) holds and complete the proof. □

Lemma I.2 (Original version)**.** *Let $A \in \mathbb{R}^{d \times n}$ be a matrix of rank d and let $U \in \mathbb{R}^{n \times n}$ be a diagonal matrix with non-zero diagonal entries and let $v \in \mathbb{R}^n$. Then*

$$\begin{bmatrix} U^{-1} & A^\top & \sqrt{U}^{-1} & 0 \\ A & 0 & 0 & 0 \\ 0 & 0 & -I & 0 \\ (\sqrt{U}^{-1})^\top & 0 & 0 & -I \end{bmatrix}^{-1} \begin{bmatrix} 0_n \\ 0_d \\ -v \\ -v \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \\ \sqrt{U}A^\top(AUA^\top)^{-1}A\sqrt{U}v \end{bmatrix} \tag{5}$$

where $\star$ represents some entries that do not care about.

Proof. If $A = U^{-1} \in \mathbb{R}^{n \times n}, B = A^\top \in \mathbb{R}^{n \times d}, C = A \in \mathbb{R}^{d \times n}, D = 0 \in n^{d \times d}$ in Fact I.1, then the matrix has full-rank (i.e. it is invertible) and the top-left block of the inverse is $U - UA^\top (AUA^\top)^{-1} AU \in \mathbb{R}^{n \times n}$. Further, consider the following block-matrix and its inverse:

$$\begin{bmatrix} M & N & 0 \\ 0 & -I & 0 \\ N^\top & 0 & -I \end{bmatrix}^{-1} = \begin{bmatrix} M^{-1} & M^{-1}N & 0 \\ 0 & -I & 0 \\ N^\top M^{-1} & N^\top M^{-1} N & -I \end{bmatrix}$$

Note that

$$M = \begin{bmatrix} U^{-1} & A^\top \\ A & 0 \end{bmatrix}$$

By the Fact I.1 and set $A = U^{-1} \in \mathbb{R}^{n \times n}, B = A^\top \in \mathbb{R}^{n \times d}, C = A \in \mathbb{R}^{d \times n}, D = 0 \in \mathbb{R}^{d \times d}$, then we have

$$M^{-1} = \begin{bmatrix} U + UA^\top (0 - AUA^\top)^{-1} AU & -UA^\top (0 - AUA^\top)^{-1} \\ -(0 - AUA^\top) AU & (0 - AUA^\top)^{-1} \end{bmatrix}$$

When $M \in \mathbb{R}^{(n+d) \times (n+d)}$ is the previous block-matrix and $N \in \mathbb{R}^{(n+d) \times n}$ block-matrix $(\sqrt{U}^{-1}, 0_{n \times d})^\top$, i.e.,

$$N = \begin{bmatrix} \sqrt{U}^{-1} \\ \mathbf{0}_{d \times n} \end{bmatrix}$$

Then we compute the $N^\top M^{-1} N \in \mathbb{R}^{n \times n}$:

$$\begin{aligned} N^\top M^{-1} N &= \sqrt{U}^{-1} (U - UA^\top (AUA^\top)^{-1} AU) \sqrt{U}^{-1} \\ &= I - \sqrt{U} A^\top (AUA^\top)^{-1} A \sqrt{U} \end{aligned}$$

Consider the matrix multiplication in Eq. (5) and its last n coordinates, we have

$$\begin{aligned} \begin{bmatrix} N^\top M^{-1} & N^\top M^{-1} N & -I \end{bmatrix} \begin{bmatrix} 0_{n+d} \\ -v \\ -v \end{bmatrix} &= 0 - N^\top M^{-1} N v + v \\ &= -(I - \sqrt{U} A^\top (AUA^\top) A \sqrt{U}) v + v \\ &= \sqrt{U} A^\top (AUA^\top) A \sqrt{U} v \end{aligned}$$

Therefore we know that Eq. (5) holds and complete the proof. □

I.2 The U and W version

Lemma I.3 (U and W). *Let $A \in \mathbb{R}^{d \times n}$ be a matrix of rank d and let $U \in \mathbb{R}^{n \times n}$ be a diagonal matrix with non-zero diagonal entries and let $v \in \mathbb{R}^n$. Let $W_1 \in \mathbb{R}^{n \times n}$ and $W_2 \in \mathbb{R}^{n \times n}$ be two diagonal matrices. Then*

$$\begin{bmatrix} U^{-1} & A^\top & \sqrt{W_2} U^{-1} & 0 \\ A & 0 & 0 & 0 \\ 0 & 0 & -I & 0 \\ (\sqrt{W_1} U^{-1})^\top & 0 & 0 & -I \end{bmatrix}^{-1} \begin{bmatrix} 0_n \\ 0_d \\ v \\ \frac{\sqrt{W_1 W_2}}{U} v \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \\ \sqrt{W_1} A^\top (AUA^\top)^{-1} A \sqrt{W_2} v \end{bmatrix} \tag{6}$$

where $\star$ represents some entries that do not care about.

Proof. If $A = U^{-1} \in \mathbb{R}^{n\times n}, B = A^\top \in \mathbb{R}^{n\times d}, C = A \in \mathbb{R}^{d\times n}, D = 0 \in n^{d\times d}$ in Fact I.1, then the matrix has full-rank (i.e. it is invertible) and the top-left block of the inverse is $U + UA^\top (AUA)^{-1} A^\top U \in \mathbb{R}^{n\times n}$. Further, consider the following block-matrix and its inverse:

$$\begin{bmatrix} M & N_2 & 0 \\ 0 & -I & 0 \\ N_1^\top & 0 & -I \end{bmatrix}^{-1} = \begin{bmatrix} M^{-1} & M^{-1}N_2 & 0 \\ 0 & -I & 0 \\ N_1^\top M^{-1} & N_1^\top M^{-1} N_2 & -I \end{bmatrix}$$

Note that

$$M = \begin{bmatrix} U^{-1} & A^\top \\ A & 0 \end{bmatrix}$$

By the Fact I.1 and set $A = U^{-1} \in \mathbb{R}^{n\times n}, B = A^\top \in \mathbb{R}^{n\times d}, C = A \in \mathbb{R}^{d\times n}, D = 0 \in \mathbb{R}^{d\times d}$, we have

$$M^{-1} = \begin{bmatrix} U + UA^\top (0 - AUA^\top)^{-1} AU & -UA^\top (0 - AUA^\top)^{-1} \\ -(0 - AUA^\top) AU & (0 - AUA^\top)^{-1} \end{bmatrix}$$

When $M \in \mathbb{R}^{(n+d)\times(n+d)}$ is the previous block-matrix, and $N_1 \in \mathbb{R}^{(n+d)\times n}$, $N_2 \in \mathbb{R}^{(n+d)\times n}$ such that,

$$N_1 = \begin{bmatrix} \sqrt{W_1} U^{-1} \\ \mathbf{0}_{d\times n} \end{bmatrix}, N_2 = \begin{bmatrix} \sqrt{W_2} U^{-1} \\ \mathbf{0}_{d\times n} \end{bmatrix},$$

The bottom-center block of the inverse is

$$\begin{aligned} N_1^\top M^{-1} N_2 &= \sqrt{W_1} U^{-1} (U + UA^\top (AUA^\top)^{-1} AU) U^{-1} \sqrt{W_2} \\ &= \frac{\sqrt{W_1 W_2}}{U} + \sqrt{W_1} A^\top (AUA^\top)^{-1} A \sqrt{W_2} \end{aligned}$$

Consider the matrix multiplication in Eq. (6) and its last n coordinates, we have

$$\begin{aligned} & \begin{bmatrix} N_1^\top M^{-1} & N_1^\top M^{-1} N_2 & -I \end{bmatrix} \begin{bmatrix} 0_{n\times d} \\ v \\ \frac{\sqrt{W_1 W_2}}{U} v \end{bmatrix} \\ &= 0 + N_1^\top M^{-1} N_2 v - \frac{\sqrt{W_1 W_2}}{U} v \\ &= (\frac{\sqrt{W_1 W_2}}{U} + \sqrt{W_1} A^\top (AUA^\top) A^\top \sqrt{W_2}) v - \frac{\sqrt{W_1 W_2}}{U} v \\ &= \sqrt{W_1} A^\top (AUA^\top) A^\top \sqrt{W_2} v \end{aligned}$$

Therefore we know that Eq. (6) holds and complete the proof.

$\square$