On Differential Privacy for Adaptively Solving Search Problems via Sketching

Shiyuan Feng [1] **Ying Feng** [2] **George Z. Li** [3] **Zhao Song** [4] **David P. Woodruff** [3] **Lichen Zhang** [2]

Abstract

Recently differential privacy has been used for a number of streaming, data structure, and dynamic graph problems as a means of hiding the internal randomness of the data structure, so that multiple possibly adaptive queries can be made without sacrificing the correctness of the responses. Although these works use differential privacy to show that for some problems it is possible to tolerate T queries using $\widetilde{O}(\sqrt{T})$ copies of a data structure, such results only apply to *numerical estimation problems*, and only return the *cost* of an optimization problem rather than the solution itself. In this paper, we investigate the use of differential privacy for adaptive queries to *search* problems, which are significantly more challenging since the responses to queries can reveal much more about the internal randomness than a single numerical query. We focus on two classical search problems: nearest neighbor queries and regression with arbitrary turnstile updates. We identify key parameters to these problems, such as the number of c-approximate near neighbors and the matrix condition number, and use different differential privacy techniques to design algorithms returning the solution vector with memory and time depending on these parameters. We give algorithms for each of these problems that achieve similar tradeoffs.

1. Introduction

In modern algorithmic design, data structures are playing a more and more important role, in particular in the recent breakthroughs in optimization, machine learning and graph problems. These highly efficient data structures are usually randomized, with one crucial distinction from traditional data structures: one needs these data structures to be *robust against an adaptive adversary*, in which the adversary could design the input to the data structure based on the prior outputs of the data structure. This is often because these data structures are incorporated in an iterative process, where the data structure takes an input formed by the process, returns a result, and the process utilizes the output to form a new input. Hence, the input to a data structure may be highly correlated with the *internal randomness* of the data structure. Most traditional randomized data structures, unfortunately, are not robust enough to handle this type of task, as they only have probabilistic guarantees against *oblivious adversaries*[1], in which the input sequence to the data structure is predetermined, and independent of its internal randomness.

To address this challenge, various reductions have been proposed to build an adaptive data structure from oblivious data structures (Ben-Eliezer et al., 2020; Cherapanamjeri & Nelson, 2020; 2022; Beimel et al., 2022; Hassidim et al., 2022; Brand et al., 2022; Song et al., 2023b; Bateni et al., 2024). These reductions usually proceed as follows: (1) create k independent copies of the oblivious data structure, (2) during the query phase sample and query a subset of the data structures, and (3) aggregate the subsampled answers.

We focus on the following *search* data structure problem: we are given n points in $\mathbb{R}^d$ represented by a matrix $U \in \mathbb{R}^{n \times d}$ that is allowed to be preprocessed. Given a sequence of adaptive queries or updates $\{v_1, \ldots, v_T\}$, we need to design a data structure that answers queries efficiently and succeeds with high probability. Throughout the paper we will consider two key examples of this: (1) the first type of problem is the approximate near-neighbor problem, where a query is a c-approximate near-neighbor of v_t in U, and (2) the second type of problem is the regression problem, where U is in addition augmented by a response vector $b \in \mathbb{R}^n$, it could be the solution to the regression problem $\arg\min_{x \in \mathbb{R}^d} \|Ux - b\|_2^2$ where either U or b can also be updated by v_t. Note that one can interpret regression as an unconstrained search problem over $\mathbb{R}^d$.

For a sequence of T adaptive queries or updates, perhaps

[1]Peking University [2]MIT [3]Carnegie Mellon University [4]University of California, Berkeley. Correspondence to: Zhao Song <magic.linuxkde@gmail.com>, Lichen Zhang <lichenz@mit.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]In the remainder of the paper, we will refer to data structures that are robust against an adaptive adversary as adaptive data structures, and data structures that are robust against an oblivious adversary as oblivious data structures.

the simplest reduction is to prepare T copies of an oblivious data structure and for each query, use a fresh new data structure to respond. Since a data structure is never used more than once, an adaptive adversary cannot design adversarial inputs to this data structure, and hence we obtain an adaptive data structure. In fact, this simple approach already has led to many improvements in convex optimization (Lee et al., 2019; Song & Yu, 2021; Jiang et al., 2021; Qin et al., 2023). However, in many applications, the overhead T in both preprocessing time and space usage is prohibitive, and more time- and space-efficient alternatives are needed. If in addition the adaptive queries are vectors in $\mathbb{R}^d$ and each data structure succeeds with constant probability, then one can adapt an ε-net argument and show that $\widetilde{O}(d)$ [2] oblivious data structures suffice; at query time, one can then sample a logarithmic number of these data structures and output the the best answer. This approach has been widely adapted for problems such as distance estimation (Cherapanamjeri & Nelson, 2020; 2022) and approximate near-neighbor search (Song et al., 2022; Bateni et al., 2024), but is insufficient if $d \approx T$ or even $d \gg T$, which is the case for convex optimization. Moreover, for many machine learning applications, T is often a hyperparameter that could be chosen and is much smaller than d, making the approach based on ε-net arguments inefficient for these applications.

Is it possible to improve upon the solutions that use $\min\{d, T\}$ data structures? In pioneering work of (Hassidim et al., 2022) and further extended by (Beimel et al., 2022), it was shown that *differential privacy* could be utilized to reduce such overhead. In particular, for numerical estimation problems, where one only needs to output a numerical value, these works show that $\widetilde{O}(\sqrt{T})$ data structures are sufficient. The idea is to view the internal randomness of the oblivious data structures as the private database one would like to protect. Concretely, one prepares $\widetilde{O}(\sqrt{T})$ oblivious data structures, and for each adaptive query, samples $\widetilde{O}(1)$ of the data structures, and outputs the differentially private median of the answers (Beimel et al., 2022). By the advanced composition theorem (Dwork et al., 2010), $\widetilde{O}(\sqrt{T})$ data structures suffice to provide a correct answer to the numerical estimation problem. This leads to drastically improved algorithms when $\sqrt{T} \ll d$, and in (Beimel et al., 2022) they apply it to a large number of graph algorithms. This technique was later extended in (Cherapanamjeri et al., 2023) for estimating the *cost* of a regression problem and the *value* of kernel density estimation.

Can these techniques be extended beyond numerical estimation, to search problems? This is a natural question, as instead of the regression cost, one could naturally be more interested in estimating the regression solution and using it to, e.g., label future examples. On the other hand, simply extending the private median framework to high dimensions seems insufficient, as the regression solution could potentially reveal much more about the internal randomness of a data structure than what only the cost would reveal. This problem becomes more evident for approximate near neighbor search: for this problem, if there exists a row u^* of U for which $\|u^* - v\| \leq r$ for an adaptively chosen query $v \in \mathbb{R}^d$ and a distance parameter r, then the data structure needs to output a row $\widetilde{u}$ of U with $\|\widetilde{u} - v\| \leq cr$ for $c > 1$. It is not clear how to extend the differential privacy framework to such a scenario, as we will have to return a row of U, while the private median mechanism would not be able to do this. Motivated by these problems and barriers, we ask

Is it possible to design robust search data structures with fewer than $\min\{d, T\}$ *independent copies?*

We provide an affirmative answer to this question under mild assumptions on these problems. We start by introducing the definition of the (c, r)-Approximate Near Neighbor ((c, r)-ANN) problem and its corresponding assumption.

Definition 1.1 ((c, r)-Approximate Near Neighbor)**.** *Let $U \subseteq \mathbb{R}^d$ be a dataset, $v \in \mathbb{R}^d$ be a query point, $c > 1$ be the approximation parameter, and $r > 0$ be the distance parameter. The* (c, r)-approximate near neighbor *problem asks, if there exists a point $u_* \in U$ with $\|u_* - v\| \leq r$, then return a point $\widetilde{u} \in U$ with $\|\widetilde{u} - v\| \leq cr$. Otherwise, the data structure can either return nothing or any point $\widetilde{u} \in U$ with $\|\widetilde{u} - v\| \leq cr$.*[3]

We now state the assumption imposed on the ANN problem, in its most general form. Given a query v and a data matrix U, we will use $f_v(U)$ to denote the set of candidate solution of querying v, For example, the predicate function for (c, r)-ANN can be defined as $f_v(U) = U \cap B(v, cr)$ where $B(v, cr)$ is the ball centered at v with radius cr under some norm. The assumption is:

Assumption 1.2. *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset, and let $\{v_1, v_2, \ldots, v_T\} \subseteq \mathbb{R}^d$ be a sequence of (possibly adaptive) queries. We assume for all $t \in [T]$, $|f_{v_t}(U)| \leq s$.*

The assumption states that the ball around v_t of radius cr cannot intersect with more than s points in U, which is equivalent to that the query v_t cannot have too many approximate near neighbors in U, when v_t has an r-near neighbor. To facilitate a comparison with other popular assumptions for this task, such as constant expansion and bounded doubling dimension, we state a version of the assumption for ANN that solely depends on the matrix U:

[2] We use $\widetilde{O}(\cdot)$ to suppress polylogarithmic factors in n, d and $1/\delta$ where δ is the failure probability of the data structure.

[3] The standard (c, r)-ANN definition allows the data structure to output any point in U if there is no point within distance r of the query v. Here we restrict it to either return nothing or return a point within distance cr, as most popular ANN data structures, such as locality-sensitive hashing, satisfy this specification.

Assumption 1.3. *Let $U \subseteq \mathbb{R}^d$, and let $\{B_u\}_{u \in U}$ be the collection of norm balls with $B_u = B(u, cr)$. We assume for each $u \in U$, that B_u intersects at most s other distinct balls in the collection.*

Let us first see how this assumption implies the condition on $f_v(U)$. By the triangle inequality, any two points in $B(v, cr)$ have distance at most $2cr$, that is, the norm balls with these two points being the respective centers with radius cr must intersect. If v has more than s approximate nearest neighbors, then there must exist some $u \in B(v, cr)$ whose norm ball intersects with more than s distinct balls, a contradiction. Intuitively, Assumption 1.3 states that the dataset U is not too dense, and in particular each point in U does not have too many close neighbors. This structural assumption can also be achieved by preprocessing U: one could run a clustering algorithm on U to group points that are close to each other into a cluster, and replace a cluster using its center. Then, the downstream ANN algorithm is performed on these centers; once a center is returned, one could return a point in the corresponding cluster. This approach has been implemented in various approximate nearest neighbor search libraries in practice, and has been a driving force for the Google ScaNN framework (Sun, 2020; Guo et al., 2020). For efficiency reasons, we will restrict $s \leq n^\rho$.

We also compare Assumption 1.3 to two popular assumptions for nearest neighbor search: constant expansion (Karger & Ruhl, 2002; Beygelzimer et al., 2006) and doubling dimension (Gupta et al., 2003; Krauthgamer & Lee, 2004a;b). The constant (local) expansion states that if $|B(v, r) \cap U| \geq \log n$, then $|B(v, 2r) \cap U| \leq c_{\exp} \cdot |B(v, r) \cap U|$ for a constant $c_{\exp}$. The query time of nearest neighbor search data structures under the constant expansion assumption usually has a polylogarithmic dependence on n but a large polynomial dependence on $c_{\exp}$. On the other hand, we only need $|B(v, cr) \cap U| \leq n^\rho$. If we let $|B(v, r) \cap U| = \log n$, then one can see that $|B(v, n^\rho r) \cap U| \leq n^\rho$. Thus, for any $c < n^\rho$ (which is a very large approximation factor), our assumption is strictly weaker. Moreover, we note that constant expansion restricts the growth across different distances, while our assumption only requires a relatively sparse neighborhood at the final level. A more robust version of constant expansion is the notion of doubling dimension, which asks how many balls of radius r are needed to cover a ball of radius $2r$. It has been shown in (Krauthgamer & Lee, 2004b) that bounded doubling dimension is a strictly stronger assumption than constant expansion, and it provides a bound of the form $|B(v, cr) \cap U| \leq \Delta^{\dim(X)}$, where $\Delta \geq 2$ is the aspect ratio of U and X is the metric space. If $X = \mathbb{R}^d$ with any norm, then $\dim(X) = \Theta(d)$, and therefore for this bound to be meaningful, one must have $d = o(\log n)$. Moreover, data structures with a doubling dimension assumption often have their preprocessing time and space exponential in $\dim(X)$.

As a final note, we would like to point out that when working with $U \subseteq \{0, 1\}^d$ and when the norm is $\|\cdot\|_1$, i.e., the Hamming ANN problem, Assumption 1.3 automatically holds for $d = n^\alpha$ and $\alpha \leq \frac{\rho}{cr}$. In particular for Hamming LSH, $\rho = O(1/c)$ (Indyk & Motwani, 1998) and thus we only need $d \leq n^{\Theta(1/(c^2 r))}$. Our result will provide a Hamming LSH that is robust against adaptive adversaries, and in particular against the attack of (Kapralov et al., 2024) in low dimensions.

Next, we consider the problem of adaptively updating the regression problem, and outputting the regression *solution vector* whenever queried.

Assumption 1.4. *Let $U \in \mathbb{R}^{n \times d}$ be the design matrix and $b \in \mathbb{R}^n$ be the response vector. Let $\{v_1, \ldots, v_T\}$ be a sequence of (possibly) adaptive updates to the problem in one of two forms: (1) Update U: $v_t \in \mathbb{R}^{n \times d}$ and U is updated via $U \leftarrow U + v_t$; (2) Update b: $v_t \in \mathbb{R}^n$ and b is updated via $b \leftarrow b + v_t$. We will use (U_t, b_t) to denote the pair after being updated by v_t. The goal is to design a data structure such that for any $t \in [T]$, it outputs a $(1+\alpha)$-approximate solution $x_t \in \mathbb{R}^d$ for which $\|U_t x_t - b_t\|_2^2 \leq (1+\alpha) \cdot \min_{x \in \mathbb{R}^d} \|U_t x - b_t\|_2^2$ holds with high probability. For all $t \in [T]$, we assume the condition number of U_t defined as $\kappa(U_t) := \frac{\sigma_{\max}(U_t)}{\sigma_{\min}(U_t)}$, is upper bounded by κ.*

The problem and assumption can be succinctly described as follows: an adversary could adaptively perturb entries of U and b, and our goal is to design a data structure that outputs the regression solution in the presence of these perturbations. In addition, we assume the perturbations are bounded, in the sense that they do not change the conditioning of the design matrix by too much. For this problem, note that one could naïvely store the matrix U and solve the regression problem exactly. In this case, as the algorithm is deterministic, it is automatically adaptive. However, this approach would require $\Omega(nd)$ space and $O(nd^{\omega-1})$ time, which is both space- and time-inefficient for large n. Alternatively, one could prepare T independent sketching matrices $S_1, \ldots, S_T$, one for each query, and store $S_t U, S_t b$ for all $t \in [T]$. When receiving an update to U or b, one can simply update the corresponding entries of $S_t U$ and $S_t b$, as sketching matrices are linear. This approach requires $\Omega(T \cdot \mathrm{poly}(d))$ space, a preprocessing time of $O(T \cdot (nd + \mathrm{poly}(d)))$, and a query time of $\mathrm{poly}(d)$. While the query time is much more efficient for $n \gg d$, the space and preprocessing time are prohibitive given that T is large. Hence, our goal is to design algorithms that use $o(T \cdot \mathrm{poly}(d))$ space, have a sublinear in T dependence in the preprocessing time, and have a query time of $\mathrm{poly}(d)$.

1.1. Main Results

We state the main results for ANN and adaptive regression in this section.

1.1.1. Adaptive ANN for Sparse Neighborhoods

Our most general result for ANN is as follows:

Theorem 1.5. *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset and let $f_v : (\mathbb{R}^d)^n \to (\mathbb{R}^d)^s$ be the predicate function. Let $\mathcal{A}$ be an oblivious algorithm, i.e., for a fixed query $v \in \mathbb{R}^d$, with probability at least $1-\delta$, $\mathcal{A}$ returns a point (or a subset) of $f_v(U)$ if $f_v(U)$ is non-empty. Moreover, $\mathcal{A}$ has preprocessing time $\mathcal{T}_{\mathrm{prep}}$, space usage $\mathcal{S}_{\mathrm{space}}$ and query time $\mathcal{T}_{\mathrm{query}}$. Then there exists an adaptive algorithm $\widetilde{\mathcal{A}}$ such that given a sequence of adaptive queries $\{v_1, \ldots, v_T\}$ satisfying Assumption 1.2: (1) Preprocesses U in time $\widetilde{O}(\sqrt{T} \cdot s) \cdot \mathcal{T}_{\mathrm{prep}}$; (2) Uses space $\widetilde{O}(\sqrt{T} \cdot s) \cdot \mathcal{S}_{\mathrm{space}}$; (3) For all $t \in [T]$, given query v_t, it returns a point (or a subset) of $f_{v_t}(U)$ in time $\widetilde{O}(s) \cdot \mathcal{T}_{\mathrm{query}}$ with probability at least $1-\delta$. In particular, the amortized cost per query is $\widetilde{O}(s/\sqrt{T}) \cdot \mathcal{T}_{\mathrm{prep}} + \widetilde{O}(s) \cdot \mathcal{T}_{\mathrm{query}}$.*

Let us interpret Theorem 1.5. It states that as long as $s = o(\sqrt{T})$, then we can turn an oblivious data structure into an adaptive data structure using fewer than T independent copies of the oblivious data structure. Of course, this comes at a cost of a slightly worse query time by a factor of $\widetilde{O}(s)$. However, if we consider the amortized cost per query for using T data structures, the cost is dominated by $\mathcal{T}_{\mathrm{prep}}$, as the algorithm needs to prepare a fresh data structure for each query. In contrast, Theorem 1.5 has an amortized cost of $\widetilde{O}(s/\sqrt{T}) \cdot \mathcal{T}_{\mathrm{prep}} + \widetilde{O}(s) \cdot \mathcal{T}_{\mathrm{query}}$ per query. Typically, $\mathcal{T}_{\mathrm{query}}$ is much smaller than $\mathcal{T}_{\mathrm{prep}}$; for example, for LSH, $\mathcal{T}_{\mathrm{prep}} = n^{1+\rho}d$ and $\mathcal{T}_{\mathrm{query}} = n^{\rho}d$, so it is much more important to obtain a reduction on the number of data structures. Applying Theorem 1.5, we immediately obtain adaptive algorithms for LSH under different norms, using fewer than T data structures.

Theorem 1.6. *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset satisfying Assumption 1.3. There exists an adaptive algorithm $\widetilde{\mathcal{A}}$ such that given a sequence of adaptive queries $\{v_1, \ldots, v_T\}$, it solves the (c, r)-ANN problem (Definition 1.1) and (1) Preprocesses U in time $\widetilde{O}(\sqrt{T} \cdot s \cdot n^{1+\rho}d)$; (2) Uses space $\widetilde{O}(\sqrt{T} \cdot s \cdot n^{1+\rho} + nd)$; (3) For all $t \in [T]$, given query v_t, it returns a point in $B(v, cr) \cap U$ if $B(v, r) \cap U \neq \emptyset$ in time $\widetilde{O}(s \cdot n^{\rho}d)$, with probability at least $1-\delta$.*

A generic LSH data structure template restricts the number of points it looks at per query to n^{ρ}, so if $s \leq n^{\rho}$, we obtain the following results:

Corollary 1.7. *Let $U \subseteq \mathbb{R}^d$ satisfy Assumption 1.3 with $s \leq n^{\rho}$. Then there exists an adaptive algorithm $\widetilde{\mathcal{A}}$ such that given a sequence of adaptive queries $\{v_1, \ldots, v_T\}$, the data structure (1) Preprocesses U in time $\widetilde{O}(\sqrt{T} \cdot n^{1+O(\rho)}d)$; (2) Uses space $\widetilde{O}(\sqrt{T} \cdot n^{1+O(\rho)} + nd)$; (3) For all $t \in [T]$, given query v_t, it returns a point in $B(v, cr) \cap U$ if $B(v, r) \cap U \neq \emptyset$ in time $\widetilde{O}(n^{O(\rho)}d)$, with probability at least $1-\delta$.*

Utilizing these adaptive data structures, we obtain improved runtime for problems such as online weighted matching with adversarial arrival, and terminal embeddings. We refer the reader to Section 4 for more details. In addition, when these data structures need to be updated (insert or delete points from the data structures), we provide procedures based on fast rectangular matrix multiplication that beat the alternative of simply updating all data structures (Section C). In Table 1, we compare our result with prior works that use d or T copies of the LSH's.

1.1.2. Adaptive Regression for Well-Conditioned Instances

For adaptive regression, we can use fewer than T copies of an oblivious data structures if the condition number upper bound κ is small:

Theorem 1.8. *Let $U \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$ and $\{v_1, \ldots, v_T\}$ be a sequence of adaptive updates to (U, b), satisfying Assumption 1.4. Then, there exists an adaptive algorithm that (1) Preprocesses (U, b) in time $\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(U) + \mathrm{nnz}(b) + d^3 + d^2\kappa^2/\alpha^2))$; (2) Uses space $\widetilde{O}(\sqrt{T} \cdot d^{2.5}\kappa^2/\alpha^2)$; (3) For all $t \in [T]$, it updates (U_{t-1}, b_{t-1})[4] to (U_t, b_t) in time $\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(v_t) + d^3 + d^2\kappa^2/\alpha^2))$; (4) For all $t \in [T]$, given update v_t, it returns a solution x_t that is a $(1+\alpha)$-approximation to the regression problem using (U_t, b_t) in time $\widetilde{O}(d^{\omega+1}\kappa^2/\alpha^2)$, with probability at least $1-\delta$.*

Theorem 1.8 offers an algorithm with extremely efficient query time. When the condition number bound κ is small, it provides a space bound of $\sqrt{T} \cdot \mathrm{poly}(d)$, both sublinear in T and with no polynomial dependence on n. The preprocessing time outperforms the simple algorithm which generates T sketches when $d \ll T$ and κ is small. While for ANN, one could prepare $\widetilde{O}(d)$ data structures during preprocessing to prepare for all possible queries, and at query time just sample a small number of these, this is not possible for regression. Indeed, for regression one would need to prepare $\widetilde{O}(nd)$ such data structures, given the number of possible design matrices, which would be prohibitive. We again compare our result with prior approaches that use nd or T copies of sketches, in Table 2.

We next study the model in (Cherapanamjeri et al., 2023), where only the response vector b is allowed to be updated in at most s positions. In (Cherapanamjeri et al., 2023), it is shown that one can corporate the techniques of (Beimel et al., 2022) to output the *cost* of the regression problems, but a key open question was whether one could output the actual solution vector to the regression problems. We resolve this question by utilizing the tools we developed in Theorem 1.8 in conjunction with a preconditioner for U to remove the dependence on κ. Below, we will use $\mathrm{nnz}(U)$

[4] We use (U_0, b_0) to denote (U, b), the initial design matrix and response vector.

Method	Space	Amortized Prep Time	Query Time	Update Time
T copies	$Tn^{1+\rho} + nd$	$n^{1+\rho}d$	$n^\rho d$	$Tn^\rho d$
d copies	$n^{1+\rho}d$	$\frac{d}{T}n^{1+\rho}d$	$n^\rho d$	$n^\rho d^2$
This paper	$\sqrt{T}sn^{1+\rho} + nd$	$\frac{s}{\sqrt{T}}n^{1+\rho}d$	$sn^\rho d$	$\sqrt{T}sn^\rho d$

Table 1: Specifications of data structures given a sequence of T adaptive queries and updates for ANN problem. We ignore $\widetilde{O}(\cdot)$ notation for clarity. We use s to denote the parameter for Assumption 1.3.

Method	Space	Amortized Prep Time	Query Time	Update Time
T copies	Td^2/α^2	$\mathrm{nnz}(U,b) + d^3 + d^2/\alpha^2$	$d^{\omega+1}/\alpha^2$	$T(\mathrm{nnz}(v_t) + d^3 + d^2/\alpha^2)$
nd copies	nd^3/α^2	$\frac{nd}{T}(\mathrm{nnz}(U,b) + d^3 + d^2/\alpha^2)$	$d^{\omega+1}/\alpha^2$	$nd(\mathrm{nnz}(v_t) + d^3 + d^2/\alpha^2)$
This paper	$\sqrt{T}d^{2.5}\kappa^2/\alpha^2$	$\sqrt{\frac{d}{T}}(\mathrm{nnz}(U,b) + d^3 + d^2\kappa^2/\alpha^2)$	$d^{\omega+1}\kappa^2/\alpha^2$	$\sqrt{Td}(\mathrm{nnz}(v_t) + d^3 + d^2\kappa^2/\alpha^2)$

Table 2: Specifications of data structures given a sequence of T adaptive queries and updates for regression. We ignore $\widetilde{O}(\cdot)$ notation for clarity. We use $\mathrm{nnz}(U,b)$ as a shorthand for $\mathrm{nnz}(U) + \mathrm{nnz}(b)$, and let κ be the parameter for Assumption 1.4.

to denote the number of nonzero entries in U.

Theorem 1.9. *Let $U \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$. Let $\{v_1, v_2, \dots, \} \subset \mathbb{R}^n$ be a sequence of adaptive updates to b, such that for all t, $\|v_t\|_0 \leq s$. Let T be a batch size parameter. There exists an adaptive algorithm that (1) It has amortized update time $\widetilde{O}(\sqrt{d/T}\cdot(\mathrm{nnz}(U) + \mathrm{nnz}(b) + d^3 + d^\omega/\alpha^2) + \sqrt{Td}\cdot(s + d^3 + d^2/\alpha^2))$; (2) It outputs a $(1+\alpha)$-approximate solution x_t in time $\widetilde{O}(d^2)$.*

The main advantage of Theorem 1.9 over Theorem 1.8 is that the quadratic dependence on the condition number can be removed via choosing a proper preconditioner in this setting, since U is not changing. We also have extremely fast query time, as the solution vector can be quickly updated via a matrix-vector product instead of solving the regression problem from scratch.

When the condition number κ is as large as $\mathrm{poly}(n)$ and U is dynamically changing, Theorem 1.8 no longer gives efficient space and preprocessing time. To address this problem, we develop an algorithm with only *logarithmic* dependence on the condition number κ, based on the bounded computation path technique (Ben-Eliezer et al., 2020).

Theorem 1.10. *Let $U \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$ and $\{v_1, \dots, v_T\}$ be a sequence of adaptive updates to (U,b), satisfying Assumption 1.4. Let $\mathcal{P}$ denote set of possible output sequences the algorithm can provide to the adversary; note that we always have $|\mathcal{P}| \leq (n\kappa)^{\Theta(dT)}$. There exists an adaptive algorithm that (1) Preprocesses (U,b) in time $\widetilde{O}(nd^{\omega-2}(d + \log|\mathcal{P}| + \log\frac{1}{\delta})/\alpha^2)$; (2) Uses space $\widetilde{O}(d(d+\log|\mathcal{P}|+\log\frac{1}{\delta})/\alpha^2)$; (3) For all $t \in [T]$, it updates (U_{t-1}, b_{t-1}) to (U_t, b_t) in time $\widetilde{O}((d+\log|\mathcal{P}|+\log\frac{1}{\delta})/\alpha^2)$; (4) For all $t \in [T]$, it returns a $(1+\alpha)$-approximate solution x_t to the regression problem using (U_t, b_t) in time $\widetilde{O}(d^{\omega-1}(d + \log|\mathcal{P}| + \log\frac{1}{\delta})/\alpha^2)$, with probability at least $1-\delta$.*

Compared to Theorem 1.8, Theorem 1.10 offers a better dependence on the condition number κ at the expense of a linear dependence on the length of update sequence T. While one might be tempted to simply generate an independent sketch for each update, it is worth noting that $|\mathcal{P}|$ could be much smaller than $(n\kappa)^{\Theta(dT)}$ whenever the updates and the solutions are known to be relatively stable. As a concrete example, when only one entry changes in between queries, then the number of computation paths is only $(nd)^T$, which may be much less than $(n\kappa)^{\Theta(dT)}$. In such scenario, it requires smaller number of sketches with improved space, preprocessing and update time. This is similar to past work where the complexity depends on a stability parameter (Hassidim et al., 2022).

2. Related Work

Differential Privacy. Differential privacy is a central concept in data privacy, introduced in (Dwork et al., 2006). The main idea of differential privacy is that when the inputs to the algorithm are close to each other, it would be almost impossible to differentiate the outputs. Since its introduction, differential privacy has seen rich applications in general machine learning (Chaudhuri & Monteleoni, 2008; Williams & McSherry, 2010; Jayaraman & Evans, 2019; Triastcyn & Faltings, 2020), deep neural network (Abadi et al., 2016; Bagdasaryan et al., 2019), computer vision (Zhu et al., 2020; Luo et al., 2021; Tan et al., 2019), natural language processing (Yue et al., 2021; Weggenmann & Kerschbaum, 2018), federated learning (Sun et al., 2023; Song et al., 2023a) and large language model (Yu et al., 2022; Gao et al., 2023; Liang et al., 2024; Gu et al., 2025; Nagesh* et al., 2025). Designing data structures with differential privacy guarantees is crucial, as it automatically ensures the privacy of any downstream tasks (Cohen-Addad et al., 2022; Dhulipala et al., 2023; Li et al., 2023; Chen et al., 2023; Andoni et al.,

2023; Li et al., 2024). It is of particular interest to design differentially private data structure in the *function release model*, where the quality of the output won't degrade as the data structure processes more queries (Hall et al., 2013; Huang & Roth, 2014; Wang et al., 2016; Aldà & Rubinstein, 2017; Coleman & Shrivastava, 2021; Wagner et al., 2023; Backurs et al., 2024; Liu et al., 2024; Hu et al., 2024; Ke et al., 2025; Li et al., 2025).

Adaptive Data Structure. In recent years, data structures have been integrated into iterative processes to speed up the algorithm. This has been the foundation for various recent breakthroughs in fast convex optimization (Cohen et al., 2019; Lee et al., 2019; Brand et al., 2020; 2022; Jiang et al., 2021; Qin et al., 2023). These data structures possess the ability to answer adaptive queries, that could depend on previous outputs from the data structure, with high success probability. For streaming problems, the adversary could also be adaptive, in the sense that it would feed the streaming algorithm with updates, after observing the prior decisions of the algorithm (Ben-Eliezer et al., 2020; Woodruff & Zhou, 2021; Ajtai et al., 2022; Feng & Woodruff, 2023; Woodruff & Zhou, 2024; Feng et al., 2024; Gribelyuk et al., 2024; Gu et al., 2025). In this work, we focus in particular on the adaptive data structures based on differential privacy (Hassidim et al., 2022; Beimel et al., 2022; Song et al., 2023b; Cherapanamjeri et al., 2023).

3. Technical Overview

We divide the technical overview into two parts. For adaptive ANN, we demonstrate a novel framework based on differentially private selection over a sparse vector. For adaptive regression, we show how to upgrade the private median framework of (Beimel et al., 2022) to output the solution vector, and how to obtain utility guarantees through a novel use of ℓ_∞ guarantees provided by the sketch-and-solve framework (Price et al., 2017).

3.1. Adaptive ANN via Differentially Private Selection

Existing Results. Before providing an overview of our techniques, we start by examining existing results for adaptive approximate nearest neighbor search data structures. The first candidates are of course deterministic data structures. Deterministic approximate nearest neighbor data structures have been a central topic of study since the 1970s (de Berg et al., 2008; Cormen et al., 2022). However, without any structural assumptions on the query or dataset, these data structures suffer from the curse of dimensionality, i.e., their preprocessing time and space usage scale exponentially in d, making them infeasible for slightly large dimensions. If one is willing to make strong structural assumptions on the dataset, e.g., given any query point v, the number of points in $B(v, 2r)$ only grows by a constant factor compared to the number of points in $B(v, r)$ (Clarkson, 1997; Karger & Ruhl, 2002; Beygelzimer et al., 2006; Krauthgamer & Lee, 2004b), then it is possible to design a data structure with polynomial preprocessing time and space (note that the dependence on the growth constant or doubling dimension is large, but still polynomial), and logarithmic query time. However, these assumptions are strong, as they greatly limit the potential geometric structure of the dataset.

The celebrated work of Indyk and Motwani (Indyk & Motwani, 1998) shows that if one relaxes the problem to allow for answering c-approximate near neighbor queries instead, then one can achieve (slightly) super-linear preprocessing time and space, and sublinear query time. These data structures are inherently random, as they rely on partitioning the space using random directions. As this randomness is determined during the preprocessing phase, these data structures are not robust against an adaptive adversary. In fact, for Hamming approximate near neighbor search, (Kapralov et al., 2024) provides an efficient attack that can always force the data structure to output a false negative if there exists at least one point that is isolated from other points.

Since the issue of false negatives is most prevalent for Monte Carlo data structures, one might consider to use a class of *Las Vegas* ANN data structures whose running times are random variables, but are guaranteed to output a correct answer with no false negatives (Kushilevitz et al., 1998; Pagh, 2016; Ahle, 2017; Wei, 2022). Unfortunately, the space and runtime analysis of these data structures are performed based on the assumption that the query sequence is *oblivious*. An adaptive adversary could design a sequence of queries with a much higher average response time than an oblivious one.

Reduction To Differentially Private Selection. We start by introducing the differentially private selection problem. Given n categories, a collection of s binary vectors over $\{0,1\}^n$ denoted by $b_{(1)}, \ldots, b_{(s)}$, the goal is to find the category $j^* = \arg\max_{j\in[n]} \sum_{i=1}^{s} b_{(i),j}$, i.e., the most common category among all vectors. This can be achieved via the following mechanism: 1). Compute the overall count vector $B = \sum_{i=1}^{s} b_{(i)}$; 2). Add independent Laplace noise $\mathrm{Lap}(1/\varepsilon)$ to each of the counts; 3). Report the index with the largest noisy count. The privacy parameter ε does not scale with either the number n of categories or the number s of vectors, as it only outputs a single index.

We now show how to frame approximate near neighbor search as a differentially private selection problem. We create a category for each point $u \in U$, and for each data structure, and we ask it to output *all* the near neighbors it finds instead of a single one of them. This creates an indicator vector for the points: if the data structure finds u_i, then the corresponding vector has its i-th entry equal to 1. We can then apply the differentially private selection

procedure for these indicator vectors, and output the point with maximum noisy count. To see this mechanism indeed protects the privacy of the internal randomness of the data structures, note that fixing the dataset U and the adaptive query point v, the indicator vector is determined by the random strings of these data structures. Thus, the mechanism is $(\varepsilon, 0)$-DP, and we can apply the advanced composition theorem of differential privacy to reduce the number of data structures from T to $\widetilde{O}(\sqrt{T})$.

This is, however, not sufficient to provide a utility guarantee, as the Laplace noise could have large magnitude, thus making all counts too noisy. While it is fortunate that we can set $\varepsilon = O(1)$, and with high probability, the magnitude of the noise is $\Theta(\log n)$, this might still be too large to be useful. To see how this issue can be resolved, let us consider the case that the query point v only has one approximate near neighbor, i.e., there exists only one u for which $\|v-u\| \leq r$ and $\|v-u\| \leq cr$. In this case, the data structure can only output u (conditioned on the data structure succeeding). If we were to sample $\omega(\log n)$ data structures and compute the count vector, we can guarantee that the entry corresponding to the near-neighbor has larger magnitude than the noise. Thus, as long as a constant fraction of the data structures succeed, we can ensure we output the correct point with high probability.

Note that our above argument assumes that a constant fraction of data structures succeed, which *is not guaranteed when facing an adaptive adversary*! Fortunately, we can circumvent this problem by proving the algorithm is differentially private first; subsequently, we can apply the generalization property of DP to ensure that indeed, a constant fraction of data structures succeed. Note the two-stage nature of this argument: we can only argue about the utility if the privacy is preserved. There are two main issues left: (1) the assumption that the query has only one near-neighbor is too restrictive, and (2) the differentially private selection procedure takes $O(n)$ time to respond to each query, making the data structure very inefficient. For the first issue, we note that we can extend this to the setting when v has only s approximate near-neighbors. In this setting, we can sample $\omega(s \cdot \log n)$ data structures for each query. By the pigeonhole principle, there must exist at least one point whose count has magnitude larger than the noise, and we obtain the desired utility guarantee. Since the LSH data structures only output n^ρ points for each query, we could pick $s = O(n^\rho)$, i.e., allow the query to have $O(n^\rho)$ near-neighbors. We can alternatively translate this assumption into a structural property on the dataset: as long as the ball centered at each point with radius cr intersects at most s other balls, then we are guaranteed that the query has at most s near-neighbors. We note that this assumption is automatically satisfied for Hamming nearest-neighbor search and for the setting of (Kapralov et al., 2024).

The second issue is algorithmic: if we naïvely implement the differentially private selection mechanism for each query, we will have to generate noise for each count which inevitably takes $\Omega(n)$ time. On the other hand, the count vector B is s-sparse, and these non-zero entries have magnitude larger than the noise. This problem has been studied before in the context of publishing a private database for sparse data (Cormode et al., 2012); however, existing solutions either require modifying the problem definition so that the privacy becomes challenging to prove, or converting a Monte Carlo algorithm into a Las Vegas one, with only expected runtime guarantees. We develop a novel algorithm to resolve these issues termed as the *sparse argmax mechanism*: given an s-sparse vector, (1) Adding s exponential noises $\mathrm{Exp}(1/\varepsilon)$ to the support of the sparse vector; (2) Generating the X from the n-th order statistics distribution of $\mathrm{Exp}(1/\varepsilon)$; (3) Flip a biased coin with head probability $\frac{s}{n}$, if head, generate $s-1$ i.i.d. exponential noises until none of them exceed X, add them including X to the support, and output the maximum entry index; (4) If tail, generate s i.i.d. exponential noises until none of them exceed X, add these noises to the support, assign X a random index in the $n-s$ non-support, output the maximum index associated with the noisy entries and X. We prove that with high probability, generating these noises can be done $O(s \log n)$ time, and the output distribution with sparse noises is the same as the generating n i.i.d. exponential noises.

3.2. Adaptive Regression via Differentially Private Median and ℓ_∞ Guarantee

Existing Results. We first recall the standard setup of solving the over-constrained ℓ_2 regression problem. Given a design matrix $U \in \mathbb{R}^{n \times d}$ with $n \gg d$ and a response vector b, the goal is to compute $x^* := \arg\min_{x \in \mathbb{R}^d} \|Ux - b\|_2^2$. In the static setting, x^* can be computed via the normal equations: $x^* = (U^\top U)^\dagger U b$, but this is usually too expensive to be directly solved. The sketch-and-solve paradigm (see, e.g., (Woodruff, 2014) for a survey) provides a wide array of algorithmic tools to speed up this process. In particular, one can pick a random matrix $S \in \mathbb{R}^{r \times n}$ from a certain structured family of random matrices, so that r is a small polynomial in d, and S can be quickly applied to U. One can then solve the sketched ℓ_2 regression problem $\min_{x \in \mathbb{R}^d} \|SUx - Sb\|_2^2$, for which the optimal solution is a good approximation to x^*. The ℓ_2 regression problem has also been extensively in the streaming (Clarkson & Woodruff, 2009; Braverman et al., 2021) and dynamic (Jiang et al., 2023; Cherapanamjeri et al., 2023) models, where either the design matrix U or the response vector b can be updated, and one has to produce a high quality approximate solution after each update. These works are either not robust to adaptive adversaries (Clarkson & Woodruff, 2009), only support inserting or removing entire rows at once (Braverman et al., 2021;

Jiang et al., 2023), or can only return the cost instead of the solution vector (Cherapanamjeri et al., 2023). In the realistic settings, the design matrix needs to tolerate perturbations to its entries due to the presence of noise or updated data. We consider the most general model where U can be updated by perturbation $v_t \in \mathbb{R}^{n\times d}$ that can be adaptively chosen.

Coordinate-wise Private Median and the ℓ_∞ Guarantee. One natural idea is to extend the private median framework of (Beimel et al., 2022) to outputting an approximation to the solution vector. Our algorithm follows the generic template: prepare k copies of sketching matrices $S_1, \ldots, S_k$ and preprocess (U, b) as $(S_i U, S_i b)$ for all $i \in [k]$. During an adaptive update, we simply update the corresponding sketches of the design matrix and of the response vector. At query time, we sample $s = \widetilde{O}(1)$ of our sketches and solve these sketched regression problems. Let $x_{(1)}, \ldots, x_{(s)}$ be the returned solution vectors. We next need to perform a private aggregation on these vectors to craft our output.

The (Beimel et al., 2022) approach for private aggregation is to compute the private median of these numbers. We first consider a natural extension: for each $i \in [d]$, we compute $g_i = \textsc{PMedian}((x_{(1)})_i, \ldots, (x_{(s)})_i)$ and output $g = (g_1, \ldots, g_d)$. The privacy of this approach is readily established: since each entry of g is private, we can conclude the final privacy guarantee by using the advanced composition theorem. This comes at a price of requiring $\sqrt{Td}$ sketches instead of the $\sqrt{T}$ data structures of (Beimel et al., 2022), but it still offers an improvement as long as $d \ll T$. The main challenge now lies in proving the *utility* of our approach, where we need to show that g is in fact a good enough solution to the regression problem.

To quantify $\|Ug - b\|_2$, note that $\|Ug - b\|_2 \le \|Ux^* - b\|_2 + \|U(x^* - g)\|_2$, where the second term can be bounded by $\|U(x^* - g)\|_2 \le \sigma_{\max}(U) \cdot \|x^* - g\|_2 \le \sigma_{\max}(U)\sqrt{d} \cdot \|x^* - g\|_\infty$. In other words, if we can get a good handle on $\|x^* - g\|_\infty$, then we can hopefully obtain a useful bound on the error. Since g is the coordinate-wise private median of $x_{(1)}, \ldots, x_{(s)}$, which are solutions of sketched ℓ_2 regression problems, the standard sketching error guarantee only ensures that the *forward error* is small, i.e., $\|Ux_{(i)} - b\|_2 \le (1+\alpha)\|Ux^* - b\|_2$ for any $i \in [s]$. For a *backward error* type guarantee, i.e., a bound on $\|x_{(i)} - x^*\|_2$, one could convert directly from the forward error guarantee. If U is reasonably well-conditioned, then closeness in forward error implies closeness in backward error.

However, for the private median guarantee, it is important that each *entry* of $x_{(i)} - x^*$ is small, rather than just the bound that $\|x_{(i)} - x^*\|_2$ is small. While the ideal scenario would be $\|x_{(i)} - x^*\|_\infty \approx \frac{1}{\sqrt{d}}\|x_{(i)} - x^*\|_2$, this is generally not true for most sketching matrices. Fortunately, for sketching matrices such as the Subsampled Randomized Hadamard Transform (SRHT), it has been shown that the sketched solution in fact has a much stronger ℓ_∞ guarantee (Price et al., 2017; Song et al., 2023c): $\|x_{(i)} - x^*\|_\infty \le \frac{\alpha}{\sqrt{d}} \cdot \|Ux^* - b\|_2 \cdot \frac{1}{\sigma_{\min}(U)}$. By properly choosing the parameters for the the private median estimator, we can show that with high probability this also holds for g. Hence, we have $\|U(x^* - g)\|_2 \le \sigma_{\max}(U)\sqrt{d} \cdot \|x^* - g\|_\infty \le \alpha\kappa(U) \cdot \|Ux^* - b\|_2$. To offset the blowup in condition number, we scale down α by a factor of κ, and thus establish the utility of the coordinate-wise private median mechanism. We find the private median to be a surprising application of sketching with the ℓ_∞ guarantee, which is a less studied guarantee in the sketching literature.

To further speed up the preprocessing and update time, we compose the SRHT sketch with Count Sketch matrices (Charikar et al., 2002) so that both operations can be realized in input sparsity time. In addition, these sketches and the sketched design matrices can be stored in $\widetilde{O}(d^2\kappa^2/\alpha^2)$ words of space. Since we use $\widetilde{O}(\sqrt{Td})$ independent sketches, the space usage is $O(\sqrt{T}d^{2.5}\kappa^2/\alpha^2)$.

4. Applications

In this section, we utilize our adaptive ANN data structures to speed up a range of optimization processes and other data structures. Let $\mathcal{T}_{\mathrm{mat}}(a, b, c)$ to be the time complexity of multiplying an $a \times b$ matrix with a $b \times c$ matrix.

4.1. Online Weighted Matching

Consider the following online weighted matching problem on bipartite graphs: we are given a set of left vertices $U \subseteq \mathbb{R}^d$ and we will receive a sequence of online right vertices $V := \{v_1, \ldots, v_n\} \subseteq \mathbb{R}^d$. The edge weight between a left vertex $u \in U$ and a right vertex $v \in V$ is defined to be $\frac{1}{\|u-v\|}$ for some norm $\|\cdot\|$. Our goal is to design an algorithm that given a vertex v_t, make an immediate decision to match with a point $u_t \in U$ with edge weight $\frac{1}{\|u_t - v_t\|}$. The goal is to compute an online matching that maximizes the total weight $\sum_{t=1}^T \frac{1}{\|u_t - v_t\|}$. The online weighted matching problem has been widely studied in the edge-arrival model (Fahrbach et al., 2022), but in many practical machine learning applications, one usually encounters the *vertex-arrival* model (Karp et al., 1990). For example, the left vertices of the graph represent Uber/Lyft drivers, and the right vertices represent customers. The goal is to match each incoming customer with a driver that is closest in geographical distance. Another common application is movie recommendations on streaming platforms such as Netflix. Here the left vertices are movies, and the right vertices are viewers, and the goal is to find each viewer his/her favorite movie, defined by the distance between the feature embeddings of movies and viewers (Koren et al., 2009; Su

& Khoshgoftaar, 2009; Shi et al., 2014). [5]

We consider the vertex-arrival online weighted matching problem when the right vertices are chosen by an *adaptive adversary*, i.e., the next vertex v_{t+1} could depend on previous matching results $\{(u_i, v_i)\}_{i=1}^{t}$. This setting is particularly common for practical applications such as Ride Apps, since if a certain app raises its price due to high demand, the customers might choose to use another app, or move to a new location with smaller demand. A good approximation scheme for this problem is the *greedy matching approach*, i.e., each time we observe v_t, we simply choose u_t to be the point which maximizes $\frac{1}{\|u_t - v_t\|}$ (Karp et al., 1990). One could reduce this problem to a nearest neighbor search problem against an adaptive adversary. There is a caveat though: if we restrict it to be a bipartite matching, then once a vertex u is matched, it should not be matched again. This means our adaptive LSH should delete the point that has been outputted by the data structure after each query. We show that it is possible to further augment our framework for the data structure to be decremental, and the deletion can be performed efficiently using fast rectangular matrix multiplication (Duan et al., 2023). Before proceeding, we let $\theta(a, b)$ for $a, b > 0$ be the value for which $\mathcal{T}_{\mathrm{mat}}(a, b, \theta(a, b)) = (ab)^{1+o(1)}$.

Theorem 4.1 (Online Weighted Matching)**.** *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset satisfying Assumption 1.3 with parameter s. Given a possibly adaptive sequence $\{v_1, \ldots, v_n\} \subseteq \mathbb{R}^d$, we can design an adaptive data structure which (1) Preprocesses U in time $\widetilde{O}(\mathcal{T}_{\mathrm{mat}}(\sqrt{T}\cdot s, d, n) + \sqrt{T}\cdot s \cdot n^{1+\rho})$; (2) Uses space $\mathcal{S}_{\mathrm{space}} = \widetilde{O}(\sqrt{T}\cdot s \cdot n^{1+\rho} + nd)$; (3) Given a point v_t, it returns a point in $U \setminus \{u_1, \ldots, u_{t-1}\}$ that is a $1.1c$-approximate near neighbor of v_t, in time $\widetilde{O}(s\cdot(d+n^\rho))$. This step succeeds with probability at least $1-\delta$; (4) Deletes a point $u \in U$ from the data structure in amortized time $\widetilde{O}((\sqrt{T}\cdot s\cdot d)^{1+o(1)}/\theta(\sqrt{T}\cdot s, d) + \sqrt{T}\cdot s \cdot n^\rho)$.*

To get a better perspective on the deletion time, suppose $s = \mathrm{poly}\log n$. Also note that if $\sqrt{T} \geq d$, then we could simply use d independent copies via a net argument, so we assume $\sqrt{T} \leq d$. Hence, $\theta(\sqrt{T}, d) \geq \theta(\sqrt{T}, \sqrt{T}) \geq T^{\alpha/2}$, where $\alpha \approx 0.32$ is the dual matrix multiplication exponent (Williams et al., 2024; Le Gall, 2024). The runtime can be further simplified to $(T^{1/2-\alpha/2}\cdot d)^{1+o(1)} + T^{1/2}\cdot n^\rho$. We compare this result to updating all $\widetilde{O}(\sqrt{T})$ data structures, which takes a total of $T^{1/2}\cdot d + T^{1/2}\cdot n^\rho$ time. By utilizing fast rectangular matrix multiplication and batch updates, we improve the exponent on $\sqrt{T}$ for the first term.

[5] We note that in movie recommendations, one movie could be matched with multiple viewers. This scenario has also been studied and our results would also apply (Fahrbach et al., 2022).

4.2. Terminal Embedding

A terminal embedding concerns the following problem: given a metric space (X, d_X) and a set of points $u_1, \ldots, u_n \in X$, the goal is to design an embedding to another metric space (Y, d_Y) such that for any $q \in X$, $C\cdot d_X(u_i, q) \leq d_Y(u_i, q) \leq C\rho \cdot d_X(u_i, q)$ for all $i \in [n]$, where $C > 0$ is a constant and $\rho \geq 1$ is the distortion factor. In contrast to metric embeddings such as the Johnson-Lindenstrauss lemma, a terminal embedding requires the distance to be preserved between a fixed set of terminals and *all points* in the metric space. When both X and Y are the Euclidean metric, it has been shown that an embedding dimension of $O(\varepsilon^{-2}\log n)$ is possible for $1+\varepsilon$ distortion. (Cherapanamjeri & Nelson, 2021) shows that it is possible to implement a data structure that answers queries in time $O(n^{1-\Theta(\varepsilon^2)} + d)$ and space $O(dn^{1+o(1)})$. At the core of their algorithm is an adaptive ANN data structure. In particular, they create $\widetilde{O}(d)$ independent copies to deal with issues of adaptivity. If you know in advance that you only need to answer T adaptive queries for $T \leq d$, then the $\widetilde{O}(d)$ copies via a net argument is sub-optimal. For a fair comparison, in the following we will assume (Cherapanamjeri & Nelson, 2021) uses T copies of data structures instead of d. We apply the adaptive LSH data structure we have designed to obtain an improvement in the space complexity of (Cherapanamjeri & Nelson, 2021)'s data structure when $\sqrt{T}\cdot s \leq d$ and the query points q satisfy Assumption 1.2. Before proceeding, we need to introduce a few concepts.

Definition 4.2. *Given $U = \{u_1, \ldots, u_n\} \subseteq \mathbb{R}^d$ and $\varepsilon > 0$, we say a matrix $S \in \mathbb{R}^{k\times d}$ is a convex hull distortion for U if for any $z \in \mathrm{conv}(T)$ where $T = \{\frac{u-v}{\|u-v\|} : u, v \in U\}$, we have $|\|Sz\| - \|z\|| \leq \varepsilon$.*

Theorem 4.3 (Terminal Embedding)**.** *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset and $V = \{v_1, \ldots, v_T\} \subseteq \mathbb{R}^d$ be a sequence of T adaptive queries that satisfy Assumption 1.2 with parameter s. Let $\rho_1, \rho_2, \rho_3, \rho_4$ and $\rho_{\mathrm{rep}} > 0$ be parameters. There exists a randomized algorithm that computes a data structure $\mathcal{D}$ and a linear map $S \in \mathbb{R}^{k\times d}$, such that (1) S is a convex hull distortion for U; (2) Given any $v_t \in V$, $\mathcal{D}$ produces a vector $z_{v_t} \in \mathbb{R}^{k+1}$ such that with probability at least $1 - 1/\mathrm{poly}(n)$, $(1-\varepsilon)\cdot\|v_t - u\| \leq \left\|z_{v_t} - \begin{bmatrix} Su \\ 0 \end{bmatrix}\right\| \leq (1+\varepsilon)\cdot\|v_t - u\|$ for all $u \in U$. (3) For any $v_t \in V$, the runtime of computing z_{v_t} is $\widetilde{O}(s\cdot(d + n^{\rho_2} + n^{\rho_4} + n^{\rho_4 + (1+\rho_3-\rho_4-\rho_{\mathrm{rep}})\rho_2}))$; (4) The space complexity of $\mathcal{D}$ is $O(\sqrt{T}\cdot s\cdot(n^{\rho_{\mathrm{rep}}+(1+\rho_1)} + n^{\rho_3+(1+\rho_1)}))$.*

To interpret our result, we note that the query time is increased by a factor of s, as in all previous results. The space complexity is reduced from T to $\sqrt{T}\cdot s$. If $s = \mathrm{poly}\log n$, then the query time is only increased by a polylogarithmic factor, and the space complexity is improved by a factor of $\sqrt{T}$.

Acknowledgement

Ying Feng was supported by an MIT Akamai Presidential Fellowship. David Woodruff would like to acknowledge support from a Simons Investigator Award and Office of Naval Research (ONR) award number N000142112647. Lichen Zhang was supported in part by NSF CCF-1955217 and NSF DMS-2022448.

Impact Statement

This paper develops novel adaptive data structures to speed up machine learning tasks. It potentially would lead to efficiency improvement and reduction in carbon emission. We do not foresee any societal consequences worth highlighting.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Ahle, T. D. Optimal las vegas locality sensitive data structures. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 938–949. IEEE, 2017.

Ajtai, M., Braverman, V., Jayram, T., Silwal, S., Sun, A., Woodruff, D. P., and Zhou, S. The white-box adversarial data stream model. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pp. 15–27. ACM, 2022.

Aldà, F. and Rubinstein, B. I. The bernstein mechanism: function release under differential privacy. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 1705–1711. AAAI Press, 2017.

Andoni, A., Laarhoven, T., Razenshteyn, I., and Waingarten, E. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 47–66. SIAM, 2017.

Andoni, A., Indyk, P., Mahabadi, S., and Narayanan, S. Differentially private approximate near neighbor counting in high dimensions. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Backurs, A., Lin, Z., Mahabadi, S., Silwal, S., and Tarnawski, J. Efficiently computing similarities to private datasets. In *The Twelfth International Conference on Learning Representations*, 2024.

Bagdasaryan, E., Poursaeed, O., and Shmatikov, V. Differential privacy has disparate impact on model accuracy. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:15479–15488, 2019.

Bateni, M., Dhulipala, L., Fletcher, W., Gowda, K. N., Hershkowitz, D. E., Jayaram, R., and Lacki, J. Efficient centroid-linkage clustering. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, 2024.

Beimel, A., Kaplan, H., Mansour, Y., Nissim, K., Saranurak, T., and Stemmer, U. Dynamic algorithms against an adaptive adversary: Generic constructions and lower bounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1671–1684, 2022.

Ben-Eliezer, O., Jayaram, R., Woodruff, D. P., and Yogev, E. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, pp. 63–80, 2020.

Beygelzimer, A., Kakade, S., and Langford, J. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pp. 97–104, 2006.

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

Brand, J. v. d., Gao, Y., Jambulapati, A., Lee, Y. T., Liu, Y. P., Peng, R., and Sidford, A. Faster maxflow via improved dynamic spectral vertex sparsifiers. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pp. 543–556, New York, NY, USA, 2022. Association for Computing Machinery.

Braverman, V., Hassidim, A., Matias, Y., Schain, M., Silwal, S., and Zhou, S. Adversarial robustness of streaming algorithms through importance sampling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NeurIPS '21, 2021.

Bun, M., Nissim, K., Stemmer, U., and Vadhan, S. Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 634–649, 2015.

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

Chaudhuri, K. and Monteleoni, C. Privacy-preserving logistic regression. In *NIPS*, volume 8, pp. 289–296. Citeseer, 2008.

Chen, J. Y., Ghazi, B., Kumar, R., Manurangsi, P., Narayanan, S., Nelson, J., and Xu, Y. Differentially private all-pairs shortest path distances: Improved algorithms and lower bounds. In *SODA*, 2023. Merge of two independent and concurrent works by Chen, Narayanan, and Xu and by Ghazi, Kumar, Manurangsi, and Nelson.

Cherapanamjeri, Y. and Nelson, J. On adaptive distance estimation. *Advances in Neural Information Processing Systems*, 33:11178–11190, 2020.

Cherapanamjeri, Y. and Nelson, J. Terminal embeddings in sublinear time. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2021.

Cherapanamjeri, Y. and Nelson, J. Uniform approximations for randomized hadamard transforms with applications. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2022.

Cherapanamjeri, Y., Silwal, S., Woodruff, D., Zhang, F., Zhang, Q., and Zhou, S. Robust algorithms on adaptive inputs from bounded adversaries. In *The Eleventh International Conference on Learning Representations*, 2023.

Clarkson, K. L. Nearest neighbor queries in metric spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pp. 609–617, 1997.

Clarkson, K. L. and Woodruff, D. P. Numerical linear algebra in the streaming model. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, 2009.

Cohen, M. B., Lee, Y. T., and Song, Z. Solving linear programs in the current matrix multiplication time. In *STOC*, 2019.

Cohen-Addad, V., Epasto, A., Mirrokni, V., Narayanan, S., and Zhong, P. Near-optimal private and scalable k-clustering. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 1–13, 2022.

Coleman, B. and Shrivastava, A. A one-pass distributed and private sketch for kernel sums with applications to machine learning at scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, pp. 3252–3265, New York, NY, USA, 2021. Association for Computing Machinery.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. MIT press, 2022.

Cormode, G., Procopiuc, C., Srivastava, D., and Tran, T. T. L. Differentially private summaries for sparse data. In *Proceedings of the 15th International Conference on Database Theory*, ICDT '12, pp. 299–311, New York, NY, USA, 2012. Association for Computing Machinery.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry (SoCG)*, pp. 253–262, 2004.

de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

Dhulipala, L., Li, G. Z., and Liu, Q. C. Near-optimal differentially private k-core decomposition. *arXiv preprint arXiv:2312.07706*, 2023.

Ding, Z., Kifer, D., E., S. M. S. N., Steinke, T., Wang, Y., Xiao, Y., and Zhang, D. The permute-and-flip mechanism is identical to report-noisy-max with exponential noise, 2021. URL https://arxiv.org/abs/2105.07260.

Duan, R., Wu, H., and Zhou, R. Faster matrix multiplication via asymmetric hashing. In *FOCS*, 2023.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 2014.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pp. 265–284. Springer, 2006.

Dwork, C., Rothblum, G. N., and Vadhan, S. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 51–60. IEEE, 2010.

Fahrbach, M., Huang, Z., Tao, R., and Zadimoghaddam, M. Edge-weighted online bipartite matching. *Journal of the ACM*, 69(6):1–35, 2022.

Feng, Y. and Woodruff, D. P. Improved algorithms for white-box adversarial streams. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 9962–9975. PMLR, 2023.

Feng, Y., Jain, A., and Woodruff, D. P. Fast white-box adversarial streaming without a random oracle. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13199–13224. PMLR, 2024.

Gao, Y., Song, Z., and Yang, X. Differentially private attention computation. *arXiv preprint arXiv:2305.04701*, 2023.

Gathen, J. v. z. and Gerhard, J. *Fast polynomial evaluation and interpolation*, pp. 295–312. Cambridge University Press, 2013.

Gribelyuk, E., Lin, H., Woodruff, D. P., Yu, H., and Zhou, S. A strong separation for adversarially robust ℓ_0 estimation for linear sketches. In *Proceedings of the 65th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2024.

Gu, J., Liang, Y., Sha, Z., Shi, Z., and Song, Z. Differential privacy mechanisms in neural tangent kernel regression. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2025.

Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning (ICML)*, pp. 3887–3896. PMLR, 2020.

Gupta, A., Krauthgamer, R., and Lee, J. R. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 534–543. IEEE, 2003.

Hall, R., Rinaldo, A., and Wasserman, L. Differential privacy for functions and functional data. *J. Mach. Learn. Res.*, 2013.

Hassidim, A., Kaplan, H., Mansour, Y., Matias, Y., and Stemmer, U. Adversarially robust streaming algorithms via differential privacy. *J. ACM*, 69(6), 2022.

Hu, J. Y.-C., Liu, E., Liu, H., Song, Z., and Zhang, L. On differentially private string distances. *arXiv preprint arXiv:2411.05750*, 2024.

Huang, Z. and Roth, A. Exploiting metric structure for efficient private query release. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pp. 523–534, 2014.

Indyk, P. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 2006.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*, pp. 604–613, 1998.

Jayaraman, B. and Evans, D. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1895–1912, 2019.

Jiang, S., Song, Z., Weinstein, O., and Zhang, H. A faster algorithm for solving general lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 823–832, 2021.

Jiang, S., Peng, B., and Weinstein, O. The complexity of dynamic least-squares regression. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, Los Alamitos, CA, USA, 2023.

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

Kapralov, M., Makarov, M., and Sohler, C. On the adversarial robustness of locality-sensitive hashing in hamming space, 2024.

Karger, D. R. and Ruhl, M. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, pp. 741–750, 2002.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pp. 352–358, 1990.

Ke, Y., Liang, Y., Sha, Z., Shi, Z., and Song, Z. Dpbloomfilter: Securing bloom filters with differential privacy. *arXiv preprint arXiv:2502.00693*, 2025.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009.

Krauthgamer, R. and Lee, J. R. The black-box complexity of nearest neighbor search. In *International Colloquium on Automata, Languages, and Programming*, pp. 858–869. Springer, 2004a.

Krauthgamer, R. and Lee, J. R. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pp. 798–807. Citeseer, 2004b.

Kushilevitz, E., Ostrovsky, R., and Rabani, Y. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 614–623, 1998.

Le Gall, F. Faster rectangular matrix multiplication by combination loss analysis. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 3765–3791. SIAM, 2024.

Lee, Y. T., Song, Z., and Zhang, Q. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*, 2019.

Li, G. Z., Nguyen, D., and Vullikanti, A. Differentially private partial set cover with applications to facility location. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)*, pp. 4803–4811. International Joint Conferences on Artificial Intelligence Organization, 2023.

Li, G. Z., Nguyen, D., and Vullikanti, A. Computing epidemic metrics with edge differential privacy. In Dasgupta, S., Mandt, S., and Li, Y. (eds.), *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4303–4311. PMLR, 2024.

Li, X., Liang, Y., Shi, Z., Song, Z., and Yu, J. Fast john ellipsoid computation with differential privacy optimization. In *Conference on Parsimony and Learning*. PMLR, 2025.

Liang, Y., Shi, Z., Song, Z., and Zhou, Y. Differential privacy of cross-attention with provable guarantee. *arXiv preprint arXiv:2407.14717*, 2024.

Liu, E., Hu, J. Y.-C., Reneau, A., Song, Z., and Liu, H. Differentially private kernel density estimation. *arXiv preprint arXiv:2409.01688*, 2024.

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

Luo, Z., Wu, D. J., Adeli, E., and Li, F.-F. Scalable differential privacy with sparse network finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5059–5068, 2021.

Nagesh*, S., Chen*, J. Y., Mishra, N., and Wagner, T. Private text generation by seeding large language model prompts. In *Preprint*, 2025.

Pagh, R. Locality-sensitive hashing without false negatives. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 1–9. SIAM, 2016.

Price, E., Song, Z., and Woodruff, D. P. Fast regression with an ℓ_∞ guarantee. In *ICALP*, 2017.

Qin, L., Song, Z., Zhang, L., and Zhuo, D. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In *AISTATS*, 2023.

Shi, Y., Larson, M., and Hanjalic, A. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.

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

Song, Z., Xu, Z., and Zhang, L. Speeding up sparsification using inner product search data structures, 2022.

Song, Z., Wang, Y., Yu, Z., and Zhang, L. Sketching for first order method: efficient algorithm for low-bandwidth channel and vulnerability. In *International Conference on Machine Learning (ICML)*, pp. 32365–32417. PMLR, 2023a.

Song, Z., Yang, X., Yang, Y., and Zhang, L. Sketching meets differential privacy: fast algorithm for dynamic kronecker projection maintenance. In *International Conference on Machine Learning (ICML)*, pp. 32418–32462. PMLR, 2023b.

Song, Z., Ye, M., Yin, J., and Zhang, L. A nearly-optimal bound for fast regression with ℓ_∞ guarantee. In *International Conference on Machine Learning (ICML)*, pp. 32463–32482. PMLR, 2023c.

Su, X. and Khoshgoftaar, T. M. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

Sun, J., Yang, X., Yao, Y., Xie, J., Wu, D., and Wang, C. Dpauc: differentially private auc computation in federated learning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

Sun, P. Announcing scann: Efficient vector similarity search. https://blog.research.google/2020/07/announcing-scann-efficient-vector.html, 2020.

Tan, S., Zhou, Z., Xu, Z., and Li, P. On efficient retrieval of top similarity vectors. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5239–5249, 2019.

Triastcyn, A. and Faltings, B. Bayesian differential privacy for machine learning. In *International Conference on Machine Learning*, pp. 9583–9592. PMLR, 2020.

Wagner, T., Naamad, Y., and Mishra, N. Fast private kernel density estimation via locality sensitive quantization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Wang, Z., Jin, C., Fan, K., Zhang, J., Huang, J., Zhong, Y., and Wang, L. Differentially private data releasing for smooth queries. *Journal of Machine Learning Research*, 2016.

Weggenmann, B. and Kerschbaum, F. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 305–314, 2018.

Wei, A. Optimal las vegas approximate near neighbors in ℓ_p. *ACM Transactions on Algorithms (TALG)*, 18(1):1–27, 2022.

Williams, O. and McSherry, F. Probabilistic inference and differential privacy. *Advances in Neural Information Processing Systems (NeurIPS)*, 23:2451–2459, 2010.

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

Williams, V. V., Xu, Y., Xu, Z., and Zhou, R. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 3792–3835. SIAM, 2024.

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

Woodruff, D. P. and Zhou, S. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *Proceedings of the 62nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1183–1196. IEEE, 2021.

Woodruff, D. P. and Zhou, S. Adversarially robust dense-sparse tradeoffs via heavy-hitters. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., Kulkarni, J., Lee, Y. T., Manoel, A., Wutschitz, L., Yekhanin, S., and Zhang, H. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022.

Yue, X., Du, M., Wang, T., Li, Y., Sun, H., and Chow, S. S. M. Differential privacy for text analytics via natural text sanitization. In *Findings, ACL-IJCNLP 2021*, 2021.

Zhu, Y., Yu, X., Chandraker, M., and Wang, Y.-X. Private-knn: Practical differential privacy for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11854–11862, 2020.

Appendix

A. Preliminaries

We use $\widetilde{O}(\cdot)$ to suppress $\mathrm{poly}\log$ factors in $n, d, 1/\delta$. We use $\mathbb{E}[\cdot]$ and $\Pr[\cdot]$ to denote expectation and probability. We use $\mathcal{T}_{\mathrm{mat}}(n, m, d)$ to denote the time to multiply an $n \times m$ matrix with an $m \times d$ matrix. For a positive integer n, we use $[n]$ to denote the set $[n] := \{1, 2, \cdots, n\}$. For a vector x, we use $x^\top$ to denote the transpose of x. For a vector $x \in \mathbb{R}^n$, we use $\|x\|_2$ to denote its ℓ_2 norm, i.e., $\|x\|_2 := (\sum_{i=1}^n x_i^2)^{1/2}$. For two real numbers $a, b > 0$, we will use $a = (1 \pm \alpha)b$ to denote $a \in [(1-\alpha)b, (1+\alpha)b]$ for $\alpha > 0$.

For a point $v \in \mathbb{R}^d$ and $r > 0$, we use $B_{\|\cdot\|}(v, r)$ to denote the ball centered at v of radius r in some norm. When the norm $\|\cdot\|$ is clear from the context, we abbreviate this with $B(v, r)$. We use $\mathbb{S}^{d-1}$ to denote the unit sphere of dimension d. Let $(\Omega, \mathcal{F})$ be a measurable space and let probability measures P, Q be defined on $(\Omega, \mathcal{F})$. The total variation (TV) distance between P and Q is defined to be $\mathrm{d_{TV}}(P, Q) = \sup_{A \in \mathcal{F}} |P(A) - Q(A)|$.

We will be using exponential random variables extensively, and we recall its definition here.

Definition A.1 (Exponential Distribution)**.** *The* exponential distribution with parameter λ*, denoted by* $\mathrm{Exp}(\lambda)$*, is the distribution with PDF*

$$f(x; \lambda) = \begin{cases} \lambda e^{-\lambda x}, & \text{if } \lambda \geq 0, \\ 0, & \text{otherwise}. \end{cases}$$

We will also work with order statistics.

Definition A.2 (Order Statistics)**.** *Let $X_1, \ldots, X_n$ be n i.i.d. random samples, the* k-th order statistics*, denoted by $X_{(k)}$, is the random variable associated with the k-th smallest value over the samples.*

For extreme order statistics such as $X_{(1)}$ and $X_{(n)}$, they could be generated efficiently via inverse CDF sampling.

Fact A.3. *Let $n \geq 1$ be a positive integer, suppose we can generate $U \sim \mathrm{Unif}(0, 1)$ in $O(1)$ time, then for any distribution with a closed-form CDF distribution, $X_{(1)}$ and $X_{(n)}$ can be generated in $O(1)$ time.*

B. Augmenting Oblivious ANN Data Structures with Differential Privacy

We develop a generic algorithm that takes an oblivious ANN data structure, and transforms it to one that is adaptive. We will start with a simple but inefficient algorithm, and improve its efficiency afterwards.

We require these data structures to output all answers they find (within their runtime budget, which we will formalize) instead of outputting only a single answer. In the context of LSH, this means that instead of outputting one approximate near neighbor, it needs to output all approximate near neighbors it finds. As the runtime budget for a query is n^ρ, it can output at most n^ρ approximate near neighbors.

Before introducing the algorithm, we give some background in differential privacy.

B.1. Preliminaries on Differential Privacy

Differential privacy refers to a class of algorithms for which, when one slightly perturbs the input, the output distribution remains relatively close. This intuitively will mean that for an adversary, it cannot learn useful information by adaptively adjusting its input to an algorithm.

Definition B.1 (Differential Privacy)**.** *We say a randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for any two databases S and S' that differ only by one row and any subset of outputs T, the following*

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

holds. Here the probability is over the randomness of $\mathcal{A}$.

Differential privacy can be composed, and the advanced composition theorem says that k-fold adaptive composition only blows up the ε parameter by a factor of roughly $\sqrt{k}$ instead of k. Similar to (Beimel et al., 2022), we will utilize advanced composition to reduce the number of oblivious data structures required and gain runtime improvements.

Theorem B.2 (Advanced Composition, see (Dwork et al., 2010))**.** *Given three parameters $\varepsilon, \delta_0, \delta \geq 0$. If $\mathcal{A}_1, \cdots, \mathcal{A}_k$ are each (ε, δ)-DP algorithms, then the k-fold adaptive composition $\mathcal{A}_k \circ \cdots \circ \mathcal{A}_1$ is $(\varepsilon_0, \delta_0 + k\delta)$-DP where*

$$\varepsilon_0 = \sqrt{2k \ln(1/\delta_0)} \cdot \varepsilon + 2k\varepsilon^2$$

One can boost the privacy budget by subsampling the dataset first, then running the differentially private algorithm on the subsampled set. We state a version where $\delta = 0$. Note that it can be easily extended to account for non-zero δ.

Theorem B.3 (Amplification via Sampling (Lemma 4.12 of (Bun et al., 2015)))**.** *Let $\varepsilon \in (0, 1]$ be parameters. Let $\mathcal{A}$ denote an $(\varepsilon, 0)$-DP algorithm. Let S denote a dataset.*

Suppose that $\mathcal{A}'$ is an algorithm that,

- *constructs a database $T \subset S$ by subsampling with repetition $k \leq n/2$ rows from S,*
- *returns $\mathcal{A}(T)$.*

Then, we have

$$\mathcal{A}' \quad \text{is} \quad \left(\frac{6k}{n}\varepsilon, 0\right) - \text{DP}.$$

Theorem B.4 (Generalization of Differential Privacy (DP))**.** *Given two accuracy parameters $\varepsilon \in (0, 1/3)$ and $\delta \in (0, \varepsilon/4)$, suppose that the parameter t satisfies that $t \geq \varepsilon^{-2} \log(2\varepsilon/\delta)$.*

We use $\mathcal{D}$ to represent a distribution over a domain $\mathcal{X}$. Suppose $S \sim \mathcal{D}^t$ is a database containing t elements sampled independently from $\mathcal{D}$. Let $\mathcal{A}$ be an algorithm that, given any database S of size t, outputs a predicate $h : \mathcal{X} \to \{0, 1\}$.

If $\mathcal{A}$ is (ε, δ)-DP, then the empirical average of h on sample S, i.e.,

$$h(S) = \frac{1}{|S|} \sum_{x \in S} h(x),$$

and h's expectation over underlying distribution $\mathcal{D}$, i.e.,

$$h(\mathcal{D}) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[h(x)]$$

are within 10ε with probability at least $1 - \delta/\varepsilon$:

$$\Pr_{S \sim \mathcal{D}^t, h \leftarrow \mathcal{A}(S)} \left[\left|h(S) - h(\mathcal{D})\right| \geq 10\varepsilon\right] \leq \delta/\varepsilon.$$

We next consider the task of differentially private selection: we have a discrete space of outcomes, and the goal is to output the largest histogram cell:

Definition B.5 (Differentially Private Selection)**.** *Given n categories, let the database S be such that, each of its rows is a binary vector of length n, with each entry corresponding to whether the row belongs to the i-th category or not. We say an algorithm is an $(\varepsilon, 0)$-*differentially private selection *if it releases the index of the approximately most popular category over the database S, and is $(\varepsilon, 0)$-DP.*

For any differentially private mechanism, one cares about privacy but also *utility*, i.e., after adding noise, how good are the estimates compared to the algorithm without adding noise? Here, we focus on privacy, and we will later use the privacy to provide utility guarantees. For count queries, the most common approach is the *Laplace mechanism*, where properly chosen Laplace noises are added to each count. Here, we use a similar approach that instead adds *exponential* noises. This approach is more widely adopted for Exponential Mechanism, yet it still offers the same privacy budget as that of the standard Laplace Mechanism for reporting the noisy max index. See e.g., (Dwork & Roth, 2014; Ding et al., 2021) for more details.

Theorem B.6 (REPORTONESIDEDNOISYARGMAX, Theorem 3.13 of (Dwork & Roth, 2014))**.** *Given a database S with each row being a binary vector of length n, consider the following algorithm:*

- *Compute the count for each category;*
- *Add an i.i.d. exponential noise* $\mathrm{Exp}(1/\varepsilon)$ *to each count;*
- *Output the category with the maximum noisy count.*

The algorithm gives $(\varepsilon, 0)$*-differentially private selection.*

B.2. Reducing Monte Carlo Search to Selection: A Slow Algorithm

In this section, we show how to utilize the REPORTNOISYMAX procedure for adaptive data structure design. Recall the framework introduced in (Hassidim et al., 2022) for streaming and later extended for improving the preprocessing and update time for dynamic data structures (Beimel et al., 2022; Song et al., 2023b; Cherapanamjeri et al., 2023): the idea is to set up the database as the random strings used by oblivious data structures. Suppose we have $\mathcal{A}_1, \ldots, \mathcal{A}_k$ oblivious data structures, each with their associated random strings $r_1, \ldots, r_k \in \{0,1\}^*$. The database is R, where the i-th row is the random string r_i. Notions such as sensitivity are in turn defined with respect to these random strings. We will now illustrate how to tie these Monte Carlo search data structures to the selection problem. We will use (c, r)-ANN as an example, but our reduction does not exploit any additional structure on (c, r)-ANN, so it can be well-extended to more general search data structures.

Given a dataset $U \in \mathbb{R}^d$ and a possibly adaptive query $v \in \mathbb{R}^d$, recall that a (c, r)-ANN data structure would have the following behavior:

- If there exists some $u \in U$ with $\|u - v\| \leq r$, then it outputs some point $u' \in U$ with $\|u' - v\| \leq cr$. This succeeds with probability at least $\frac{9}{10}$ (note we do not need probability boosting for our base data structure, at this stage). We modify the data structure to output all such points it has found.
- If there is no such point, report NULL.

Note that ANN data structures can only produce false negatives but no false positives. We can associate the output of the data structure with a binary vector of length n as follows: the i-th entry of the vector corresponds to the decision the data structure has made on the point u_i, i.e., if it outputs u_i, then we set the i-th entry to be 1, and 0 otherwise. If the data structure outputs nothing, we set the vector to be $\mathbf{0}_n$. Fixing dataset U and query v, these binary vectors are completely determined by $r_1, \ldots, r_k$. Denote these binary vectors as $b^1, \ldots, b^k$. We then execute the REPORTONESIDEDNOISYARGMAX mechanism on these vectors, and output the corresponding index. We now describe the details of our algorithm.

We will use T to denote the number of queries, and k to denote the total number of data structures. We assume each data structure succeeds with probability at least $1 - \delta$. We let $\varepsilon_{\mathrm{DP}}$ denote the DP parameter for REPORTONESIDEDNOISYARGMAX. We will use δ_{fail} to denote the overall failure probability of our algorithm, and $\beta = \delta_{\mathrm{fail}}/T$ to denote the failure probability of a single query step. The algorithm is as follows.

Initialization.

- Prepare k independent copies of oblivious data structures, denoted by $\mathcal{A}_1, \ldots, \mathcal{A}_k$, over the dataset U.

Query.

- Receive query vector $v \in \mathbb{R}^d$.
- Sample $l = \widetilde{O}(s)$ indices independent and uniformly from $[k]$ with replacement. Denote the corresponding data structures by $\mathcal{A}_{(1)}, \ldots, \mathcal{A}_{(l)}$.
- Feed v into $\mathcal{A}_{(1)}, \ldots, \mathcal{A}_{(l)}$, receive binary vectors $b^{(1)}, \ldots, b^{(l)}$.
- Compute REPORTONESIDEDNOISYARGMAX$(b^{(1)}, \ldots, b^{(l)})$ with parameter $\varepsilon_{\mathrm{DP}}$. Let i be the corresponding index.

- Output u_i if $\|u_i - v\|_2 \leq cr$, otherwise output NULL.

Update.

- Receive update vector $u \in \mathbb{R}^d$.
- Update $\mathcal{A}_1, \ldots, \mathcal{A}_k$ with u.

We will specify the parameters to show that the algorithm is indeed differentially private with respect to the internal randomness of the data structures.

Setting Parameters. We start by setting the parameter $\varepsilon_{\mathrm{DP}}$ which affects the privacy guarantee of REPORTONESIDED-NOISYARGMAX. We will set $\varepsilon_{\mathrm{DP}} = 1/2$ (in fact, for our analysis, $\varepsilon_{\mathrm{DP}}$ can be any constant smaller than 1). Set the number of samples c to be

$$l = O(s \log^2(n/\beta))$$

and the total number of data structures k to be

$$\begin{aligned} k &= 200 \cdot 6l \cdot \varepsilon_{\mathrm{DP}} \cdot \sqrt{2T \ln(100/\beta)} \\ &= \widetilde{O}(\sqrt{T} \cdot s). \end{aligned}$$

Privacy Guarantees. Our analysis differs from standard DP analysis, in which one could reason over the privacy and utility simultaneously. We will start by showing that the algorithm is indeed differentially private, then use this fact to argue the output of the algorithm provides good utility.

To understand the privacy of the algorithm, we adapt the framework from (Beimel et al., 2022): let $r_1, \ldots, r_k \in \{0,1\}^*$ denote the random strings used by data structures $\mathcal{A}_1, \ldots, \mathcal{A}_k$ and let $R = \{r_1, \ldots, r_k\}$ be the database whose i-th row is r_i. We will prove the transcript of the interaction between the adaptive adversary and algorithm is differentially private with respect to R. Fix a time step t. We let $\mathbf{out}_t(R)$ denote the index output by the algorithm at time t. Even fixing R, the output $\mathbf{out}_t(R)$ is still a random variable as it depends on i.i.d. exponential noise that is oblivious to the algorithm. Similar to (Beimel et al., 2022), we define the transcript at time t as $\mathcal{T}_t(R) = (v_t, \mathbf{out}_t(R))$. Define the overall transcript as

$$\mathcal{T}(R) = \mathcal{T}_1(R), \ldots, \mathcal{T}_T(R).$$

We view $\mathcal{T}_t$ and $\mathcal{T}$ as algorithms that given a database R, output the transcripts. This enables us to reason about differential privacy with respect to R.

Lemma B.7. *For any time step t, $\mathcal{T}_t$ is $(\frac{6l}{k} \cdot \varepsilon_{\mathrm{DP}}, 0)$-DP with respect to R.*

Proof. Note that $\mathcal{T}_t(R) = (v_t, \mathbf{out}_t(R))$. For any fixed single step t, v_t does not provide extra information about R, so we only need to consider $\mathbf{out}_t(R)$.

We let $\widetilde{\mathbf{out}}_t(R) :=$ REPORTONESIDEDNOISYARGMAX$(b^{(1)}, \ldots, b^{(l)})$ with parameter $\varepsilon_{\mathrm{DP}}$. As we have argued in the preceding discussion, the binary vectors are determined by $\{r_{(1)}, \ldots, r_{(l)}\} \subset R$ which are subsampled from R. As $\widetilde{\mathbf{out}}_t(R)$ is $(\varepsilon_{\mathrm{DP}}, 0)$-DP with respect to the subset (Theorem B.6), by Theorem B.3, it is $(\frac{6l}{k} \cdot \varepsilon_{\mathrm{DP}}, 0)$-DP with respect to R. Finally, observe that $\mathbf{out}_t(R)$ is a post-processing of $\widetilde{\mathbf{out}}_t(R)$ as it's a deterministic mapping that only depends on the output of $\widetilde{\mathbf{out}}_t(R)$. Hence, we conclude that $\mathbf{out}_t(R)$ is $(\frac{6l}{k} \cdot \varepsilon_{\mathrm{DP}}, 0)$-DP with respect to R, as desired. □

Given that a single step is DP with respect to R, we can then perform advanced composition to prove the privacy of $\mathcal{T}$.

Lemma B.8. *$\mathcal{T}$ is $(\frac{1}{100}, \frac{\beta}{100})$-DP with respect to R.*

Proof. By Lemma B.7, we know that for any fixed $t \in [T]$, $\mathcal{T}_t(R)$ is $(\frac{6l}{k} \cdot \varepsilon_{\mathrm{DP}}, 0)$-DP with respect to R. Further, observe that $\mathcal{T}$ is an adaptive composition of $\mathcal{T}_T \circ \ldots \circ \mathcal{T}_1$, so we can apply Theorem B.2 with $\varepsilon = \frac{6l}{k} \cdot \varepsilon_{\mathrm{DP}}$ and $\delta = 0, \delta_0 = \beta/100$. This yields a mechanism that is $(\varepsilon_0, \delta_0)$-DP with respect to R, where

$$\varepsilon_0 = \sqrt{2T \ln(100/\beta)} \cdot \varepsilon + 2T\varepsilon^2$$

$$\leq \frac{1}{200} + \frac{1}{200}$$
$$= \frac{1}{100}.$$

Thus, we conclude $\mathcal{T}$ is $(\frac{1}{100}, \frac{\beta}{100})$-DP with respect to R. □

Accuracy: Differentiating Signals from Noise. Given the privacy guarantees of $\mathcal{T}$, we are ready to prove accuracy against an adaptive adversary. Similar to (Beimel et al., 2022), define $\overline{v}_t = (v_1, v_2, \ldots, v_t)$ to be the query sequence up to time t and consider feeding a random string r (for initialization) and $\overline{v}_t$ to an oblivious data structure $\mathcal{A}$. Let $\mathcal{A}(r, \overline{v}_t)$ denote the output. Finally, define $\mathrm{acc}_{\overline{v}_t}(r)$ to be the indicator of whether $\mathcal{A}(r, \overline{v}_t)$ succeeds, i.e., if v_t is a "yes" instance, then $\mathcal{A}(r, \overline{v}_t)$ indeed outputs a subset of points in U that are at most cr away from v_t. We will prove that with high probability, a large constant fraction of *all* data structures succeed.

Lemma B.9. *For any fixed time step $t \in [T]$, we have $\sum_{j=1}^{k} \mathrm{acc}_{\overline{v}_t}(r_j) \geq \frac{4}{5}k$ with probability at least $1 - \beta$.*

Proof. The proof will rely on the generalization property of DP. First note that $\mathrm{acc}_{\overline{v}_t}$ is determined by the transcript $\mathcal{T}$, as $\overline{v}_t$ is a substring of $\mathcal{T}$ and $\overline{v}_t$ determines $\mathrm{acc}_{\overline{v}_t}$. To invoke Theorem B.4, we first note that each row of R is drawn uniformly, and due to Lemma B.8, we know that $\mathcal{T}$ is $(\frac{1}{100}, \frac{\beta}{100})$-DP with respect to R. Thus, we have that the empirical average of $\mathrm{acc}_{\overline{v}_t}$ over R which is $\frac{1}{k}\sum_{j=1}^{k} \mathrm{acc}_{\overline{v}_t}(r_j)$ and the expectation of $\mathrm{acc}_{\overline{v}_t}$ over the uniform distribution $\mathbb{E}[\mathrm{acc}_{\overline{v}_t}(r)]$ are close. Setting $\varepsilon = 1/100$, $\delta = \beta/100$ and $k \gg \frac{1}{\varepsilon^2}\log(2\varepsilon/\delta)$, we conclude

$$\Pr\left[\left|\frac{1}{k}\sum_{j=1}^{k} \mathrm{acc}_{\overline{v}_t}(r_j) - \mathbb{E}[\mathrm{acc}_{\overline{v}_t}(r)]\right| \geq \frac{1}{10}\right] \leq \beta.$$

It remains to get a good handle on the expectation. If we let $\mathcal{U}$ denote the uniform distribution we sample r from, then we can write the expectation $\mathbb{E}_{r\sim\mathcal{U}}[\mathrm{acc}_{\overline{v}_t}(r)] = \Pr_{r\sim\mathcal{U}}[\mathrm{acc}_{\overline{v}_t}(r) = 1]$ as $\mathrm{acc}_{\overline{v}_t}(r)$ is an indicator. The crucial point here is that the probability is taken over the randomness of r when the sequence of queries $\overline{v}_t$ is *fixed*. Thus, we can utilize the property of our oblivious data structure, i.e., $\Pr_{r\sim\mathcal{U}}[\mathrm{acc}_{\overline{v}_t}(r) = 1] \geq \frac{9}{10}$. To put it together, we have

$$\frac{1}{k}\sum_{j=1}^{k} \mathrm{acc}_{\overline{v}_t}(r_j) \geq \frac{9}{10} - \frac{1}{10}$$
$$= \frac{4}{5}.$$

Thus, we complete the proof. □

To improve the efficiency of querying, recall that we resort to sampling. We prove that sampling does not hurt the accuracy.

Lemma B.10. *For all $t \in [T]$, let l be the sampled indices of time t, we have $\sum_{j=1}^{l} \mathrm{acc}_{\overline{v}_t}(r_{(j)}) \geq \frac{3}{4}l$ with probability at least $1 - \delta_{\mathrm{fail}}$.*

Proof. Fix any $t \in [T]$, we condition on the event that Lemma B.9. Therefore $\sum_{j=1}^{k} \mathrm{acc}_{\overline{v}_t}(r_j) \geq \frac{4}{5}k$. This means that any i.i.d. sample (with replacement) from these k indices succeeds with probability at least $4/5$. Consequently $\mathbb{E}[\sum_{j=1}^{l} \mathrm{acc}_{\overline{v}_t}(r_{(j)})] \geq \frac{4}{5}l$. By Hoeffding's bound, we have

$$\Pr\left[\left|\sum_{j=1}^{l} \mathrm{acc}_{\overline{v}_t}(r_{(j)}) - \mathbb{E}[\sum_{j=1}^{l} \mathrm{acc}_{\overline{v}_t}(r_{(j)})]\right| \geq \alpha\right] \leq 2\exp(-\frac{2\alpha^2}{l}),$$

and setting $\alpha = \frac{1}{5}l$, we conclude that $\sum_{j=1}^{l} \mathrm{acc}_{\overline{v}_t}(r_{(j)}) \geq \frac{3}{5}l$ with probability at least $1 - \exp(-\Theta(l))$. □

We present an abstract way to reason over the data structure task. Fix the preprocessed dataset $U \in \mathbb{R}^d$. Consider any (possibly adaptive) query $v \in \mathbb{R}^d$, and let $b \in \{0,1\}^n$ denote the characteristic vector of v with respect to U, in terms of (c,r)-ANN: if there exists some $u_i \in U$ with $\|u_i - v\| \leq r$, then: for any $j \in [n]$ with $\|u_j - v\| \leq cr$, we set $b_j = 1$. Note that if no point in U is r-close to v, then b is either the all-0s vector or it has some nonzero entries that are cr-close to v. Similarly, for each data structure $\mathcal{A}_1, \ldots, \mathcal{A}_k$ and their corresponding random strings $r_1, \ldots, r_k$, we use $b^{(i)}$ to denote the characteristic vector *under the random string* r_i, i.e., $b^{(i)}_j = 1$ if and only if the LSH $\mathcal{A}_i$ discovers that $\|u_j - v\| \leq cr$. Note that it is completely possible that the vector b has large support size, but some $b^{(i)} = \mathbf{0}_n$.

Now, imagine we have an $O(n)$ time budget. Then our algorithm is essentially identical to reporting the noisy max: we first sum over all characteristic vectors: $b^{\text{sum}} = \sum_{i=1}^k b^{(i)}$. Note that the i-th entry of b^{sum} counts the total number of successful data structures that report point u_i. Then, we sample n independent Laplacian noise variables of scale $1/\varepsilon_{\text{DP}}$, and then add these noise variables to each entry of b^{sum}. Then we simply report the maximum index of the noisy counts with a simple post-processing by directly computing the distance between the corresponding point and the query. The algorithm is naturally $(\varepsilon_{\text{DP}}, 0)$-DP with respect to the random strings $r_1, \ldots, r_k$!

To prove the utility of the algorithm, i.e., that we can actually differentiate the signal from the noise, we require the following tail bound for exponential random variables:

Lemma B.11. *Let $Y \sim \text{Exp}(b)$, then*

$$\Pr[Y \geq t \cdot b] \leq \exp(-t)$$

Let us choose $t = C \log n$ for some absolute constant $C > 0$. This means that even if we choose ε_{DP} to be $1/2$, the magnitude of the noise will be roughly $\Theta(\log n)$. Consider a simple case, where the query v only has one point $u \in U$ with $\|u - v\|_2 \leq r$ and $\|u - v\|_2 \leq cr$. Then, for a successful data structure $\mathcal{A}_i$, it will have exactly one non-zero entry. Let us assume that we sample $l = O(\log^2 n)$ independent data structures to perform the mechanism, using $\varepsilon_{\text{DP}} = 1/2$. Then by amplification via subsampling, the algorithm is $\frac{6l}{k} \cdot \varepsilon_{\text{DP}}$-DP. Performing an advanced composition over all T queries, we conclude the algorithm is $(\frac{1}{100}, \frac{1}{100 \cdot \text{poly}(n)})$-DP. We can then use the generalization property of DP to conclude that, with probability at least $1 - 1/\text{poly}(n)$, at least a $3/4$-fraction of the sampled data structures succeed.

Conditioning on these events, we argue that the noisy max is indeed the correct answer: we know that a large constant fraction of data structures succeed, therefore the corresponding point in b^{sum} has count $\Omega(\log^2 n)$. On the other hand, the exponential noise itself has magnitude at most $O(\log n)$. This means that adding the exponential noise, we can still differentiate the point. We formalize this argument with the following lemma.

Lemma B.12. *For all $t \in [T]$, the algorithm outputs a point u_t that is a c-ANN for the adaptive query v_t if v_t satisfies Assumption 1.2, with probability at least $1 - \delta_{\text{fail}} - 1/\text{poly}(n)$.*

Proof. By Lemma B.10, among the l samples, at least a $\frac{3}{4}$-fraction of the data structures succeed, i.e., if there exists an r-near neighbor of v_t, then at least $\frac{3}{4}l$ data structures have their characteristic vectors with at least 1 non-zero entry. By Assumption 1.2, the support size of each characteristic vector can be at most s; since we set $l = O(s \log^2(n/\beta))$, by the pigeonhole principle, there exists at least one entry in b^{sum} with magnitude $\Omega(\log^2(n/\beta))$. Now, conditioning on the event that all n Laplacian noise variables have magnitude $O(\log n)$; this holds with probability $1 - 1/\text{poly}(n)$. Thus, we know that there exists at least one entry of b^{sum} which has magnitude $\Omega(\log^2 n)$ and we can differentiate this entry from all the noise added.

Now, consider the case that there is no point $u \in U$ with $\|u - v_t\|_2 \leq r$. Then the data structure is allowed to either return points that are cr-near neighbors of v_t, or return nothing. Since the base data structure either outputs a false negative (a cr-near neighbor) or outputs nothing, we analyze two cases:

- Case 1: $\|b^{\text{sum}}\|_\infty \geq C \log^2 n$ for some fixed constant C. In this case, we know that the max index before adding the noise must correspond to a cr-near neighbor. Hence, adding the noise and outputting the noisy max will not hide this large entry, and so we can return it and satisfy the specification of an c-ANN data structure.
- Case 2: $\|b^{\text{sum}}\|_\infty < C \log^2 n$. In this case, it must be the case there is no r-near neighbor. Regarding the REPORTONESIDEDNOISYARGMAX mechanism, since the magnitude of the signal in this case is $O(\log n)$ and with high probability, the exponential noise has magnitude $O(\log n)$, we have that REPORTONESIDEDNOISYARGMAX

outputs a noisy index (meaning that the corresponding index is not a cr-near neighbor). In the post-processing phase, we ensure the algorithm outputs only a cr-near neighbor by checking the distance.

This concludes the proof of correctness.

To compute the failure probability, note that a constant fraction of the data structures succeed with probability at least $1 - \delta_{\text{fail}}$, and all exponential random variables have magnitude $\Theta(\log n)$ with probability at least $1 - 1/\operatorname{poly}(n)$. A union bound concludes the bound on the failure probability. □

Of course, this algorithm is by no means efficient — it takes $O(n)$ time to generate all n noise variables per query, making the sublinear query time linear. In the next section, we will show it is sufficient to generate s exponential variables for the support of b^{sum}, and take the noisy max over these entries.

B.3. Sparse Noise and Order Statistics: A Fast Algorithm

We make the following observation for the REPORTONESIDEDNOISYARGMAX algorithm: if $b^{\text{sum}} = \mathbf{0}_n$, then the entry with the max noise must be the one contains $X_{(n)}$, the largest order statistics of the exponential distribution. If b^{sum} has s nonzero entries, then there are two cases: if $X_{(n)}$ is among the nonzero entries, since those entries are all positive, the max entry must be within the s nonzero entries; if $X_{(n)}$ is not among the nonzero entries, then the max entry is among the entry contains $X_{(n)}$, and the nonzero entry with the largest magnitude after adding noises. Note that the first event happens with probability $\frac{s}{n}$, and for this case, we generate $s-1$ i.i.d. noises until they do not exceed $X_{(n)}$ so that they obey the order statistics distribution. For the second case, we generate s i.i.d. noises until they do not exceed $X_{(n)}$, randomly assign a zero entry to $X_{(n)}$, and output the max between $X_{(n)}$ and the largest nonzero noisy entry. Note that this algorithm is extremely efficient — according to Fact A.3, $X_{(n)}$ can be generated in $O(1)$ time, and the above procedure runs in $O(s \log n)$ time with high probability, proportional to the support size.

Fact B.13. *Let Z be a geometric random variable with success probability $p \in (0,1)$, then for any positive integer k, we have*

$$\Pr[Y > k] \leq \exp(-k \log(1/(1-p)))$$

Proof. Note that by definition,

$$\begin{aligned} \Pr[Y > k] &= (1-p)^k \\ &= \exp(k \log(1-p)) \\ &= \exp(-k \log(1/(1-p))). \end{aligned}$$

□

Lemma B.14. *Let $\mathcal{A}_1, \mathcal{A}_2$ be two algorithms, with the following behavior. Let $s > 0$:*

- *$\mathcal{A}_1$, at time t, it receives an s-sparse vector $u \in \mathbb{N}^n$ and let $U_t \subseteq [n]$ denote the support of u. We add a dense exponential noise vector with parameter 2, denoted by $\eta \in \mathbb{R}^n$ and add this to $u \in \mathbb{R}^n$, and output index i_* such that $i_* = \arg\max_{j \in [n]} (u + \eta)_j$;*
- *$\mathcal{A}_2$, at time t, it receives an s-sparse vector $u \in \mathbb{N}^n$ and let $U_t \subseteq [n]$ denote the support of u. We flip a biased coin with probability of being head $\frac{s}{n}$. If head, we generate a max noise X from the distribution $X_{(n)}$ of parameter 2, and then repeatedly generating $s-1$ i.i.d. exponential noises of parameter 2 until none of them exceed X, and let $\eta_1, \ldots, \eta_{s-1}$ denote these noises, form vector $\eta \in \mathbb{R}^n$ with the support being U_t, and the values being randomly assigned from $\eta_1, \ldots, \eta_{s-1}, X$. Then output $i_* = \arg\max_{j \in [n]} (\eta + \mu)_j$. If the coin flips tail, then generate X from $X_{(n)}$, and generate s i.i.d. exponential noises until none of them exceed X. Randomly assign X to one of the non-support entry denoted by i_1, and let η denote the noises for the support, output i_1 if $X > \max_{j \in [n]} (\mu + \eta)_j$ and output $\arg\max_{j \in [n]} (\mu + \eta)_j$ otherwise.*

Then, $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same output distribution. Moreover, $\mathcal{A}_2$ runs in $O(s \log n)$ time.

Proof. We will prove that $\mathcal{A}_2$ has exactly the same output distribution as $\mathcal{A}_1$. We do so by analyzing an hypothetical algorithm $\mathcal{A}_2'$ that generates X from $X_{(n)}$ first and assign it to a uniformly chosen entry i_*, then generate the noises for all

other entries conditioning they are smaller than X. We will prove the distribution of the noise generated by $\mathcal{A}_2'$ is identical to $\mathcal{A}_1$, then it's easy to see that the output distribution of $\mathcal{A}_2$ is the same as $\mathcal{A}_1$.

Suppose the noise η_i is generated from distribution with PDF f and CDF F, then the CDF of $X_{(n)}$ is $G(x) = F(x)^n$, and PDF $g(x) = nF(x)^{n-1}f(x)$. Let the joint CDF of noise generated by $\mathcal{A}_1$ be $F_1(x_1, \ldots, x_n) = F(x_1) \cdot F(x_2) \cdot \ldots \cdot F(x_n)$, and the PDF noise generated by $\mathcal{A}_2'$ be $F_2(x_1, \ldots, x_n)$. Let the sorted array of $x_1, \ldots, x_n$ be $x_1^*, \ldots, x_n^*$, and $x_0^* = -\infty$ we have :

$$\begin{aligned}
F_2(x_1, \ldots, x_n) &= \frac{1}{n} \sum_{i=1}^n \int_0^{x_i} g(x) \prod_{x_j < x} \frac{F(x_j)}{F(x)} \, \mathrm{d}x \\
&= \sum_{i=1}^n \sum_{j=1}^i \int_{x_{j-1}^*}^{x_j^*} f(x) F(x)^{n-1} \prod_{k=1}^{j-1} \frac{F(x_k^*)}{F(x)} \, \mathrm{d}x \\
&= \sum_{j=1}^n (n-j+1) \int_{x_{j-1}^*}^{x_j^*} f(x) F(x)^{n-j} \prod_{k=1}^{j-1} F(x_k^*) \, \mathrm{d}x \\
&= \sum_{j=1}^n \Big(\prod_{k=1}^{j-1} F(x_k^*) \Big) \cdot \Big(F(x_j^*)^{n-j+1} - F(x_{j-1}^*)^{n-j+1} \Big) \\
&= F(x_1^*) \cdot \cdots \cdot F(x_n^*) \\
&= F_1(x_1, \ldots, x_n)
\end{aligned}$$

Hence, the noise distributions of these two algorithms are identical.

To see $\mathcal{A}_2$ and $\mathcal{A}_2'$ have the same noise distribution, consider two cases. If the max noise X is in the support (happens with probability $\frac{s}{n}$), then we know that the max index must be within the support. For the noise distribution, it's easy to see that as long as all η_i's are smaller than X, then the noise distributions are identical. For the tail case, the reasoning is similar, except that our maximum must be over X and the noisy entries in the support.

Regarding the running time, we only need to examine the probability that the noises $\eta_1, \ldots, \eta_{s-1}$ or $\eta_1, \ldots, \eta_s$ do not exceed X. We without loss of generality prove for the case where we generate s independent exponential noises. Let λ be the parameter, note that this is equivalent to the probability that $Y_{(s)} \leq X_{(n)}$, where $Y_{(s)}$ is the max order statistics for another independent sequence of exponential noises. The CDF of $Y_{(s)}$ is $F_{Y_{(s)}}(y) = (1 - e^{-\lambda y})^s$, and the PDF of $X_{(n)}$ is $f_{X_{(n)}}(x) = n\lambda e^{-\lambda x}(1 - e^{-\lambda x})^{n-1}$, then we have

$$\begin{aligned}
\Pr[Y_{(s)} \leq X_{(n)}] &= \int_0^\infty \Pr[Y_{(s)} \leq x] f_{X_{(n)}}(x) \, \mathrm{d}x \\
&= \int_0^\infty (1 - e^{-\lambda x})^s n\lambda e^{-\lambda x} (1 - e^{-\lambda x})^{n-1} \, \mathrm{d}x \\
&= \int_0^\infty n\lambda e^{-\lambda x} (1 - e^{-\lambda x})^{n+s-1} \, \mathrm{d}x
\end{aligned}$$

we do a change of variable $u = 1 - e^{-\lambda x}$, so $\mathrm{d}u = \lambda e^{-\lambda x} \mathrm{d}x$, and for $x = 0$, $u = 0$ and for $x = \infty$, $u = 1$, therefore

$$\begin{aligned}
\int_0^\infty n\lambda e^{-\lambda x} (1 - e^{-\lambda x})^{n+s-1} \, \mathrm{d}x &= n \int_0^1 u^{s+n-1} \, \mathrm{d}u \\
&= \frac{n}{n+s},
\end{aligned}$$

similarly, for $s - 1$ noises, the probability is $\frac{n}{n+s-1}$. In both cases, we have that the success probability is at least $\frac{1}{2}$, and note that this is a geometric random variable Z, by Fact B.13, we have that

$$\Pr[Z > k] \leq \exp(-k),$$

set $k = c \log n$ for some large enough constant c so that k is an integer, we have this happens with probability at most $1/\operatorname{poly}(n)$, hence with high probability, we only need to generate the noises for $k = O(\log n)$ times. Moreover, X can be efficiently generated via inverse CDF sampling, so the overall runtime for generating these noises is $O(s \log n)$, as desired. □

C. Speeding Up Updates via Batching

In this section, we focus on developing a fast update procedure for decremental ℓ_2 LSH against an adaptive adversary. This is crucial for applications such as online weighted matching (see Section 4.1).

C.1. Adaptive Low-Dimensional ℓ_2 LSH

We start by reviewing the algorithm for low-dimensional ℓ_2 LSH, against an *oblivious adversary*. Throughout, we let $\varepsilon, \delta \in (0, 1)$ denote the precision and failure probability.

Initialization.

- Prepare one Johnson-Lindenstrauss (Johnson & Lindenstrauss, 1984) transform $S : \mathbb{R}^d \to \mathbb{R}^m$ where $m = O(\varepsilon^{-2} \log(n/\delta))$.
- Compute SU.
- Initialize an (c, r)-ℓ_2 LSH data structure on SU.

Query.

- Given query point $v \in \mathbb{R}^d$, first compute Sv.
- Query the LSH with Sv, output the corresponding point.

Deletion.

- Given a point $u \in U$ to-be-deleted, first compute Su.
- Locate Su in the hash buckets of the LSH, remove Su.

The deletion procedure first computes the embedded point in time $kd = O(\varepsilon^{-2} d \log(n/\delta))$, then locates the hash bucket in $O(n^\rho)$ time. In order to make such data structure adaptive, we need to create $\sqrt{T} \cdot s$ independent copies, and naïvely update all $\sqrt{T} \cdot s$ data structures takes time (up to polylogarithmic factors and assume $\varepsilon = O(1)$):

$$\sqrt{T} \cdot s \cdot (d + n^\rho)$$

While the term $\sqrt{T} \cdot s \cdot n^\rho$ seems to be unavoidable as we will have to update all LSH data structures, the first step of applying Johnson-Lindenstrauss could be further acclerated via batching and fast rectangular matrix multiplication.

C.2. Faster Update via Batching

Recall that we define $\theta(a, b)$ for $a, b > 0$ to be the value such that $\mathcal{T}_{\mathrm{mat}}(a, b, \theta(a, b)) = (ab)^{1+o(1)}$. Note that θ is monotone in its argument, i.e., fixing one argument, θ grows or decreases with the other argument. Our idea is the following: instead of eagerly updating all data structures whenever a point is deleted, we can delay the deletion until the data structure is queried, which is when the update must be performed. We will partition the length-T query sequence into blocks of size $\theta(\sqrt{T} \cdot s, d)$ and for simplicity, we will denote it as θ in the following.

Initialization.

- Prepare $k = \sqrt{T} \cdot s$ JL transforms (Johnson & Lindenstrauss, 1984) $S_1, \ldots, S_k : \mathbb{R}^d \to \mathbb{R}^m$ where $m = O(\varepsilon^{-2} \log(n/\delta))$.
- Set $\mathcal{S} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_k \end{bmatrix} \in \mathbb{R}^{mk \times d}$.
- Compute $\mathcal{S}U$.
- Initialize k (c, r)-ℓ_2 LSH's on $S_1 U, S_2 U, \ldots, S_k U$.
- Associate a deletion list to each data structure. The initial deletion lists are empty. Also initialize a global deletion list that keeps track of all points deleted so far.

Query.

- Receive query point $v_t \in \mathbb{R}^d$.
- Sample $l = O(s \log^2 n)$ data structures, denote the sampled JLs as $S_{(1)}, \ldots, S_{(l)}$.
- Batch compute $\begin{bmatrix} S_{(1)} \\ \vdots \\ S_{(l)} \end{bmatrix} v_t \in \mathbb{R}^{lm}$.
- Update the sampled data structures that are not up-to-date, update their corresponding deletion lists.
- Query corresponding LSH's with $S_{(1)} v_t, \ldots, S_{(l)} v_t$.
- Compute the characteristic vectors of these LSH's outputs, apply sparse REPORTONESIDEDNOISYARGMAX to the count vector and compute the order statistics.
- Output the noisy max index.

Deletion.

- If the algorithm is at the end of a block:
  - Update all k data structures with θ points in the block. In particular, let $u_{(1)}, \ldots, u_{(\theta)}$ denote the points to-be-deleted.
  - Compute $\mathcal{S} \begin{bmatrix} | & \ldots & | \\ u_{(1)} & \ldots & u_{(\theta)} \\ | & \ldots & | \end{bmatrix}$.
  - Delete $S_i u_{(1)}, \ldots, S_i u_{(\theta)}$ from the i-th LSH.
  - Update the deletion list of all k data structures and the global deletion list.
- Otherwise, update the global deletion list.

We first note that the correctness of the data structure follows directly from the design: for each query, we will update the data structures first. So we focus on analyzing the runtime.

Theorem C.1. *Let $U \subseteq \mathbb{R}^d$ satisfy Assumption 1.3 and $\{v_1, \ldots, v_T\} \subseteq \mathbb{R}^d$ be a sequence of adaptive queries. There exists a randomized data structure with the following guarantees:*

- *It preprocesses U in time $\widetilde{O}(\mathcal{T}_{\mathrm{mat}}(\sqrt{T} \cdot s, d, n) + \sqrt{T} \cdot s \cdot n^{1+\rho})$;*

- *It uses space* $\widetilde{O}(\sqrt{T} \cdot s \cdot n^{1+\rho} + nd)$*;*
- *Given a query* v_t *for* $t \in [T]$*, it returns a point* $u \in U$ *with* $\|v_t - u\|_2 \leq cr$ *if* $B(v, r) \cap U \neq \emptyset$ *and NULL otherwise, with probability at least* $1 - 1/\operatorname{poly}(n)$*. This procedure takes time* $\widetilde{O}(s \cdot (d + n^\rho))$*;*
- *If* u_t *is the output of* v_t*, it deletes* u_t *in amortized time* $\widetilde{O}((\sqrt{T} \cdot s \cdot d)^{1+o(1)}/\theta(\sqrt{T} \cdot s, d) + \sqrt{T} \cdot s \cdot n^\rho)$*.*

Proof. We note that preprocessing time, space and query time, together with their guarantees are straightforward, so we will focus on bounding the deletion time. Our analysis will be over a block of size θ, and the final update time will be amortized over θ steps. There are two cases to consider.

Case 1. Update during the query. Each time we receive a query, we need to sample $l = O(s \log^2 n)$ data structures and update these data structures accordingly. Suppose we are at the i-th step of the block. Then in the worst case, all of these data structures need to be updated for all prior $i - 1$ points deleted. We can compute the update time over the block as follows:

$$\sum_{i=1}^{\theta} \underbrace{\mathcal{T}_{\mathrm{mat}}(l, d, i)}_{\text{time to apply JL}} + \underbrace{iln^\rho}_{\text{time to update LSH}} \leq \sum_{i=1}^{\theta} \widetilde{O}(is \cdot (d + n^\rho)) = \widetilde{O}(\theta^2 s \cdot (d + n^\rho)).$$

Thus, the amortized cost per update-during-the-query is $\widetilde{O}(\theta s \cdot (d + n^\rho))$.

Case 2. Update at the end of the block. In this case, we need to update all $k = \widetilde{O}(\sqrt{T} \cdot s)$ data structures with θ points. Applying JL takes time

$$\mathcal{T}_{\mathrm{mat}}(\sqrt{T} \cdot s, d, \theta(\sqrt{T} \cdot s, d)) = \widetilde{O}((\sqrt{T} \cdot s \cdot d)^{1+o(1)})$$

by the definition of θ, and updating k LSH's takes time $\widetilde{O}(\theta \cdot \sqrt{T} \cdot s \cdot n^\rho)$. Thus, the amortized cost per step is

$$\widetilde{O}((\sqrt{T} \cdot s \cdot d)^{1+o(1)}/\theta + \sqrt{T} \cdot s \cdot n^\rho).$$

As the second cost dominates, we obtain the desired update time. □

Remark C.2. *As our data structure gains an advantage over using* d *copies via a net argument when* $\sqrt{T} \cdot s \leq d$*, we know that* $\theta(\sqrt{T} \cdot s, d) \geq \theta(\sqrt{T} \cdot s, \sqrt{T} \cdot s) = (\sqrt{T} \cdot s)^\alpha$ *as* θ *is monotone, and* $\alpha \approx 0.32$ *is the dual matrix multiplication exponent (Williams et al., 2024; Le Gall, 2024) where* $\mathcal{T}_{\mathrm{mat}}(n, n, n^\alpha) = n^{2+o(1)}$*. This implies an amortized cost per update of*

$$(\sqrt{T} \cdot s)^{1-\alpha+o(1)} \cdot d^{1+o(1)} + \sqrt{T} \cdot s \cdot n^\rho \approx (\sqrt{T} \cdot s)^{0.68+o(1)} \cdot d^{1+o(1)} + \sqrt{T} \cdot s \cdot n^\rho \tag{1}$$

Compared to the naïve update in which the first term is $\sqrt{T} \cdot s \cdot d$*, this improvement is significant for relatively large* $\sqrt{T} \cdot s$*.*

C.3. Generalization to ℓ_p LSH

We note that the above algorithm does not exploit any particular structure of ℓ_2; we use the Johnson-Lindenstrauss transform (Johnson & Lindenstrauss, 1984) in its most general formulation, as the speedup we obtain comes from batch matrix multiplication. Thus, we could generalize the algorithm to any ℓ_p norm for $p \in (0, 2]$, using p-stable sketches (Indyk, 2006; Datar et al., 2004).

Corollary C.3. *Let* $U \subseteq \mathbb{R}^d$ *satisfying Assumption 1.3 and* $\{v_1, \ldots, v_T\} \subseteq \mathbb{R}^d$ *be a sequence of adaptive queries. Let* $p \in (0, 2]$*. There exists a randomized data structure with the following guarantees:*

- *It preprocesses* U *in time* $\widetilde{O}(\mathcal{T}_{\mathrm{mat}}(\sqrt{T} \cdot s, d, n) + \sqrt{T} \cdot s \cdot n^{1+\rho})$*;*
- *It uses space* $\widetilde{O}(\sqrt{T} \cdot s \cdot n^{1+\rho} + nd)$*;*
- *Given a query* v_t *for* $t \in [T]$*, it returns a point* $u \in U$ *with* $\|v_t - u\|_p \leq cr$ *if* $B(v, r) \cap U \neq \emptyset$ *and NULL otherwise, with probability at least* $1 - 1/\operatorname{poly}(n)$*. This procedure takes time* $\widetilde{O}(s \cdot (d + n^\rho))$*;*
- *If* u_t *is the output of* v_t*, it deletes* u_t *in amortized time* $\widetilde{O}((\sqrt{T} \cdot s \cdot d)^{1+o(1)}/\theta(\sqrt{T} \cdot s, d) + \sqrt{T} \cdot s \cdot n^\rho)$*.*

D. Adaptive Regression with Private Median and ℓ_∞ Guarantee

In this section, we provide the necessary background for adaptive regression problem under turnstile updates, with a variety of algorithms with diverse guarantees.

D.1. Preliminaries on Adpative Regression and Sketching

The adaptive regression we will be studying is defined in Assumption 1.4, we will state here again for completeness.

Let $U \in \mathbb{R}^{n\times d}$ be a design matrix and $b \in \mathbb{R}^n$ be a response vector with $n \gg d$, the goal is to solve the ℓ_2 regression problem:

$$x^* := \arg\min_{x\in\mathbb{R}^d} \|Ux - b\|_2^2,$$

with the optimal solution given by the normal equation:

$$x^* = (U^\top U)^\dagger U^\top b$$

with $M^\dagger$ being the Moore–Penrose inverse of the matrix M. As solving the normal equation exactly is time- and space-consuming, so one is usually interested in finding an approximate solution $\widetilde{x} \in \mathbb{R}^d$ such that

$$\|U\widetilde{x} - b\|_2 \leq (1+\alpha)\|Ux^* - b\|_2$$

for some $\alpha > 0$.

In the adaptive turnstile update model, we assume an adversary could curate a sequence of T updates $\{v_1, \ldots, v_T\}$ adaptively, where each v_t for $t \in [T]$ could take one of the two forms:

- $v_t \in \mathbb{R}^{n\times d}$, i.e., v_t is an update to the design matrix;
- $v_t \in \mathbb{R}^n$, i.e., v_t is an update to the response vector.

We need to design an algorithm that processes these adaptive updates to either U or b, for simplicity we will use (U_t, b_t) to denote the design matrix and response vector after the t-th update. Our algorithm needs to respond with an approximate solution x_t to the t-th ℓ_2 regression:

$$\|U_t x_t - b_t\|_2 \leq (1+\alpha)\min_{x\in\mathbb{R}^d} \|U_t x - b_t\|_2.$$

The main objective is to design an algorithm that

- Uses space that is sublinear in n, sublinear in T and small polynomial in d;
- Has efficient update that depends on the size of v_t and small polynomial in d.

In the following, for the simplicity of presentation, we will make common assumptions that the length of the stream is $T = \mathrm{poly}(n)$, and that in any round $t \leq T$, entries of U_t, b_t are always in the range $[-n^\gamma, n^\gamma]$ for some constant $\gamma > 0$. These assumptions can be removed at the expense of introducing additional factors to our results.

Sketching will be a key algorithmic tool to speed up the procedure of solving ℓ_2 regression, we introduce them in the following.

Definition D.1 (Oblivious Subspace Embedding)**.** *Fix dimension parameters n, d, approximation factor α, and failure probability β. Suppose $\mathcal{D}$ is a distribution on $r \times n$ matrices S, where r is a function of n, d, α, and β. Suppose that with probability at least $1-\beta$, for any fixed $n \times d$ matrix U, a matrix S drawn from $\mathcal{D}$ satisfies: for all vectors $x \in \mathbb{R}^d$,*

$$\|SUx\|_2^2 = (1 \pm \alpha)\|Ux\|_2^2,$$

then we call $\mathcal{D}$ an (α, β)-Oblivious Subspace Embedding (OSE).

We will mainly need the OSE property with $\alpha = O(1)$ and $\beta = O(1)$. We refer to this as $O(1)$-OSE. We list a collection of sketching matrices that will be used throughout this section.

Definition D.2 (Count Sketch (Charikar et al., 2002)). *Let $S \in \mathbb{R}^{r\times n}$ be constructed via the following procedure: Randomly draw $h : [n] \to [r]$ from a pairwise independent hash family, and draw $\sigma : [n] \to \{-1, 1\}$ from a 4-wise independent hash family. For each of the n columns $S_{i\in[n]}$, we choose a row $h(i) \in [r]$ and an element $\sigma(i)$ from $\{-1, 1\}$. We set $S_{h(i),i} = \sigma(i)$ for all $i \in [n]$, and set all other entries of S to be 0. We call such S a Count Sketch matrix.*

Definition D.3 (Gaussian Sketching Matrix). *We say $S \in \mathbb{R}^{r\times n}$ is a Gaussian sketching matrix if all entries are independently sampled from the distribution $\mathcal{N}(0, 1/r)$.*

Definition D.4 (Subspace Randomized Hardamard Transform (SRHT) (Lu et al., 2013)). *The Subspace Randomized Hardamard Transform (SRHT) matrix $S \in \mathbb{R}^{r\times n}$ is defined as the scaled matrix product $S := \frac{1}{\sqrt{r}} PHD$, where each row of matrix $P \in \{0, 1\}^{r\times n}$ contains exactly one 1 at a random position, H is the $n \times n$ Hadamard matrix, and D is a $n \times n$ diagonal matrix with each diagonal entry being a value in $\{-1, 1\}$ with equal probability.*

We will use the following differentially private median procedure:

Theorem D.5. *There exists an $(\varepsilon, 0)$-differentially private algorithm that given a database $S \in X^*$, outputs an element $x \in X$ such that with probability at least $1 - \beta$, there are at least $|S|/2 - \Gamma$ elements in S that are bigger or equal to x, and there are at least $|S|/2 - \Gamma$ elements in S that are smaller or equal to x, where $\Gamma = O(\frac{1}{\varepsilon} \log \frac{|X|}{\beta})$. Moreover, private median runs in time*

$$O(\varepsilon^{-1}|S| \log^3(|X|/\beta) \cdot \mathrm{poly} \log |S|).$$

We will use $\mathrm{PMEDIAN}(x_1, \ldots, x_s)$ to denote the invocation of private median on $x_1, \ldots, x_s$.

D.2. Generic Algorithm with Private Median

A critical property we will be leveraging in designing the algorithm is the ℓ_∞ *guarantee* of the sketched solution (Price et al., 2017; Song et al., 2023c): this guarantee states that the sketched solution $\widetilde{x}$ and the optimal solution x^* are close not only in terms of their *costs*, i.e., $\|U\widetilde{x} - b\|_2 = (1 \pm \alpha)\|Ux^* - b\|_2$, but are close in the sense that $\|\widetilde{x} - x^*\|_\infty$ is small. Note that a naïve bound of $\|\widetilde{x} - x^*\|_\infty$ is just $\|\widetilde{x} - x^*\|_2$, as one could have the scenario that all the discrepancy concentrate in a few coordinates with most coordinates are the same. When S is an SRHT matrix, (Price et al., 2017) shows that $\|\widetilde{x} - x^*\|_2$ is too pessimistic and a stronger bound exists, in particular, they prove

$$\|\widetilde{x} - x^*\|_\infty \leq \frac{\alpha}{\sqrt{d}} \|Ux^* - b\|_2 \cdot \frac{1}{\sigma_{\min}(U)} \tag{2}$$

holds with probability $1 - \beta$. The number of rows required is further improved in (Song et al., 2023c). To improve the space usage and runtime efficiency, we further compose the SRHT matrix with a Count Sketch matrix.

Lemma D.6. *Let $\mathcal{D}$ be a distribution of matrix product $S_{\mathrm{SRHT}} \cdot S_{\mathrm{CS}}$, where $S_{\mathrm{SRHT}} \in \mathbb{R}^{r\times m}$ is an SRHT matrix and $S_{\mathrm{CS}} \in \mathbb{R}^{m\times n}$ is a Count Sketch matrix. For some $r = O(\frac{d \log^3(n/\beta)}{\alpha^2})$ and $m = O(d^2 + \frac{d}{\alpha^2\beta})$, $\mathcal{D}$ satisfies the ℓ_∞ guarantee in Equation 2. We will refer this property as (α, β)-accuracy.*

Proof. As shown in (Price et al., 2017), all we need is that both S_{SRHT} and S_{CS} are drawn from distributions that individually satisfy Equation (2) (up to a constant factor loss in failure probability and approximation factor). The main result of (Song et al., 2023c) proves that SRHT with r rows satisfies Equation (2).

For S_{CS}, following the core lemma in (Song et al., 2023c), it suffices to show that it is an Oblivious Subspace Embedding (OSE) with an $O(1)$ approximation factor, and it satisfies the following Matrix Multiplication guarantee: for any fixed vectors $g, h \in \mathbb{R}^n$,

$$\Pr[|g^\top S_{\mathrm{CS}}^\top S_{\mathrm{CS}} h - g^\top h| \geq \frac{\alpha}{\sqrt{m}} \|g\|_2 \|h\|_2] \leq \beta.$$

It is shown, e.g. in (Woodruff, 2014) that

1. Count Sketch with $m = O(d^2/\operatorname{poly}\log d)$ is an $O(1)$-OSE, and
2. Count Sketch with $m = O(d/\alpha^2\beta)$ satisfies the Approximate Matrix Multiplication property.

This concludes that $\mathcal{D}$ satisfies the ℓ_∞ guarantee in Equation (2). □

We are now ready to describe our algorithm. Throughout, we will let $\alpha \in (0,1)$ be the approximation factor, $\beta \in (0,1)$ be the failure probability and we require an extra parameter κ which is an upper bound on the condition number of U_t throughout the update sequence. Since our algorithm would utilize private median, we let $\varepsilon_{\mathrm{DP}}$ to denote the privacy parameter which will be specified later. We will refer this algorithm to ADAPTIVEREGDP.

Initialization.

- (Parameters for sketches): Let $\alpha' = \frac{\alpha}{\kappa}$ and $\beta' = 0.01$, set $r = O(\frac{d\log^3(n/\beta')}{\alpha'^2})$ and $m = O(d^2 + \frac{d}{\alpha'^2\beta'})$;
- (Parameters for privacy): Let $\Gamma = O(\frac{1}{\varepsilon_{\mathrm{DP}}}\log\frac{Td|X|}{\beta})$ where $X = \{-n^\gamma, \ldots, -n^{-\gamma}, 0, n^{-\gamma}, \ldots, n^\gamma\}$, set $k = O(\varepsilon_{\mathrm{DP}}\cdot\Gamma\cdot\sqrt{Td\log(1/\beta)})$;
- Prepare k independent copies $S_1, \ldots, S_k \in \mathbb{R}^{r\times n}$ according to Lemma D.6;
- Let $\mathrm{sk}_U^i = S_i U$ and $\mathrm{sk}_b^i = S_i b$ for all $i \in [k]$.

Update.

- Receive update v_t;
- If v_t is an update to U, then update $\mathrm{sk}_U^i \leftarrow \mathrm{sk}_U^i + S_i v_t$ for all $i \in [k]$, otherwise update $\mathrm{sk}_b^i \leftarrow \mathrm{sk}_b^i + S_i v_t$.

Query.

- Sample with replacement $s = \widetilde{O}(1)$ indices from $[k]$, let $j_1, \ldots, j_s$ denote the sampled indices;
- Compute $x_{j_i} = \arg\min_{x\in\mathbb{R}^d} \|\mathrm{sk}_U^i x - \mathrm{sk}_b^i\|_2$ for all $i \in [s]$;
- Compute $g_l = \text{PMEDIAN}((x_{j_1})_l, \ldots, (x_{j_s})_l)$ with privacy budget $\varepsilon_{\mathrm{DP}}$ (Theorem D.5) for all $l \in [d]$;
- Output $g = (g_1, g_2, \ldots, g_d)$.

We now prove the privacy and utility of our algorithm.

Lemma D.7. *The algorithm* ADAPTIVEREGDP *satisfies (ε,δ)-differential privacy w.r.t. the collection of random strings $\mathcal{R}$ for $\varepsilon := 1/100$ and $\delta := \beta/100$.*

Proof. By Theorem D.5, each instance of primed is $(\varepsilon_{\mathrm{DP}}, 0)$-differential private. But since we sample $s = \widetilde{O}(1)$ copies of the oblivious algorithm to use, this amplifies the privacy of each PMEDIAN call to $(\frac{6s}{K}\varepsilon_{\mathrm{DP}}, 0)$-DP by Theorem B.3. In total, we have at most $d\cdot T$ instances of PMEDIAN.

By the advanced composition theorem, the entire algorithm is (ε,δ)-differential private for

$$\varepsilon = \sqrt{2Td\ln(100/\beta)}\cdot(\frac{6s}{k}\varepsilon_{\mathrm{DP}}) + 2Td\cdot(\frac{6s}{k}\varepsilon_{\mathrm{DP}})^2 \le 1/100$$

and $\delta = 100/\beta$, since we set $K = 200\cdot 6s\varepsilon_{\mathrm{DP}}\cdot\sqrt{2Td\ln(100/\beta)}$. □

In the following discussion, we follow previous work (Hassidim et al., 2022) and assume the returned solution vectors to the regressions have their coordinates in the range $[-n^\gamma, -1/n^\gamma]\cup\{0\}\cup[1/n^\gamma, n^\gamma]$.

Lemma D.8. *With probability at least $1-\beta$, in all rounds $t \in [T]$ during the update sequence, The algorithm* AdaptiveRegDP *outputs $\widetilde{g}$ that satisfies*

$$\|U_t\widetilde{g} - b_t\|_2 \leq (1+\alpha)\min_{x\in\mathbb{R}^d}\|U_t x - b_t\|_2,$$

for underlying design matrix U_t and vector b_t.

Proof. Consider any round $t \in [T]$ during the stream, let (U_t, b_t) be the underlying vectors defined by the stream up to round t. And let $\sigma_{\min}$ denote the minimum singular value of U_t and $\sigma_{\max}$ denote the maximum. Let $f_{U_t,b_t}(r)$ be the indicator for the following event:

$$\begin{aligned}\|x^* - x_t\|_\infty \leq \frac{\alpha'}{\sqrt{d}}\|U_t x^* - b_t\|_2 \cdot \frac{1}{\sigma_{\min}} \text{ (i.e., Equation (2)) :}\\ x^* := \arg\min_{x\in\mathbb{R}^d}\|U_t x - b_t\|_2\\ x_t := \arg\min_{x\in\mathbb{R}^d}\|S(U_t x - b_t)\|_2 \text{ where } S \text{ is generated as Lemma D.6, using random string } r.\end{aligned}$$

For $\varepsilon = 1/100$ and $\delta = \beta/100$, observe that $s \gg \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon}{\delta})$. Thus, we can apply Theorem B.4 with $n = s$ to show that

$$|\mathop{\mathbb{E}}_r[f_{U_t,b_t}(r)] - \frac{1}{s}\sum_{i=1}^{s} f_{U_t,b_t}(r_{j_i})| \leq 10\varepsilon = 1/10.$$

with probability at least $1 - \delta/\varepsilon = 1-\beta$. In the following, we condition on the event that this holds.

We have $\mathbb{E}_r[f_{U_t,b_t}(r)] \geq 9/10$ by the (α',β')-accuracy of each copy of sketch. Therefore, at least $4s/5$ of the samples $\{x_{j_i} : i \in [s]\}$ satisfies $\|x^* - x_{j_i}\|_\infty \leq \frac{\alpha'}{\sqrt{d}}\|U_t x^* - b_t\|_2 \cdot \frac{1}{\sigma_{\min}}$. We call such x_{j_i} a "good approximation".

The algorithm runs PMedian on each coordinate $l \in [d]$ across all approximated vectors $x_{j_1}, x_{j_2}, \cdots, x_{j_s}$. For each $l \in [d]$, Theorem D.5 combined with our choice of $s = 100\Gamma$ guarantees that with probability at least $1 - \beta/(Td)$, we have

$$|\{i \in [s] : (x_{j_i})_l \geq (g)_l\}| \geq 4s/10 \text{ and } |\{i \in [s] : (x_{j_i})_l \leq (g)_l\}| \geq 4s/10.$$

Since there are at least $4s/5$ good approximations satisfying Equation (2), this means there exist good approximations x_{j_p}, x_{j_q} such that $(g)_l \in [(x_{j_p})_l, (x_{j_q})_l]$, thus $|(g)_l - (x^*)_l| \leq \frac{\alpha'}{\sqrt{d}}\|U_t x^* - b_t\|_2 \cdot \frac{1}{\sigma_{\min}}$. This holds simultaneously for all $l \in [d]$ and all k independent copies with probability at least $1-\beta$.

Condition on the above event, we have

$$\begin{aligned}\|U_t\widetilde{g} - b_t\|_2 &\leq \|U_t x^* - b_t\|_2 + \|U_t(\widetilde{g} - x^*)\|_2\\ &\leq \|U_t x^* - b_t\|_2 + \sigma_{\max}\|\widetilde{g} - x^*\|_2\\ &\leq \|U_t x^* - b_t\|_2 + \sigma_{\max}(\sqrt{d}\cdot\|\widetilde{g} - x^*\|_\infty)\\ &\leq \|U_t x^* - b_t\|_2 + \frac{\sigma_{\max}}{\sigma_{\min}}(\sqrt{d}\frac{\alpha'}{\sqrt{d}}\|U_t x^* - b_t\|_2)\\ &= (1 + \frac{\sigma_{\max}}{\sigma_{\min}}\alpha')\|U_t x^* - b_t\|_2.\end{aligned}$$

By setting $\alpha' = \frac{\alpha}{\kappa}$, this gives a $(1+\alpha)$-approximation. Overall, with probability at least $1-\beta$, all the approximation vectors are accurate to within a multiplicative error of $(1+\alpha)$. □

It remains to show that AdaptiveRegDP is both time- and space-efficient.

Theorem D.9. *Given a sequence of T adaptive updates $\{v_1, \ldots, v_T\}$, algorithm* AdaptiveRegDP *has the following specifications:*

- *It preprocesses* (U, b) *in time* $\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(U) + \mathrm{nnz}(b) + d^3 + d^2\kappa^2/\alpha^2))$*;*
- *It uses space* $\widetilde{O}(\sqrt{T} \cdot d^{2.5}\kappa^2/\alpha^2)$*;*
- *Given an update* v_t *for* $t \in [T]$*, it takes time* $\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(v_t) + d^3 + d^2\kappa^2/\alpha^2))$ *to update;*
- *It outputs a* $(1+\alpha)$*-approximate solution* x_t *in time* $\widetilde{O}(d^{\omega+1}\kappa^2/\alpha^2)$*.*

Proof. We prove the theorem item by item.

Preprocessing time. During preprocessing, we prepare $k = \widetilde{O}(\sqrt{Td})$ independent copies of sketching matrices due to Lemma D.6 and compute S_iU, S_ib for all $i \in [k]$. Note that S_i is a composition of a Count Sketch matrix and an SRHT matrix, so S_iU takes time $O(\mathrm{nnz}(U))$ for the Count Sketch, and this results in a matrix of size $m \times d$, applying SRHT to this matrix takes time $\widetilde{O}(md) = \widetilde{O}(d^3 + d^2\kappa^2/\alpha^2)$. Hence, the overall time for preprocessing is

$$\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(U) + \mathrm{nnz}(b) + d^3 + d^2\kappa^2/\alpha^2)).$$

Space usage. The algorithm stores $\mathcal{R}$ and $\mathrm{sk}_U^i, \mathrm{sk}_b^i$ for all $i \in [k]$. We start by considering the random strings to generate k pairs of SRHT and Count Sketch matrices. To store the pairwise and 4-wise independent hash functions for Count Sketch, we only need $O(1)$ bits of space. To store an $r \times m$ SRHT matrix $S_{\mathrm{SRHT}} = \frac{1}{\sqrt{r}}PHD$, since H is deterministic, we only consider $P \in \mathbb{R}^{r\times m}$ and $D \in \mathbb{R}^{m\times m}$. P contains exactly one 1 in each row (and contains 0 everywhere else), thus can be stored in $O(r \log m)$ bits of space. D is a diagonal matrix containing $\{-1, 1\}$, which can be stored in $O(m)$ bits of space. Therefore, R in total takes

$$O(\frac{d\log^3 n}{\alpha'^2}\log m + d^2 + \frac{d}{\alpha'^2}) = O(\frac{d\kappa^2 \operatorname{poly}\log n}{\alpha^2} + d^2)$$

bits of space.

For $\mathrm{sk}_U^k, \mathrm{sk}_b^k$ for $i \in [k]$, the total bits of space is

$$\begin{aligned} O(k \cdot r \cdot d\log n) &= O(\operatorname{poly}\log n \cdot \log(\frac{T}{\beta}) \cdot \sqrt{Td\log(\frac{1}{\beta})} \cdot \frac{d\kappa^2\log^3 n}{\alpha^2} \cdot d\log n) \\ &= O(\frac{d^{2.5}\kappa^2\sqrt{T}}{\alpha^2} \cdot \operatorname{poly}\log(n, L, 1/\beta)) \end{aligned}$$

assuming a word size of $O(\log n)$ bits. This is also asymptotically the total space usage.

Update time. For update, we simply apply the sketch S_i to the update v_t for all $i \in [k]$, hence the total update time is $\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(v_t) + d^3 + d^2\kappa^2/\alpha^2))$.

Query time. To answer the query, we need to solve a sketched regression for s instances, where each regression could be solved in time $\widetilde{O}(rd^{\omega-1}) = \widetilde{O}(d^{\omega+1}\kappa^2/\alpha^2)$. The private median is then performed over s numbers, and each private median runs in time $\widetilde{O}(s)$. Repeating this procedure for all d coordinates, the overall time for forming g is then $\widetilde{O}(d)$. Hence, the query time is

$$\widetilde{O}(d^{\omega+1}\kappa^2/\alpha^2).$$

□

D.3. Improved Algorithm with Bounded Computation Path Technique

When the condition number κ is as large as $\operatorname{poly} n$, a quadratic dependence on κ is prohibitive. We provide an algorithm for such case based on the *bounded computation path technique* (Hassidim et al., 2022). Let $\mathcal{P}$ denote the set of all possible computation paths given the adaptive update sequence $\{v_1, \ldots, v_T\}$. The intuition is that the total number of possible computation paths $|\mathcal{P}|$ is at most $n^{\Theta(dT)}$, hence we could prepare a large Gaussian sketching matrix with size roughly $\log|\mathcal{P}|$, that is guaranteed to succeed against all such paths. In the case where the updates are stable, e.g., when the updates are always concentrated in a few coordinates, then $|\mathcal{P}|$ could be much smaller than $n^{\Theta(T)}$. This leads to a more efficient alternative and a better dependence on the condition number κ. We will refer to this algorithm as AdaptiveRegPath.

Initialization.

- (Parameters for rounding): Let $\mathcal{P}$ be the set of all possible output sequences of the algorithm and let $\beta_0 = \beta/(C \cdot |\mathcal{P}|)$ for large constant C;
- (Parameters for sketches): Let $r = O((d + \log(1/\beta_0))/\alpha^2)$, generate the Gaussian sketching matrix $S \in \mathbb{R}^{r \times n}$ by a length-$\widetilde{O}(r)$ bits random seed $\mathtt{r}_S$;
- Let $\mathrm{sk}_U = SU$ and $\mathrm{sk}_b = Sb$.

Update.

- Receive update v_t;
- Compute Sv_t using $\mathtt{r}_G$ and Lemma D.13, if v_t is an update to U then $\mathrm{sk}_U \leftarrow \mathrm{sk}_U + Sv_t$, otherwise $\mathrm{sk}_b \leftarrow \mathrm{sk}_b + Sv_t$.

Query.

- Compute $x_t = \arg\min_{x \in \mathbb{R}^d} \|\mathrm{sk}_{U_t} x - \mathrm{sk}_{b_t}\|_2$;
- Round x_t to the nearest point in any sequence in $\mathcal{P}$, denote it by g;
- Output g.

We start by showing the utility guarantee of AdaptiveRegPath when the updates are *oblivious*.

Claim D.10. *If the update sequence is oblivious, at each round $t \in [T]$,* AdaptiveRegPath *outputs a vector g that satisfies*

$$\|U_t g - b_t\|_2 \leq (1+\alpha) \min_{x \in \mathbb{R}^d} \|U_t x - b_t\|_2$$

for underlying input matrix U_t and vector b_t, with probability at least $1 - \beta_0$.

This follows from our setting of $r = O((d + \log(1/\beta_0))/\alpha^2)$, which is sufficient for the distribution of Gaussian matrices to be an (α, β_0)-OSE (Woodruff, 2014).

Lemma D.11. *The Gaussian sketching matrix S in* AdpativeRegPath *can be pseudo-randomly generated using a random seed of length $\widetilde{O}(d + \log(1/\beta_0)/\alpha^2)$ bits, while still satisfying Claim D.10.*

Proof Sketch. This can be seen by opening up the proof e.g., in Section 2.1 of (Woodruff, 2014) that the Gaussian distribution satisfies OSE. The i.i.d. property is used for showing that the random Gaussian matrix is a JL transform (Lemma 2.12 in (Williams, 2012)). However, there one needs to consider at most an $O(\log(9^d/\beta_0)/\alpha^2)$-tuple of Gaussian variables, where 9^d is the size of the $1/2$-net for a unit sphere in the subspace that we hope to embed in. Therefore the proof goes through as long as the entries of S are generated using at least $r = O(d + \log(1/\beta_0)/\alpha^2)$-wise independence. It is well known that such an r-wise independent hash function can be compressed as a length $\widetilde{O}(r)$-bit seed, which concludes the proof. □

Lemma D.12. *With probability at least $1 - \beta$, in all rounds $t \in [T]$,* AdaptiveRegPath *given an adaptive update v_t outputs a vector g that satisfies:*

$$\|U_t g - b_t\|_2 \leq (1+\alpha) \min_{x \in \mathbb{R}^d} \|U_t x - b_t\|_2,$$

for underlying input matrix U_t and vector b_t.

Proof. As argued in (Ben-Eliezer et al., 2020), we may assume the adversary to be deterministic. This means, in particular, that the output sequence we provide to the adversary fully determines its stream of updates $v_1, v_2, \cdots, v_T$. Note that the collection of all distinct output sequence after the rounding step is at most $|\mathcal{P}|$. Each output sequence as above uniquely determines a corresponding stream of updates for the deterministic adversary. For every stream, with probability at least

$1-\beta_0$, ADAPTIVEREGPATH will be correct in every round. Applying the union bound over these streams, we can conclude that with probability at least $1 - O(|\mathcal{P}|) \cdot \beta_0 = 1 - \beta$, it outputs a $(1+\alpha)$-approximation in every round against any adversary. □

Lemma D.13. *The time to generate S via r_G is $\widetilde{O}(rn)$.*

Proof Sketch. Note that to generate one column S_i of the sketching matrix, we need r evaluations of the r-wise independent hash function. Using fast multipoint evaluation of polynomials (Gathen & Gerhard, 2013), these evaluations can be done in $\widetilde{O}(r)$ time. Hence, generating S takes $\widetilde{O}(rn)$ time, as desired. □

Theorem D.14. *Given a sequence of T adaptive updates $\{v_1, \ldots, v_T\}$, algorithm* ADAPTIVEREGPATH *has the following specifications:*

- *It preprocesses (U, b) in time $\widetilde{O}(nd^{\omega-2}(d + \log|\mathcal{P}| + \log\frac{1}{\beta})/\alpha^2)$;*
- *It uses space $\widetilde{O}(d(d + \log|\mathcal{P}| + \log\frac{1}{\beta})/\alpha^2)$;*
- *Given an update v_t for $t \in [T]$, it takes time $\widetilde{O}((d + \log|\mathcal{P}| + \log\frac{1}{\beta})/\alpha^2)$ to update;*
- *It outputs a $(1+\alpha)$-approximate solution x_t in time $\widetilde{O}(d^{\omega-1}(d + \log|\mathcal{P}| + \log\frac{1}{\beta})/\alpha^2)$.*

Here β denotes the failure probability.

Proof. We prove the theorem item by item. The parameter r is expanded as $d + \log|\mathcal{P}| + \log\frac{1}{\beta}$ to obtain the theorem.

Preprocessing time. During preprocessing, we generate the sketching matrix S using our random seed, which by Lemma takes $\widetilde{O}(rn)$ time. Afterward, computing the sketches is dominated by computing the matrix product of r-by-n matrix S and n-by-d matrix U, which is $\widetilde{O}(rnd^{\omega-2})$.

Space usage. The sketches sk_U and sk_b together take $\widetilde{O}(dr)$ bits of space, and the random seed takes $\widetilde{O}(r)$ bits of space.

Update time. We first generate one column of the sketching matrix, which takes $\widetilde{O}(r)$ time. Then we compute the inner product between this column and the update, which takes at most $\widetilde{O}(r)$ time.

Query time. To answer the query, we solve a sketched regression instance in time $\widetilde{O}(rd^{\omega-1})$. After that, we round it to the nearest point in $\mathcal{P}$, which can be done in $\widetilde{O}(\log|\mathcal{P}|)$ time using binary search. □

D.4. Handling Sparse Label Shifts with Preconditioner

In machine learning, it is common that the design or data matrix remains unchanged, as it might be generated via embedding raw data into high dimensional vectors, and these embeddings are expensive to compute. The response or label vector, in contrary, might constantly drift. Hence (Cherapanamjeri et al., 2023) studies a model where only b receives a sparse update v_t with $\|v_t\|_0 \le s$. For example, b could be the count of a large class of images, and only a few popular classes get their counts incremented frequently. In (Cherapanamjeri et al., 2023), they are only able to output regression cost against sparse, adaptive updates to b. Here, we develop an algorithm based on ADAPTIVEREGDP to output the solution vector. We will use ADAPTIVEREGPRECONDITIONER to denote this algorithm.

Initialization.

- (Parameters for sketches): Let $\alpha' = \alpha$ and $\beta' = 0.01$, set $r = O(\frac{d\log^3(n/\beta')}{\alpha'^2})$ and $m = O(d^2 + \frac{d}{\alpha'^2\beta'})$;
- (Parameters for privacy): Let $\Gamma = O(\frac{1}{\varepsilon_{\mathrm{DP}}}\log\frac{Td|X|}{\beta})$ where $X = \{-n^\gamma, \ldots, -n^{-\gamma}, 0, n^{-\gamma}, \ldots, n^\gamma\}$, set $k = O(\varepsilon_{\mathrm{DP}} \cdot \Gamma \cdot \sqrt{Td\log(1/\beta)})$;

- (Parameters for batch size): Let T be a parameter denote the batch size depending on $\mathrm{nnz}(U), d$ and α. Let $\mathrm{counter} = 1$ be the batch counter parameter.
- Prepare k independent copies $S_1, \ldots, S_k \in \mathbb{R}^{r \times n}$ according to Lemma D.6;
- Compute a preconditioner $P \in \mathbb{R}^{d \times d}$ such as $\kappa(AP) = O(1)$;
- Prepare $M_i = (S_i U P)^\dagger S_i$ and $\mathrm{sk}^i = M_i b$ for all $i \in [k]$.

Update.

- If $\mathrm{counter} = T$, regenerate sketching matrices $S_1, \ldots, S_k \in \mathbb{R}^{r \times n}$, set $M_i = (S_i U P)^\dagger S_i$ and $\mathrm{sk}^i = M_i b$ for all $i \in [k]$. Reset the $\mathrm{counter} = 1$;
- Receive update v_t;
- Update $b \leftarrow b + v_t$ and $\mathrm{sk}^i \leftarrow \mathrm{sk}^i + M_i v_t$ for all $i \in [k]$.

Query.

- Sample with replacement $s = \widetilde{O}(1)$ indices from $[k]$, let $j_1, \ldots, j_s$ denote the sampled indices;
- Compute $g_l = \textsc{PMedian}((\mathrm{sk}^{j_1})_l, \ldots, ((\mathrm{sk}^{j_s})_l)$ with privacy budget $\varepsilon_{\mathrm{DP}}$ (Theorem D.5) for all $l \in [d]$;
- Let $\widetilde{g} = (g_1, g_2, \ldots, g_d)$, output $g = P\widetilde{g}$.

We note that the privacy and utility guarantee remains the same as AdaptiveRegDP, so we focus on the complexity. Instead of fixing the length of update sequence in advance, we allow for arbitrary length of update sequence, and we will restart the data structure every T updates.

Theorem D.15. *Given a sequence of adaptive updates $\{v_1, v_2, \ldots, \}$, algorithm* AdaptiveRegPreconditioner *has the following specifications:*

- *It has amortized update time $\widetilde{O}(\sqrt{d/T} \cdot (\mathrm{nnz}(U) + \mathrm{nnz}(b) + d^3 + d^\omega/\alpha^2) + \sqrt{Td} \cdot (s + d^3 + d^2/\alpha^2))$;*
- *It outputs a $(1+\alpha)$-approximate solution x_t in time $\widetilde{O}(d^2)$.*

Proof. We analyze over batches. At the start of each batch, we generate k sketching matrices and $S_i U$ for all $i \in [k]$, this step takes $\widetilde{O}(\sqrt{Td} \cdot (\mathrm{nnz}(U) + d^3 + d^2/\alpha^2))$ time. Right multiplying by P then computing the pseudoinverse takes $\widetilde{O}(\sqrt{Td} \cdot d^\omega/\alpha^2)$ time. Finally, note that we don't need to right multiply by S_i, rather we could first compute $S_i b$ in $\mathrm{nnz}(b) + d^2 + d/\alpha^2$ time, then multiplying the matrix with the vector in $\widetilde{O}(\sqrt{Td} \cdot rd) = \widetilde{O}(\sqrt{Td} \cdot d^2/\alpha^2)$ time. Hence, the amortized time for this process over T steps is

$$\widetilde{O}((\sqrt{d} \cdot (\mathrm{nnz}(U) + \mathrm{nnz}(b)) + d^{3.5} + d^{\omega+0.5}/\alpha^2)/\sqrt{T}).$$

When receiving v_t, it takes s time to update the response vector, and $s + d^3 + d^2/\alpha^2$ time to update sk^i. Hence, the overall time is

$$\widetilde{O}(\sqrt{T} \cdot (sd^{0.5} + d^{3.5} + d^{2.5}/\alpha^2)).$$

Now, to output a query, we simply compute private median over s sampled solution vectors in d coordinates, and output $P\widetilde{g}$ in d^2 time.

In the next few paragraphs, we will explain how to make choice for T.

For the case when $\mathrm{nnz}(U)$ terms are small, and the ideal $\omega \approx 2$, the right choice of $T = \Theta(1)$.

For the case when $\mathrm{nnz}(U)$ terms are small, and we use the current omega, the right choice is $T = \Theta(d^{\omega-2})$.

For the case when $\mathrm{nnz}(U)$ terms are large, then we just need to balance $\sqrt{d}\,\mathrm{nnz}(U)/\sqrt{T}$ with $(d^{3.5} + d^{2.5}/\alpha^2)\sqrt{T}$ which means we need to chose $T = \Theta(\frac{\mathrm{nnz}(U)}{d^3 + d^2/\alpha^2})$. □

To obtain the optimal choice for the batch size T, we case on $\mathrm{nnz}(U)$. Let C be some constant.

Claim D.16. *If $\mathrm{nnz}(U) \leq C \cdot (d^3 + d^\omega/\alpha^2)$, then let T be $O((d^3 + d^\omega/\alpha^2)/s)$, the update time of the algorithm is $\widetilde{O}(\sqrt{s} \cdot (d^2 + d^{\omega/2+0.5}/\alpha))$.*

Claim D.17. *If $\mathrm{nnz}(U) > C \cdot (d^3 + d^\omega/\alpha^2)$, then let T be $O(\mathrm{nnz}(U)/s)$, the update time of the algorithm is $\widetilde{O}(\sqrt{d \cdot \mathrm{nnz}(U)/s})$.*

E. Improved Algorithm Against Adversarial Attack for Hamming Space LSH

In a recent work, (Kapralov et al., 2024) shows that for Hamming space, there exists a query efficient algorithm that can quickly compute a point v adaptively, such that there exists $u \in U$ such that $\|u - v\| \leq r$ (here $\|\cdot\|$ is the Hamming distance) but the data structure returns nothing. We state their result here.

Definition E.1. *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset, we say a point $z \in U$ is an isolated point if for all points $u \in U$ with $u \neq z$, $\|u - z\| \geq 2cr$. Here $\|\cdot\|$ is the Hamming distance.*

Theorem E.2 ((Kapralov et al., 2024))**.** *Let n denote the size of dataset, d be the dimension, (c, r) be the parameters for ANN, λ be a complexity parameter. Suppose they satisfy the following relations:*

- $cr \leq d$;
- $\ln^3 n \leq r \leq d/\ln n$;
- $\lambda \leq \min\{\frac{r}{\ln n}, n^{1/8}\}$;
- $c \geq 1 + \frac{\ln \lambda}{\ln n}$.

Then with probability at least $1/4 - 1/n$, there exists an algorithm that finds a point q such that an isolated point z in the dataset is at most r away from q, but the algorithm returns nothing. The algorithm makes $O(\log(cr) \cdot \lambda)$ queries to the LSH.

In their original theorem statement, they also require $c \leq \ln n$. This is unnecessary, as it is derived using the upper bound $r \leq d/\ln n$ and $cr \leq d$. If r is much smaller, then c can be larger. Their algorithm starts from the isolated point z, then gradually moves away from z while ensuring no other points could collide with it. In the end, it moves at most r-away from z, but with high probability, no point in U has been hashed to the same bucket as it.

To combat such an adversarial attack, one could either repeat the data structures d times or T times, which are both relatively inefficient. We will show that whenever the dimension d is relatively small, then we could use Theorem 1.6 with $\sqrt{T} \cdot s$ data structures, against the attack of (Kapralov et al., 2024).

Lemma E.3. *Let n, d be positive integers and $c \geq 1$ and $r > 0$ be parameters. Let $\rho \in (0, 1)$ be a parameter. If $3d = s^\alpha$ and $\alpha \leq \frac{1}{2cr}$, then Assumption 1.3 holds for any n-point set $U \subseteq \{0, 1\}^d$. In particular, if $3d \leq n^{\frac{\rho}{2cr}}$ then Assumption 1.3 holds for all $s \leq n^\rho$.*

Proof. We note that Assumption 1.3 could be rephrased as for each $u \in U$, it can have at most $2cr$-near neighbors. In the Hamming space, it means that they could differ by at most $2cr$ bits. Fix a vector u. The total number of points that differ by at most $2cr$ bits from u is at most

$$\begin{aligned}\sum_{i=1}^{2cr} \binom{d}{i} &\leq 2cr \binom{d}{2cr} \\ &\leq (d \cdot e)^{2cr} \cdot (2cr)^{1-2cr} \\ &\leq (3d)^{2cr}\end{aligned}$$

as long as $2cr \log_s(3d) \leq 1$, we are guaranteed the total number of possible near neighbors is at most s. By a simple calculation, this also gives us the bound when $s \leq n^\rho$. □

For Hamming space LSH, it is known that $\rho = O(1/c)$, and thus the bound on d becomes $d \leq n^{\Theta(1/(c^2 r))}$. In order for it to be useful against Theorem E.2, we could let $r = \ln^3 n$ and $c = 1 + o(1)$. Thus the bound on d becomes $d \leq \exp(-\Theta(\log^2 n)) = n^{o(1)}$. We obtain a corollary against Theorem E.2 in this setting.

Corollary E.4. *If $d = n^{o(1)}$ and with the parameters in Theorem E.2, there exists an adaptive LSH data structure for Hamming space with*

- *Preprocessing time $\widetilde{O}(\sqrt{T} \cdot n^{1+O(\rho)}d)$;*
- *Space usage $\widetilde{O}(\sqrt{T} \cdot n^{1+O(\rho)} + nd)$;*
- *For all $t \in [T]$, with high probability, if $B(v_t, r) \cap U \neq \emptyset$ then it returns a point in $B(v_t, cr) \cap U$ in time $\widetilde{O}(n^{O(\rho)}d)$. Here $B(\cdot, \cdot)$ is the Hamming ball.*

In particular, to be robust against the attack of Theorem E.2, it suffices to pick $T = O(\log(cr) \cdot \lambda)$.

The above corollary essentially asserts that, to be robust against the attack of (Kapralov et al., 2024), it is sufficient to blow up the number of data structures by a $\sqrt{\log(cr) \cdot \lambda}$ factor, even in the presence of an isolated point.

F. Locating the Thin Level via Approximate Counting

If one considers Assumption 1.2 for (c, r)-ANN, it states that for an adaptive query v, if $B(v, r) \cap U \neq \emptyset$ then $|B(v, cr) \cap U| \leq s$. In an iterative process, one usually does not care for a particular r but only wants to find an approximate nearest neighbor, and this is often achieved via binary search over the choice of r. We also note that since our assumption is strictly weaker than constant expansion and doubling dimension, it is completely possible that for some small r, Assumption 1.2 holds for query v and it no longer holds for $2r$. In fact, even if $|B(v, cr) \cap U|$ is small, $|B(v, 2cr) \cap U|$ could be the entirety of U if the diameter of U is between cr and $2cr$. Thus, in order to deploy our data structure, it is crucial that we have the ability to identify the *thin level*, i.e., given v, the largest possible r for which the assumption holds. Recall that due to efficiency concerns and the nature of LSH data structures, we set $s \leq n^{\rho}$.

To do so, we will utilize an approximate near neighbor counting procedure for ℓ_2 norm, developed in (Andoni et al., 2023). In essence, they develop an algorithm based on the LSH scheme of (Andoni et al., 2017) that, given an oblivious query v, it can with high probability return an approximate count `ans` such that

$$(1 - o(1)) \cdot |B(v, r) \cap U| \leq \texttt{ans} \leq |B(v, cr) \cap U| + O(n^{\rho + o(1)}).$$

We first note this provides a tool for us to do binary search on r, to locate the thin level: we could start with small r, and run the approximate counting procedure of (Andoni et al., 2023). Conditioned on these counts being accurate, whenever we are still below or at the thin level, `ans` must be below $O(n^{\rho+o(1)})$. If we are at a level where $|B(v, r) \cap U| \geq \omega(n^{\rho+o(1)})$, then the counting data structure could successfully detect it, so the only troublesome case is when $|B(v, r) \cap U|$ is small but $|B(v, cr) \cap U| \geq \omega(n^{\rho+o(1)})$. We can indeed detect this case by searching over the range $(cr, c^2 r)$, and the data structure should indeed report that the count is already too large. We could then refine our search by looking at the range $(r/c, r)$, in which the correct thin level is guaranteed to be in this interval.

One last issue remains: the (Andoni et al., 2023) data structure only works for oblivious queries; for adaptive queries, they run a net argument with d independent copies. To improve upon that, we note that the data structure only outputs a real number, and thus it falls into the category of *estimation* data structures, which can be augmented via the framework of (Beimel et al., 2022).

Theorem F.1 (Adaptive Approximate Near Neighbor Counting)**.** *Let $U \subseteq \mathbb{R}^d$ be an n-point dataset and $\{v_1, \ldots, v_T\} \subseteq \mathbb{R}^d$ be a sequence of adaptive queries. Let $c \geq 1$, $r \geq 0$ be parameters and $\rho \in (0, 1)$ be a parameter that depends on c. There exists a randomized data structure with*

- *Preprocessing time $\widetilde{O}(\sqrt{T} \cdot n^{1+o(1)}d)$;*
- *Space usage $\widetilde{O}(\sqrt{T} \cdot n^{1+o(1)}d)$;*
- *Given v_t, it outputs a number $\widetilde{\texttt{ans}}$ such that*

$$(1 - o(1)) \cdot |B_{\|\cdot\|_2}(v_t, r) \cap U| \leq \widetilde{\texttt{ans}} \leq 1.01 |B_{\|\cdot\|_2}(v_t, cr) \cap U| + O(n^{\rho + o(1)})$$

holds with probability at least $1 - 1/\operatorname{poly}(n)$. The time to output $\widetilde{\texttt{ans}}$ is $\widetilde{O}(n^{\rho+o(1)}d)$.

Proof. The proof is a combination of the data structure of (Andoni et al., 2023) and augmentation of (Beimel et al., 2022). We start from the oblivious data structure. In the proof of Lemma 3.3 from (Andoni et al., 2023), we note that their argument is to first provide a bound on the noiseless count, and then bound the error incurred by Laplacian noise. The noiseless count `ans` indeed has a bound

$$(1 - o(1)) \cdot |B_{\|\cdot\|_2}(v_t, r) \cap U| \leq \texttt{ans} \leq |B_{\|\cdot\|_2}(v_t, cr) \cap U| + O(n^{\rho + o(1)})$$

with high probability, albeit for oblivious v_t. To augment it for an adaptive adversary, we apply Theorem 3.1 of (Beimel et al., 2022), which states one could use $\sqrt{T}$ copies of the data structure when the output is a real number. Regarding the output quality, only the upper bound is increased by a factor of $1 + \alpha$, if we set $\alpha = 0.01$, we obtain the desired result. □

Given such an adaptive data structure for approximate near neighbor counting, we could then instantiate a meta algorithm for ℓ_2 LSH under Assumption 1.2. Without loss of generality, assume $U \subseteq \mathbb{S}^{d-1}$ and all queries lie on the unit sphere. We first pick a small discretization value τ and apply the transformation $x \mapsto \begin{bmatrix} \lfloor x_1/\tau \rfloor \cdot \tau \\ \vdots \\ \lfloor x_d/\tau \rfloor \cdot \tau \end{bmatrix}$ to all points in U and all queries. We use $\overline{U}$ and $\overline{v}$ to denote transformed points. Note that these points have their entries being a multiple of τ, and the difference of the norm $\|\overline{x} - x\|_2 \leq \sqrt{n}\tau$, if we choose $\tau = n^{-C}$ for a large enough constant C, then the difference is $1/\operatorname{poly}(n)$, which is negligible. The advantage of this discretization framework is that the difference of norms for points over the unit sphere could also be discretized into $O(1/\tau)$ levels. Hence we could perform binary search over these discrete $O(1/\tau)$ levels in $O(\log(1/\tau)) = O(\log n)$ steps. We will prepare $O(\log(1/\tau))$ data structures by initializing an LSH for $r = (c/2)^i\tau$ where $i \in \{0, 1, \ldots, \log(1/\tau)\}$ for $c \geq 2$. This ensures that for any r in this form, there exists a node in between r/c and c, as desired.