Fast Distance Oracles for Any Symmetric Norm

Anonymous Author(s)
Affiliation
Address
email

Abstract

In the *Distance Oracle* problem, the goal is to preprocess n vectors $x_1, x_2, \cdots, x_n$ in a d-dimensional metric space $(\mathbb{X}^d, \|\cdot\|_l)$ into a cheap data structure, so that given a query vector $q \in \mathbb{X}^d$ and a subset $S \subseteq [n]$ of the input data points, all distances $\|q - x_i\|_l$ for $x_i \in S$ can be quickly approximated (faster than the trivial $\sim d|S|$ query time). This primitive is a basic subroutine in machine learning, data mining and similarity search applications. In the case of ℓ_p norms, the problem is well understood, and optimal data structures are known for most values of p.

Our main contribution is a fast $(1+\varepsilon)$ distance oracle for *any symmetric* norm $\|\cdot\|_l$. This class includes ℓ_p norms and Orlicz norms as special cases, as well as other norms used in practice, e.g. top-k norms, max-mixture and sum-mixture of ℓ_p norms, small-support norms and the box-norm. We propose a novel data structure with $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ preprocessing time and space, and $t_q = \widetilde{O}(d + |S| \cdot \mathrm{mmc}(l)^2)$ query time, for computing distances to a subset S of data points, where $\mathrm{mmc}(l)$ is a complexity-measure (concentration modulus) of the symmetric norm. When $l = \ell_p$, this runtime matches the aforementioned state-of-art oracles.

1 Introduction

Estimating and detecting similarities in datasets is a fundamental problem in computer science, and a basic subroutine in most industry-scale ML applications, from optimization [CMF+20, CLP+20, XSS21] and reinforcement learning [SSX21], to discrepancy theory [SXZ22] and covariance estimation [Val12, Alm19], Kernel SVMs [CS09, SSSSC11], compression and clustering [IRW17, MMR22], to mention a few. Such applications often need to quickly compute distances of online (test) points to a subset of points in the input data set (e.g., the training data) for transfer-learning and classification. These applications have motivated the notion of *distance oracles* (DO) [Pel00, GPPR04, WP11]: In this problem, the goal is to preprocess a dataset of n input points $X = (x_1, x_2, \ldots, x_n)$ in some d-dimensional metric space, into a small-space data structure which, given a query vector q and a subset $\mathsf{S} \subseteq [n]$, can quickly estimate all the distances $\mathsf{d}(q, x_i)$ of q to S (note that the problem of estimating a *single* distance $\mathsf{d}(q, x_i)$ is not interesting in $\mathbb{R}^d$, as this can be trivially done in $O(d)$ time, which is necessary to merely read the query q). The most well-studied case (in both theory and practice) is when the metric space is in fact a *normed* space, i.e., the data points $\{x_i\}_{i\in[n]} \in \mathbb{R}^d$ are endowed with some predefined norm $\|\cdot\|$, and the goal is to quickly estimate $\|x_i - q\|$ simultaneously for all i, i.e., in time $\ll nd$ which is the trivial query time. Distance oracles can therefore be viewed as generalizing *matrix-vector* multiplication: for the inner-product distance function $\langle x_i, q\rangle$, the query asks to approximate $X \cdot q$ in $\ll nd$ time.

For the most popular distance metrics—the Euclidean distance (ℓ_2-norm) and Manhattan distance (ℓ_1-norm)—classic dimension-reduction (sketching) provide very efficient distance oracles [JL84, AC09, LDFU13]. However, in many real-world problems, these metrics do not adequately capture similarities between data points, and a long line of work has demonstrated that more com-

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

plex (possibly *learnable*) metrics can lead to substantially better prediction and data compression [DKJ+07]. In particular, many works over the last decade have been dedicated to extending various optimization problems beyond Euclidean/Manhattan distances, for example in (kernel) linear regression [SWY+19], approximate nearest neighbor [ANN+17, ANN+18], sampling [LSV18], matrix column subset selection [SWZ19], and statistical queries [LNRW19].

For ℓ_p norms, the DO problem is well-understood [BYJKS04],where the standard tool for constructing the data structure is via randomized *linear sketching* : The basic idea is to reduce the dimension (d) of the data points by applying some *sketching matrix* $\Phi \in \mathbb{R}^{m \times d}$ ($m \ll d$) to each data point x_i and store the sketch $\Phi x_i \in \mathbb{R}^m$. For a query point q, linearity then allows to estimate the distance from $\Phi(q - x_i)$. The seminal works of [JL84, AMS99, CCF04, TZ12] developed polylogarithmic-size sketching methods for the ℓ_2-norm, which was extended, in a long line of work, to any ℓ_p norm with $0 < p < 2$ [Ind06, KNPW11, CN20]. For $p > 2$, the sketch-size (d) becomes *polynomial*, yet still sublinear in d [SS02, BYJKS04].

In this work, we consider general *symmetric norms*, which generalize ℓ_p norms. More formally, a norm norm $l : \mathbb{R}^d \to \mathbb{R}$ is symmetric if, for all $x \in \mathbb{R}^d$ and every $d \times d$ permutation matrix P, it satisfies $l(Px) = l(x)$ and also $l(|x|) = l(x)$, where $|x|$ is the coordinate-wise absolute value of x (for a broader introduction, see [Bha97] Chapter IV). Important special cases of symmetric norms are ℓ_p norms and *Orlicz norms* [ALS+18], which naturally arise in harmonic analysis and model data with sub-gaussian properties. Other practical examples of symmetric norms include top-k norms, max-mix of ℓ_p norms, sum-mix of ℓ_p norms, the k-support norm [AFS12] and the box-norm [MPS14].

Several recent works have studied dimension-reduction (sketching) for special cases of symmetric norms such as the Orlicz norm [BBC+17, SWY+19, ALS+18], for various numerical linear algebra primitives [SWZ19]. These sketching techniques are quite ad-hoc and are carefully tailored to the norm in question. It is therefore natural to ask whether *symmetry alone* is enough to guarantee dimensionality-reduction for symmetric similarity search, in other words

Is there an efficient $(1+\epsilon)$-distance oracle for general symmetric norms?

By "efficient", we mean small space and preprocessing time (ideally $\sim nd$), fast query time ($\ll n|S|$ for a query $(q, S \subseteq [n])$) and ideally supporting dynamic updates to x_i's in $\widetilde{O}(d)$ time. Indeed, most ML applications involve rapidly-changing *dynamic datasets*, and it is becoming increasingly clear that static data structures do not adequately capture the requirements of real-world applications [JKDG08, CMF+20, CLP+20]. As such, it is desirable to design a *dynamic distance oracle* which has both small update time (t_u) for adding/removing a point $x_i \in \mathbb{R}^d$, and small query time (t_q) for distance estimation. We remark that most known DOs are dynamic by nature (as they rely on linear sketching techniques), but for general metrics (e.g., graph shortest-path or the ℓ_∞ norm) this is much less obvious, and indeed *linearity* of encoding/decoding will be a key challenge in our data structure (see next section). The problem is formally defined as follows:

Definition 1.1 (Symmetric-Norm Distance Oracles)**.** *Let $\|\cdot\|_{\text{sym}}$ denote the symmetric norm. We need to design a data structure that efficiently supports any sequence of the following operations:*

- $\textsc{Init}(\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1))$. *The data structure takes n data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input.*
- $\textsc{UpdateX}(z \in \mathbb{R}^d, i \in [n])$. *Update the data structure with the i-th new data point z.*
- $\textsc{EstPair}(i, j \in [n])$ *Outputs a number* pair *such that* $(1-\epsilon)\|x_i - x_j\|_{\text{sym}} \leq \text{pair} \leq (1+\epsilon) \cdot \|x_i - x_j\|_{\text{sym}}$ *with probability at least* $1 - \delta$.
- $\textsc{Query}(q \in \mathbb{R}^d)$. *Outputs a vector $\widetilde{d} \in \mathbb{R}^n$ such that $\forall i \in [n], (1-\epsilon)\|q - x_i\|_{\text{sym}} \leq \text{dst}_i \leq (1+\epsilon)\|q - x_i\|_{\text{sym}}$. with probability at least $1 - \delta$.*

where $\|x\|_{\text{sym}}$ is the symmetric norm of vector x.

This problem can be viewed as an *online* version of the (approximate) *closest-pair* problem [Val12], which asks to find the closest pair of points among an *offline batch* of data points $X = x_1, \ldots, x_n \in \mathbb{R}^d$, or equivalently, the smallest entry of the *covariance matrix* $XX^\top$. One major (theoretical)

advantage of the offline case is that it enables the use of fast matrix-multiplication (FMM) to speed-up the computation of the covariance matrix [Val12, AWY14, AWY18, Alm19] (i.e., sub-linear amortized per query). By contrast, in the online setting such speedups are conjectured to be impossible [HKNS15, LW17].

Notations. For any positive integer n, we use $[n]$ to denote $\{1, 2, \cdots, n\}$. For any function f, we use $\widetilde{O}(f)$ to denote $f \cdot \mathrm{poly}(\log f)$. We use $\Pr[]$ to denote probability. We use $\mathbb{E}[]$ to denote expectation. We use both $l(\cdot)$ and $\|\cdot\|_{\mathrm{sym}}$ to denote the symmetric norm. We use $\|\cdot\|_2$ to denote the entry-wise ℓ_2 norm.

1.1 Our Results

Two important complexity measures of (symmetric) norms, which capture their "intrinsic dimensionality", are the *concentration modulus* (mc) and *maximum modulus* (mmc) parameters. We now define these quantities along the lines of [BBC+17, SWY+19].

Definition 1.2 (Modulus of concentration (mc))**.** *Let $X \in \mathbb{R}^n$ be uniformly distributed on S^{n-1}, the l_2 unit sphere. The median of a symmetric norm l is the (unique) value M_l such that $\Pr[l(X) \geq M_l] \geq 1/2$ and $\Pr[l(X) \leq M_l] \geq 1/2$. Similarly, $\mathfrak{b}_l$ denotes the maximum value of $l(x)$ over $x \in S^{n-1}$. We call the ratio*

$$\mathrm{mc}(l) := \mathfrak{b}_l / M_l$$

the modulus of concentration of the norm l.

For every $k \in [n]$, the norm l induces a norm $l^{(k)}$ on $\mathbb{R}^k$ by setting

$$l^{(k)}((x_1, x_2, \ldots, x_k)) := l((x_1, x_2, \ldots, x_k, \ldots, 0, \ldots, 0))$$

Definition 1.3 (mmc)**.** *Define the maximum modulus of concentration of the norm l as*

$$\mathrm{mmc}(l) := \max_{k \in [n]} \mathrm{mc}(l^{(k)}) = \max_{k \in [n]} \frac{\mathfrak{b}_{l^{(k)}}}{M_{l^{(k)}}}$$

Next, we present a few examples (in Table 1) for different norm l's $\mathrm{mmc}(l)$.

Norm l	$\mathrm{mmc}(l)$
$\ell_p (p \leq 2)$	$\Theta(1)$
$\ell_p (p > 2)$	$\Theta(d^{1/2-1/p})$
top-k norms	$\widetilde{\Theta}(\sqrt{d/k})$
k-support norms and the box-norm	$O(\log d)$
max-mix and sum-mix of ℓ_1 and ℓ_2	$O(1)$
Orlicz norm $\|\cdot\|_G$	$O(\sqrt{C_G \log d})$

Table 1: Examples of $\mathrm{mmc}(l)$, where max-mix of ℓ_1 and ℓ_2 is defined as $\max\{\|x\|_2, c\|x\|_1\}$ for a real number c, sum-mix of ℓ_1 and ℓ_2 is defined as $\|x\|_2 + c\|x\|_1$ for a real number c, and C_G of Orlicz norm is defined as the number that for all $0 < x < y$, $G(y)/G(x) \leq C_G (y/x)^2$.

We are now ready to state our main result:

Theorem 1.4 (Main result, informal version of Theorem D.1)**.** *Let $\|\cdot\|_l$ be any symmetric norm on $\mathbb{R}^d$. There is a data structure for the online symmetric-norm Distance Oracle problem (Definition 1.1), which uses $n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(d/\delta))$ space, supporting the following operations:*

- $\textsc{Init}(\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d, \epsilon \in (0,1), \delta \in (0,1))$*: Given n data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter ϵ and a failure probability δ as input, the data structure preprocesses in time $n(d + \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(d/\delta))$.*
- $\textsc{UpdateX}(z \in \mathbb{R}^d, i \in [n])$*: Given an update vector $z \in \mathbb{R}^d$ and index $i \in [n]$, the data data structure receives z and i as inputs, and updates the i-th data point $x_i \leftarrow z$, in $d \cdot \mathrm{poly}(1/\epsilon, \log(d/\delta))$ time.*

- QUERY$(q \in \mathbb{R}^d, \mathsf{S} \subseteq [n])$*: Given a query point* $q \in \mathbb{R}^d$ *and a subset of the input points* $\mathsf{S} \subseteq [n]$*, the* QUERY *operation outputs a* ϵ*- multiplicative approximation to each distance from* q *to points in* S*, in time*

$$(d + |\mathsf{S}| \cdot \mathrm{mmc}(l)^2) \cdot \mathrm{poly}(1/\epsilon, \log(d/\delta))$$

i.e. it provides a set of estimates $\{\mathrm{dst}_i\}_{i \in \mathsf{S}}$ *such that:*

$$\forall i \in \mathsf{S}, (1-\epsilon)\|q - x_i\|_l \leq \mathrm{dst}_i \leq (1+\epsilon)\|q - x_i\|_l$$

with probability at least $1 - \delta$*. Note that* $\mathrm{mmc}()$ *is defined as Definition 1.3*

- ESTPAIR$(i, j \in [n])$ *Given indices* $i, j \in [n]$*, the* ESTPAIR *operation takes* i *and* j *as input and approximately estimates the symmetric norm distances from* i*-th to the* j*-th point* $x_i, x_j \in \mathbb{R}^d$ *in time* $\mathrm{mmc}(l)^2 \cdot \mathrm{poly}(1/\epsilon, \log(d/\delta))$ *i.e. it provides a estimated distance* pair *such that:*

$$(1-\epsilon)\|x_i - x_j\|_l \leq \mathrm{pair} \leq (1+\epsilon)\|x_i - x_j\|_l$$

with probability at least $1 - \delta$*.*

Roadmap. In Section 2, we give an overview of techniques we mainly use in the work. Section 3 gives an introduction of the Sparse Recovery Data we use for sketching. Section 4 analyze the running time and space of our data structure with their proofs, respectively. Section 5 show the correctness of our data structure and give its proof. Finally in Section 6 we conclude our work.

2 Technique Overview

Our distance oracle follows the "sketch-and-decode" approach, which was extensively used in many other sublinear-time compressed sensing and sparse recovery problems [Pri11, HIKP12, LNNT16, NS19, SSWZ22]. The main idea is to compress the data points into smaller dimension by computing, for each data point $x_i \in \mathbb{R}^d$, a (randomized) linear sketch $\Phi \cdot x_i \in \mathbb{R}^{d'}$ with $d' \ll d$ at preprocessing time, where Φx_i is an unbiased estimator of $\|x_i\|$. At query time, given a query point $q \in \mathbb{R}^d$, we analogously compute its sketch Φq. By *linearity* of Φ, the distance between q and x_i (i.e., $\ell(q - x_i)$) can be trivially decoded from the sketch *difference* $\Phi q - \Phi x_i$. As we shall see, this simple virtue of linearity is less obvious to retain when dealing with general symmetric norms.

Layer approximation Our algorithm uses the *layer approximation* method proposed by Indyk and Woodruff in [IW05] and generalized to symmetric norms in [BBC+17, SWY+19]. Since symmetric norms are invariant under reordering of the coordinates, the main idea in [IW05] is to construct a "layer vector" as follows: for a vector $v \in \mathbb{R}^d$, round (the absolute value of) each coordinate to the nearest power α^i for some fixed $\alpha \in \mathbb{R}$ and $i \in \mathbb{N}$, and then sort the coordinates in an increasing order. This ensures that the i-th layer contains all the coordinates $j \in [d]$ satisfying: $\alpha^{i-1} < |v_j| \leq \alpha^i$. In particular, the layer vector of v has the form

$$\mathcal{L}(v) := (\underbrace{\alpha^1, \ldots, \alpha^1}_{b_1 \text{ times}}, \underbrace{\alpha^2, \ldots, \alpha^2}_{b_2 \text{ times}}, \cdots, \underbrace{\alpha^P, \ldots, \alpha^P}_{b_P \text{ times}}, 0, \ldots, 0) \in \mathbb{R}^d,$$

where b_i is the number of coordinates in layer-i. More importantly, since the norm is symmetric, the layer vector $\mathcal{L}(v)$ has a succinct representation: $(b_i)_{i \in [P]}$.

Then, it suffices to estimate b_i for each $i \in [P]$, where the Indyk-Woodruff sketching technique can be used to approximate the vector. At the i-th layer, each coordinate of v is sampled with probability P/b_i, and then the algorithm identifies the ℓ_2*-heavy-hitters* of the sampled vector. [BBC+17] gave a criterion for identifying the important layers, whose heavy-hitter coordinates in the corresponding *sampled* vectors, is enough to recover the entire symmetric norm $\|v\|_{\mathrm{sym}}$.

Unfortunately, this technique for norm estimation does not readily translate to estimating *distances* efficiently:

- *Too many layers:* In previous works, each data point x_i is sub-sampled *independently* in R layers, i.e, generates R subsets of coordinates $S_i^1, \ldots, S_i^R \subset [d]$. The sketch of the query point $S(q)$ then needs to be compared to each $S(x_i)$ in every layer. Since there are $R = \Omega(n)$ layers in [BBC+17, IW05] of size $\Omega(d)$ across all all data points, the total time complexity will be at least $\Omega(nd)$, which is the trivial query time.

- *Non-linearity:* The aforementioned sketching algorithms [BBC+17, IW05] involve nonlinear operations, and thus cannot be directly used for distance estimation.
- *Slow decoding:* The aforementioned sketches take linear time to decode the distance from the sketch, which is too slow for our application.

To overcome these challenges, we use the following ideas:

Technique I: shared randomness To reduce the number of layers, we let all the data points use the same set of layers. That is, in the initialization of our algorithm, we independently sample R subsets $S^1, \ldots, S^R$ with different probabilities. Then, for each data point x_i, we consider $(x_i)_{S^j}$ as the sub-sample for the j-th layer, and perform sketch on it. Hence, the number of different layer sets is reduced from nt to t. For a query point q, we just need to compute the sketches for $(q)_{S^1}, \ldots, (q)_{S^R}$. And the distance between q and x_i can be decoded from $\{\Phi(x_i)_{S^j} - \Phi(q)_{S^j}\}_{j \in [R]}$. We also prove that the shared randomness in all data points will not affect the correctness of layer approximation.

Technique II: linearization We choose a different sketching method called BATCHHEAVYHITTER (see Theorem D.3 for details) to generate and maintain the *linear sketches*, which allows us to decode the distance from sketch difference.

Technical III: locate-and-verify decoding We design a *locate-and-verify* style decoding method to recover distance from sketch. In our data structure, we not only store the sketch of each sub-sample vector, but also the vector itself. Then, in decoding a sketch, we can first apply the efficient sparse-recovery algorithm to identify the position of heavy-hitters. Next, we directly check the entries at those positions to verify that they are indeed "heavy" (comparing the values with some threshold), and drop the non-heavy indices. This verification step is a significant difference from the typical sparse recovery approaches, which employ complex (and time-consuming) subroutines to *reconstruct the values* of the heavy-hitter coordinates. Instead, our simple verification procedure eliminates all the *false-positive* heavy-hitters, therefore dramatically reducing the running time of the second step, which can now be performed directly by reading-off the values from the memory.

With these three techniques, we obtain our sublinear-time distance estimation algorithm. Our data structure first generate a bunch of randomly selected subsets of coordinates as the layer sets. Then, for each data point, we run the BATCHHEAVYHITTER procedure to sketch the sub-sample vector in each layer[1]. In the query phase, we call the DECODE procedure of BATCHHEAVYHITTER for the sketch differences between the query point q and each data point x_i, and obtain the heavy hitters of each layer. We then select some "important layers" and use them to approximately recover the layer vector $\mathcal{L}(q - x_i)$, which gives the estimated distance $\|q - x_i\|_{\mathrm{sym}}$.

Finally, we summarize the time and space costs of our data structure. Let ϵ be the precision parameter and δ be the failure probability parameter. Our data structure achieves $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$-time for initialization , $\widetilde{O}(d)$-time per data point update, and $\widetilde{O}(d + n \cdot \mathrm{mmc}(l)^2))$-time per query. As for space cost, our data structure uses the space of $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ in total. Note that mmc is defined as Defnition 1.3.

3 Sparse Recovery Data Structure

We design a data structure named BATCHHEAVYHITTER to generate sketches and manage them. In our design, it is a "linear sketch" data structure, and providing the following functions:

- INIT($\epsilon \in (0, 0.1), n, d$). Create a set of Random Hash functions and all the n copies of sketches share the same hash functions. This step takes $\mathcal{T}_{\mathrm{init}}(\epsilon, n, d)$ time.
- ENCODE($i \in [n], z \in \mathbb{R}^d, d$). This step takes $\mathcal{T}_{\mathrm{encode}}(d)$ encodes z into i-th sketched location and store a size $\mathcal{S}_{\mathrm{space}}$ linear sketch.
- ENCODESINGLE($i \in [n], j \in [d], z \in \mathbb{R}, d$). This step takes $\mathcal{T}_{\mathrm{encodesingle}}(d)$ updates one sparse vector $e_j z \in \mathbb{R}^d$ into i-th sketched location.

[1]The total sketch size of each data point is $\mathrm{mmc}(l)^2 \cdot \mathrm{poly} \log d$. In the ℓ_p-norm case with $p > 2$, $\mathrm{mmc}(l) = d^{1/2-1/p}$ (see Table 1) and our sketch size is $\widetilde{O}(d^{1-2/p})$, matching the lower bound of ℓ_p-sketching. When $p \in (0, 2]$, $\mathrm{mmc}(l) = \Theta(1)$ and our sketch size is $\widetilde{O}(1)$, which is also optimal.

- SUBTRACT$(i, j, l \in [n])$. Update the sketch at i-th location by j-th sketch minus l-th sketch.
- DECODE$(i \in [n], \epsilon \in (0, 0.1), d)$. This step takes $\mathcal{T}_{\mathrm{decode}}(\epsilon, d)$ such that it returns a set $L \subseteq [d]$ of size $|L| = O(\epsilon^{-2})$ containing all ϵ-heavy hitters $i \in [n]$ under ℓ_p. Here we say i is an ϵ-heavy hitter under ℓ_2 if $|x_i| \geq \epsilon \cdot \|x_{\overline{[\epsilon^{-2}]}}\|_2$ where $x_{\overline{[k]}}$ denotes the vector x with the largest k entries (in absolute value) set to zero. Note that the number of heavy hitters never exceeds $2/\epsilon^2$.

With this data structure, we are able to generate the sketches for each point, and subtract each other with its function. And one can get the output of heavy hitters of each sketch stored in it with DECODE function. More details are deferred to Section D.2.

4 Running Time and Space of Our Algorithm

Algorithm 1 Data structure for symmetric norm estimation: members, init, informal version of Algorithm 4

data structure DISTANCEONSYMMETRICNORM ▷ Theorem D.1
members
  $\{x_i\}_{i=1}^n \in \mathbb{R}^d$
  $\{S_{r,l,u}\}_{r \in [R], l \in [L], u \in [U]}$ ▷ A list of the BATCHHEAVYHITTER
  $\{H_{r,l,u}\}_{r \in [R], l \in [L], u \in [U]} \subset [d] \times \mathbb{R}$
end members

public:
procedure INIT$(\{x_1, \cdots, x_n\} \subset \mathbb{R}^d, \delta, \epsilon)$ ▷ Lemma 4.1
  Initialize the sparse-recovery data structure $\{S_{r,l,u}\}$
  Create $\{\mathrm{bmap}_{r,l,u}\}$ shared by all $i \in [n]$ ▷ $\{\mathrm{bmap}_{r,l,u}\}$ is list of a layer set map
  for $i \in [n], j \in [d]$ **do**
    if $\mathrm{bmap}_{r,u,l}[j] = 1$ **then**
      Sample $x_{i,j}$ into each subvector $\overline{x}_{r,u,l,i}$
    end if
  end for
  Encode $\{x_{r,u,l,i}\}_{i \in [n]}$ into $\{S_{r,l,u}\}$
end procedure
end data structure

Algorithm 2 Data structure for symmetric norm estimation: query, informal version of Algorithm 6

data structure DISTANCEONSYMMETRICNORM
public:
procedure QUERY$(q \in \mathbb{R}^d)$ ▷ Lemma 4.3, 5.1
  Encode q into $\{S_{r,l,u}\}$
  for $i \in [n]$ **do**
    Subtract the sketch of x_i and q, get the estimated heavy-hitters of $q - x_i$
    Decode the sketch and store returned estimation of heavy-hitters into $H_{r,l,u}$
    for $u \in [U]$ **do**
      if $H_{r,l,u}$ provide correct indices of heavy hitters **then**
        Select it as good set and store it in $H_{r,l}$
      end if
    end for
    Generate estimated layer sizes $\{c_k^i\}_{k \in [P]}$
    $\mathrm{dst}_i \leftarrow$ LAYERVETCORAPPROX$(\alpha, c_1^i, c_2^i, \ldots, c_P^i, d)$
    Reset $\{H_{r,l,u}\}$ for next distance
  end for
  return $\{\mathrm{dst}_i\}_{i \in [n]}$
end procedure
end data structure

We first analyze the running time of different procedures of our data structure DISTANCEONSYMMETRICNORM. See Algorithm 4, with the linear sketch technique, we spend the time of $\widetilde{O}(n(d + \mathrm{mmc}(l)^2))$ for preprocessing and generate the sketches stored in the data structure. When it comes a data update (Algorithm 5), we spend $\widetilde{O}(d)$ to update the sketch. And when a query comes (Algorithm 6), we spend $\widetilde{O}(d + n \cdot \mathrm{mmc}(l)^2))$ to get the output distance estimation. The lemmas of running time and their proof are shown below in this section.

Lemma 4.1 (INIT time)**.** *Given data points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$, an accuracy parameter $\epsilon > 0$, and a failure probability $\delta > 0$ as input, the procedure* INIT *(Algorithm 4) runs in time $O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{13}(d/\delta)))$.*

Proof. The INIT time includes these parts:

- Line 27 takes $O(RLU \cdot \mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d))$ to initialize sketches
- Line 32 to Line 34 takes $O(RUdL)$ to generate the bmap;
- Line 39 takes $O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))$ to generate sketches

By Theorems D.3, we have

- $\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) = n \cdot O(\beta^{-1} \log^2 d) = O(n \cdot \mathrm{mmc}(l)^2 \log^7(d) \epsilon^{-5})$,
- $\mathcal{T}_{\mathrm{encodesingle}}(d) = O(\log^2(d))$.

Adding them together we got the time of

$$\begin{aligned}
& O(RLU\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d)) + O(RUdL) + O(ndRUL \cdot \mathcal{T}_{\mathrm{encodesingle}}(d)) \\
= & O(RLU(\mathcal{T}_{\mathrm{init}}(\sqrt{\beta}, n, d) + nd \cdot \mathcal{T}_{\mathrm{encodesingle}}(d))) \\
= & O(\epsilon^{-4} \log(1/\delta) \log^4(d) \cdot \log(d) \cdot \log(d^2/\delta)(n \cdot \mathrm{mmc}(l)^2 \log^7(d) \epsilon^{-5} + nd \log^2(d))) \\
= & O(\epsilon^{-9} n(d + \mathrm{mmc}(l)^2) \log^{13}(d/\delta)))
\end{aligned}$$

where the first step follows from merging the terms, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encodesingle}}(d), \mathcal{T}_{\mathrm{init}}$, the third step follows from merging the terms.

Thus, we complete the proof. □

Lemma 4.2 (UPDATE time)**.** *Given a new data point $z \in \mathbb{R}^d$, and an index i where it should replace the original data point $x_i \in \mathbb{R}^d$. The procedure* UPDATE *(Algorithm 5) runs in time $O(\epsilon^{-4} d \log^8(d/\delta))$*

Proof. The UPDATE operation calls BATCHHEAVYHITTER.ENCODE for RLU times, so it has the time of

$$\begin{aligned}
O(RLU \cdot \mathcal{T}_{\mathrm{encode}}(d)) &= O(\epsilon^{-4} \log(1/\delta) \log^4(d) \cdot \log(d) \cdot \log(d^2/\delta) \cdot d \log^2(d)) \\
&= O(\epsilon^{-4} d \log^8(d/\delta))
\end{aligned}$$

where the first step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}(d)$, the second step follows from

$$\begin{aligned}
& \log(1/\delta) \log^4(d) \log(d) \log(d^2/\delta) \log^2(d) \\
= & (\log(1/\delta))(\log^7 d)(2 \log d + \log(1/\delta)) \\
= & O(\log^8(d/\delta))
\end{aligned}$$

Thus, we complete the proof. □

Here, we present a QUERY for outputting all the n distances. In Section E, we provide a more general version, which can take any input set $\mathsf{S} \subseteq [n]$, and output distance for only them in a shorter time that proportional to $|\mathsf{S}|$.

Lemma 4.3 (QUERY time)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* QUERY *(Algorithm 6) runs in time*

$$O(\epsilon^{-9}(d + n \cdot \mathrm{mmc}(l)^2)\log^{13}(d/\delta)).$$

Proof. The QUERY operation has the following two parts:

- **Part 1:** Line 5 takes $O(RLU \cdot \mathcal{T}_{\mathrm{encode}})$ time to call ENCODE to generate sketches for q.
- **Part 2:** For every $i \in [n]$:
  - Line 13 takes $O(RLU \cdot \mathcal{T}_{\mathrm{subtract}})$ time to compute sketch of the difference between q and x_i, and store the sketch at index of $n + 1$.
  - Line 14 takes $O(RLU \cdot \mathcal{T}_{\mathrm{decode}})$ time to decode the BATCHHEAVYHITTER and get estimated heavy hitters of $q - x_i$.
  - Line 16 to Line 24 takes $O(RLU \cdot 2/\beta)$ time to analyze the BATCHHEAVYHITTER and get the set of indices, where $2/\beta$ is the size of the set.
  - Line 30 takes $O(LP \cdot 2/\beta)$ time to compute size of the layer sets cut by α.
  - Line 32 to Line 36 takes $O(PL)$ time to compute the estimation of each layer.

  The total running time of this part is:

$$\begin{aligned} & n \cdot (O(RLU \cdot \mathcal{T}_{\mathrm{subtract}}) + O(RLU \cdot \mathcal{T}_{\mathrm{decode}}) + O(RLU \cdot 2/\beta) + O(LP \cdot 2/\beta) + O(LP)) \\ = & O(nL(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1})) \end{aligned}$$

  time in total.

Taking these two parts together we have the total running time of the QUERY procedure:

$$\begin{aligned} & O(RLU \cdot \mathcal{T}_{\mathrm{encode}}) + O(nL(RU(\mathcal{T}_{\mathrm{subtract}} + \mathcal{T}_{\mathrm{decode}} + \beta^{-1}) + P\beta^{-1})) \\ = & O(RLU(\mathcal{T}_{\mathrm{encode}} + n \cdot \mathcal{T}_{\mathrm{subtract}} + n \cdot \mathcal{T}_{\mathrm{decode}} + n\beta^{-1}) + nLP\beta^{-1}) \\ = & O(\epsilon^{-4}\log^6(d/\delta)(d\log^2(d) + n\beta^{-1}\log^2(d)) + n\epsilon^{-1}\log^2(d)\beta^{-1}) \\ = & O(\epsilon^{-9}(d + n \cdot \mathrm{mmc}(l)^2)\log^{13}(d/\delta)) \end{aligned}$$

where the first step follows from the property of big O notation, the second step follows from the definition of $R, L, U, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{encode}}, \mathcal{T}_{\mathrm{subtract}}, \mathcal{T}_{\mathrm{decode}}$ (Theorem D.3) , P, the third step follows from merging the terms.

Thus, we complete the proof.

□

Next, we analyze the space usage in our algorithm. We delay the proofs into Section G.

Lemma 4.4 (Space complexity of our data structure, informal version of Lemma G.1)**.** *Our data structure (Algorithm 4) uses $O(\epsilon^{-9}n(d + \mathrm{mmc}(l)^2)\log^{13}(d/\delta))$ bits of space.*

5 Correctness of Our Algorithm

The correctness of our distance oracle is proved in the following lemma:

Lemma 5.1 (QUERY correctness)**.** *Given a query point $q \in \mathbb{R}^d$, the procedure* QUERY *(Algorithm 6) takes q as input and approximately estimates $\{\mathrm{dst}_i\}_{i\in[n]}$ the distance between q and every x_i with the norm l, such that for every dst_i, with probability at least $1 - \delta$, we have*

$$(1 - \epsilon) \cdot \|q - x_i\|_{\mathrm{sym}} \leq \mathrm{dst}_i \leq (1 + \epsilon) \cdot \|q - x_i\|_{\mathrm{sym}}$$

Proof. Without loss of generality, we can consider a fixed $i \in [n]$. For simplicity, we denote x_i by x.

Let $v := q - x$. By Lemma C.3, it is approximated by its layer vector $\mathcal{L}(v)$, namely,

$$\|v\|_{\mathrm{sym}} \leq \|\mathcal{L}(v)\|_{\mathrm{sym}} \leq (1 + O(\epsilon))\|v\|_{\mathrm{sym}}, \tag{1}$$

where $\|\cdot\|_{\mathrm{sym}}$ is a symmetric norm, denoted also by $l(\cdot)$. We omit (v) for if its clear from text.

We assume without loss of generality that $\epsilon \geq 1/\operatorname{poly}(d)$. Our algorithm maintains a data structure that eventually produces a vector $\mathcal{J}(v)$, which is created with the layer sizes $c_1, c_2, \ldots, c_P$, where c's denotes the estimated layer sizes output by out data structure, and the b's are the ground truth layer sizes. We will show that with high probability, $\|\mathcal{J}\|_{\mathrm{sym}}$ approximates $\|\mathcal{L}\|_{\mathrm{sym}}$. Specifically, to achieve $(1 \pm \epsilon)$-approximation to $\|v\|_{\mathrm{sym}}$, we set the approximation guarantee of the layer sets (Definition C.1) to be $\epsilon_1 := O(\frac{\epsilon^2}{\log(d)})$ and the importance guarantee to be $\beta_0 := O(\frac{\epsilon^5}{\mathrm{mmc}(l)^2 \log^5(d)})$, where $\mathrm{mmc}(l)$ is defined as Definition 1.3.

Observe that the number of non-empty layer sets $P = O(\log_\alpha(d)) = O(\log(d)/\epsilon)$. Let E_{succeed} denote the event $(1-\epsilon_1)b_k \leq c_k \leq b_k$. By Lemma F.1, it happens with high probability $1-\delta$. Conditioned on this event.

Denote by $\mathcal{J}$ the vector generated with the layer sizes given by Line 35, and by $\mathcal{L}^*$ the vector $\mathcal{L}$ after removing all buckets (Definition C.2) that are not β-contributing (Definition C.5), and define $\mathcal{J}^*$ similar to $\mathcal{L}^*$, where we set $\beta := \epsilon/P = O(\epsilon^2/\log(d))$. Every β-contributing layer is necessarily β_0-important (Definition C.1) by Lemma C.7 and Lemma C.8 and therefore satisfies $c_k \geq (1-\epsilon_1)b_k$ (Lemma F.1). We bound the error of $\|\mathcal{L}^*\|_{\mathrm{sym}}$ by Lemma C.6, namely,

$$(1 - O(\epsilon))\|\mathcal{L}\|_{\mathrm{sym}} \leq (1 - O(\log_\alpha d) \cdot \beta)\|\mathcal{L}\|_{\mathrm{sym}} \leq \|\mathcal{L}^*\|_{\mathrm{sym}} \leq \|\mathcal{L}\|_{\mathrm{sym}}.$$

where the first step follows from the definition of β, the second step and the third step follow from Lemma C.6.

Then, we have

$$\begin{aligned}
\|\mathcal{J}\|_{\mathrm{sym}} &\geq \|\mathcal{J}^*\|_{\mathrm{sym}} \\
&= \|\mathcal{L}^* \backslash \mathcal{L}_{k_1} \cup \mathcal{J}_{k_1} \ldots \backslash \mathcal{L}_{k_\kappa} \cup \mathcal{J}_{k_\kappa}\|_{\mathrm{sym}} \\
&\geq (1-\epsilon_1)^P \|\mathcal{L}^*\|_{\mathrm{sym}} \\
&\geq (1 - O(\epsilon))\|\mathcal{L}^*\|_{\mathrm{sym}}. \qquad (2)
\end{aligned}$$

where the first step follows from monotonicity (Lemma B.1), the second step follows from the definition of $\mathcal{J}^*$, the third step follows from Lemma C.4, and the fourth step follows from the definition of ϵ_1 and P.

Combining Eq. (1) and (2), we have

$$(1 - O(\epsilon)) \cdot \|v\|_{\mathrm{sym}} \leq \|\mathcal{J}^*\|_{\mathrm{sym}} \leq \|v\|_{\mathrm{sym}}, \qquad (3)$$

which bounds the error of $\|\mathcal{J}^*\|_{\mathrm{sym}}$ as required. Note that, with Lemma C.6 we have

$$(1 - O(\epsilon)) \cdot \|\mathcal{J}\|_{\mathrm{sym}} \leq \|\mathcal{J}^*\|_{\mathrm{sym}} \leq \|\mathcal{J}\|_{\mathrm{sym}} \qquad (4)$$

Combining the Eq.(3) and (4) we have

$$(1 - O(\epsilon)) \cdot \|v\|_{\mathrm{sym}} \leq \|\mathcal{J}\|_{\mathrm{sym}} \leq (1 + O(\epsilon)) \cdot \|v\|_{\mathrm{sym}}$$

Note that E_{succeed} has a failure probability of δ. Thus, we complete the proof. □

6 Conclusion

Similarity search is the backbone of many large-scale applications in machine-learning, optimization, databases and computational geometry. Our work strengthens and unifies a long line of work on metric embeddings and sketching, by presenting the first Distance Oracle *for any symmetric norm*, with nearly-optimal query and update times. The generality of our data structure allows to apply it as a *black-box* for data-driven *learned* symmetric distance metrics [DKJ+07] and in various optimization problems involving symmetric distances.

Our work raises several open questions for future study:

- The efficiency of our data structure depends on $\mathrm{mmc}(l)$, the concentration property of the symmetric norm. Is this dependence necessary?
- Can we generalize our data structure to certain *asymmetric norms* ?

We believe our work is also likely to influence other fundamental problems in high-dimensional optimization and search, e.g, kernel linear regression, geometric sampling and near-neighbor search.

References

[AC09] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[AFS12] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

[Alm19] Josh Alman. An illuminating algorithm for the light bulb problem. In *2nd Symposium on Simplicity in Algorithms (SOSA)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[ALS+18] Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with orlicz norm. In *International Conference on Machine Learning (ICML)*, pages 224–233. PMLR, 2018.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

[ANN+17] Alexandr Andoni, Huy L Nguyen, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 902–913, 2017.

[ANN+18] Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 159–169. IEEE, 2018.

[AWY14] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 218–230. SIAM, 2014.

[AWY18] Josh Alman, Joshua R Wang, and Huacheng Yu. Cell-probe lower bounds from online communication complexity. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1003–1012, 2018.

[BBC+17] Jarosław Błasiok, Vladimir Braverman, Stephen R Chestnut, Robert Krauthgamer, and Lin F Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 716–729, 2017.

[Bha97] Rajendra Bhatia. Matrix analysis, volume 169 of. *Graduate texts in mathematics*, 1997.

[BYJKS04] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[CCF04] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[CLP+20] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations (ICLR)*, 2020.

[CMF+20] Beidi Chen, Tharun Medini, James Farwell, Charlie Tai, Anshumali Shrivastava, et al. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems*, 2:291–306, 2020.

[CN20] Yeshwanth Cherapanamjeri and Jelani Nelson. On adaptive distance estimation. In *Advances in Neural Information Processing Systems*, 2020.

[CS09] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

[DKJ+07] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[GMS86] M Gromov, V Milman, and G Schechtman. Asymptotic theory of finite dimensional normed spaces, volume 1200 of lectures notes in mathematics, 1986.

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[HIKP12] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 563–578, 2012.

[HKNS15] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 21–30, 2015.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[IRW17] Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Practical data-dependent metric compression with provable guarantees. *Advances in Neural Information Processing Systems*, 30, 2017.

[IW05] Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208, 2005.

[JKDG08] Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *NIPS*, volume 8, pages 761–768. Citeseer, 2008.

[JL84] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[KNPW11] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 745–754, 2011.

[LDFU13] Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems (NIPS)*, pages 369–377, 2013.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyên, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, 2016.

[LNRW19] Jerry Li, Aleksandar Nikolov, Ilya Razenshteyn, and Erik Waingarten. On mean estimation for general norms with statistical queries. In *Conference on Learning Theory (COLT)*, pages 2158–2172. PMLR, 2019.

[LSV18] Yin Tat Lee, Zhao Song, and Santosh S Vempala. Algorithmic theory of odes and sampling from well-conditioned logconcave densities. *arXiv preprint arXiv:1812.06243*, 2018.

[LW17] Kasper Green Larsen and Ryan Williams. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2182–2189. SIAM, 2017.

[MMR22] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson–lindenstrauss transform for k-means and k-medians clustering. *SIAM Journal on Computing*, (0):STOC19–269, 2022.

[MPS14] Andrew M McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. *Advances in neural information processing systems*, 27, 2014.

[NS19] Vasileios Nakos and Zhao Song. Stronger l2/l2 compressed sensing; without iterating. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 289–297, 2019.

[Pag13] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[Pel00] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.

[Pri11] Eric Price. Efficient sketches for the set query problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 41–56, 2011.

[SS02] Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 360–369, 2002.

[SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[SSWZ22] Zhao Song, Baocheng Sun, Omri Weinstein, and Ruizhe Zhang. Sparse fourier transform over lattices: A unified approach to signal reconstruction. http://arxiv.org/abs/2205.00658, 2022.

[SSX21] Anshumali Shrivastava, Zhao Song, and Zhaozhuo Xu. Sublinear least-squares value iteration via locality sensitive hashing. *arXiv preprint arXiv:2105.08285*, 2021.

[SWY+19] Zhao Song, Ruosong Wang, Lin Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. *Advances in Neural Information Processing Systems*, 32, 2019.

[SWZ19] Zhao Song, David Woodruff, and Peilin Zhong. Towards a zero-one law for column subset selection. *Advances in Neural Information Processing Systems*, 32, 2019.

[SXZ22] Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification with inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022.

[TZ12] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

[Val12] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20. IEEE, 2012.

[WP11] Oren Weimann and David Peleg. A note on exact distance labeling. *Information processing letters*, 111(14):671–673, 2011.

[XSS21] Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] , we discuss the theoretical guarantee of the accuracy-efficiency tradeoff of our algorithm in Section 4 and Section E.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] . Our theoretical work does not have explicitly negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] , we explicitly stated the assumptions.
   (b) Did you include complete proofs of all theoretical results? [Yes] , please refer to Section 4, Section 5 and supplementary materials.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]