Fast Nearest Neighbor Search for ℓ_p Metrics

Robert Krauthgamer
The Harry Weinrebe Professorial Chair of Computer Science, Weizmann Institute of Science, Rehovot, Israel

Nir Petruschka
Weizmann Institute of Science, Rehovot, Israel

Abstract

The Nearest Neighbor Search (NNS) problem asks to design a data structure that preprocesses an n-point dataset X lying in a metric space $\mathcal{M}$, so that given a query point $q \in \mathcal{M}$, one can quickly return a point of X minimizing the distance to q. The efficiency of such a data structure is evaluated primarily by the amount of space it uses and the time required to answer a query. We focus on the fast query-time regime, which is crucial for modern large-scale applications, where datasets are massive and queries must be processed online, and is often modeled by query time $\mathrm{poly}(d \log n)$ when $\mathcal{M}$ is a d-dimensional normed space. Our main result is such a randomized data structure for NNS in ℓ_p^d spaces, $p > 2$, that achieves $p^{O(1)+\log\log p}$ approximation with fast query time and $\mathrm{poly}(dn)$ space. Our data structure improves, or is incomparable to, the state-of-the-art for the fast query-time regime from [Bartal and Gottlieb, TCS 2019] and [Krauthgamer, Petruschka and Sapir, FOCS 2025].

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Nearest neighbor algorithms

Keywords and phrases Nearest neighbor search, metric embeddings, ℓ_p norm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.66

Related Version *arXiv Version*: `https://arxiv.org/abs/2603.21148`

Funding *Robert Krauthgamer*: Work partially supported by the Israel Science Foundation grant #1336/23.

Acknowledgements We thank Shay Sapir and Asaf Petruschka for helpful discussions.

1 Introduction

The Nearest Neighbor Search (NNS) problem asks to design a data structure (also called a scheme) that preprocesses an n-point dataset X lying in a metric space $\mathcal{M}$, so that given a query point $q \in \mathcal{M}$, one can quickly return a point of X minimizing the distance to q (or approximately minimizing it in the approximate version). The efficiency of such a data structure is evaluated primarily by the amount of space it uses and the time required to answer a query. The preprocessing time is a secondary measure and is usually comparable to the space usage. Because of its central role in areas such as machine learning, data analysis, and information retrieval, NNS has been the subject of extensive research, both practical and theoretical (see, e.g., the surveys [1, 6]).

It is well known that approximate NNS can be reduced to solving $\mathrm{polylog}(n)$ instances of the approximate *near* neighbor problem [18]. For this reason, we restrict attention to the latter.

▶ **Definition 1.1.** *The Approximate Near Neighbor problem for a metric space* $(\mathcal{M}, d_{\mathcal{M}})$ *and parameters* $c \geq 1$*,* $r > 0$*, abbreviated* (c, r)-ANN*, is the following. Design a data structure that preprocesses an* n*-point subset* $X \subseteq \mathcal{M}$*, so that given a query* $q \in \mathcal{M}$ *with* $d_{\mathcal{M}}(q, X) \leq r$*,*[1] *it reports* $x \in X$ *such that*

$$d_{\mathcal{M}}(q, x) \leq cr.$$

In a randomized data structure, the reported $x \in X$ *satisfies this with probability at least* $2/3$*.*

We focus on the fast query-time regime, which is crucial for modern large-scale applications where datasets are massive and queries must be processed online, and is often modeled by query time $\mathrm{poly}(d \log n)$ when $\mathcal{M}$ is a d-dimensional normed space. We study the metric spaces ℓ_p^d, which refer to $\mathbb{R}^d$ with distances defined by the ℓ_p norm, and we may omit the dimension d when it is clear from the context. In ℓ_p spaces, ANN in this regime is well understood for $1 \leq p \leq 2$ [19, 25, 18] and for $p = \infty$ [20, 4, 21]. For $2 < p < \infty$, the situation is less clear: there exists a handful of data structures, each suitable for a different range of p, as detailed in Table 1. We present a new scheme for ANN in ℓ_p, $p > 2$, with fast query time, that offers an improved tradeoff between approximation and space, as follows.

▶ **Theorem 1.2.** *Let* $p > 2$*,* $d \geq 1$*. Then for every* $r > 0$*, there is a randomized data structure for* (c, r)*-ANN in* ℓ_p^d*, where* $c = p^{O(1)+\log\log p}$*, that has query time* $\mathrm{poly}(d \log n)$ *and preprocessing (both space and time)* $\mathrm{poly}(dn)$*.*[2]

Studying ANN in ℓ_p, $p > 2$, is important both practically and theoretically. Real-world data and applications may motivate norms that emphasize outliers, e.g., for anomaly detection, or alter the presence of "hubs", e.g., to affect classification; such data structures were indeed used for time-series classification [28], see [14] for additional references. From a theoretical perspective, the geometry of ℓ_p spaces undermines existing algorithmic techniques and requires developing new ones. A key challenge is to bridge between $p = 2$ and $p = \infty$. In the fast query-time regime, this means interpolating between the classical $(O(1), r)$-ANN in ℓ_2 [18] and the $(O(\log \log d), r)$-ANN in ℓ_∞ [20], which both use only $\mathrm{poly}(dn)$ space. It is natural to conjecture that $2 < p < \infty$ exhibits an interpolation between these two guarantees, and since ℓ_∞^d is $O(1)$-equivalent to $\ell_{\log d}^d$ by Hölder's inequality, this interpolated data structure is conjectured to achieve $O(\log p)$ approximation using $\mathrm{poly}(nd)$ space. The first step towards this conjecture, in [2, 3], devised a reduction from ℓ_p to ℓ_∞, and obtained a data structure with $O(\log^{1/p} n \log \log d)$-approximation, which is mainly suited for large values of p. A nontrivial reduction, devised in [14], reduced ℓ_p to ℓ_2 and obtains $2^{O(p)}$-approximation, which is a major improvement for small values of p, although it is doubly-exponentially worse than the conjecture. A more sophisticated reduction, that was devised recently in [23], achieves approximation $\mathrm{poly}(p)$, which is an exponential improvement. However, it goes through multiple intermediate ℓ_t spaces $(2 \leq t < p)$ via a recursive argument that increases the space to $n^{O(\log p)}$, much higher than conjectured. Our Theorem 1.2 essentially completes the improvement of [23], by decreasing the space complexity back to the conjectured $\mathrm{poly}(dn)$, albeit slightly increasing the approximation to $p^{\log \log p}$. In particular, it resolves a question posed in [23], of whether the recursion can avoid this higher space complexity.

The proof of Theorem 1.2 is based on a simple yet powerful enhancement of the known NNS schemes from [14, 23], which utilizes classical results from [11, 12] about sparse covers. We provide an overview of the algorithms of [14, 23] in Section 3, along with an intuitive explanation of our approach at the beginning of Section 4.

---

1 If $d_{\mathcal{M}}(q, X) > r$, it may report anything, where as usual, $d_{\mathcal{M}}(q, X) \coloneqq \min_{x^* \in X} d_{\mathcal{M}}(x^*, q)$.

2 The time and space bounds are independent of p.

Table 1 Known data structures for ANN in ℓ_p, $p > 2$, in the fast query time regime.

Approximation	Space	Reference
$O(\log^{1/p} n \log\log d)$	$\mathrm{poly}(nd)$	[2, 3]
$2^{O((\log d)^{2/3}(\log\log d)^{1/3})}$	$\mathrm{poly}(nd)$	[8]
$2^{O(p)}$	$\mathrm{poly}(nd)$	[14]
$p^{O(1)}$	$n^{O(\log p)}$	[23]
c	$n^{O(p/c)\cdot\log\log n}$	[13]
$p^{O(1)+\log\log p}$	$\mathrm{poly}(nd)$	Theorem 1.2

Related Work

Many of the existing results on approximate nearest neighbor search in ℓ_p spaces focus on the case $1 \le p \le 2$ [25, 19, 18, 17, 5, 3, 27]. In this setting, $O(1)$-approximation can be achieved with $\mathrm{poly}(dn)$ preprocessing (space and time), and query time polynomial in $d \log n$ [25, 18].

In recent years, significant progress has been made for the case $p \in (2, \infty)$, and current results can be broadly divided into three regimes. The first regime consists of data structures that achieve moderate approximation and query time using near-linear space [3, 7, 8, 9, 24, 10]. The second regime has small approximation factor, say $O(1)$ or even $1 + \varepsilon$, in which case both the query time and the preprocessing requirements (space and time) are typically very large [14, 13]. The third regime, which is the focus of our work, has fast query time, namely, $\mathrm{poly}(d \log n)$, and existing results either achieve $2^{O(p)}$-approximation with $\mathrm{poly}(dn)$ space [14], or better approximation $o(2^p)$ at the cost of a much bigger $n^{\omega(1)}$ space [13, 23]. Our result is the first to obtain both $o(2^p)$ approximation and $\mathrm{poly}(dn)$ space. Table 1 provides a comparison of all known data structures in this regime.

We point out that the recursive embedding technique from [23], which is used here, has been applied successfully also to other problems involving ℓ_p spaces, such as the construction of Lipschitz decompositions [22, 23, 26], geometric spanners [22, 23], and low-distortion embeddings [23, 26].

2 Preliminaries

Given a metric space $\mathcal{M} = (X, d_{\mathcal{M}})$, we denote by $B_{d_{\mathcal{M}}}(x, r) \coloneqq \{y \in X : d_{\mathcal{M}}(x, y) \le r\}$ the ball of radius $r > 0$ centered at a point $x \in \mathcal{M}$.

For every $p, q \in [1, \infty)$, the Mazur map $M_{p,q} : \ell_p^d \to \ell_q^d$ is computed by taking, in each coordinate, the absolute value raised to power p/q, but keeping the original sign. Our algorithm crucially relies on the following property of this map.

▶ **Theorem 2.1** ([15, 14]). *Let $1 \le q < p < \infty$ and $C_0 > 0$, and let M be the Mazur map $M_{p,q}$ scaled down by factor $\frac{p}{q} {C_0}^{p/q-1}$. Then for all $x, y \in \ell_p^d$ such that $x, y \in B_{\ell_p}(0, C_0)$,*

$$\tfrac{q}{p}(2C_0)^{1-p/q} ||x - y||_p^{p/q} \le ||M(x) - M(y)||_q \le ||x - y||_p.$$

3 Overview of Algorithms from [14, 23]

In this section, we review the algorithms of [14] and [23] for ANN in ℓ_p, $p > 2$. For the rest of the section, fix a dataset $X \subset \ell_p^d$ with $|X| = n$ for some $p > 2$.

3.1 $(2^{O(p)}, r)$-ANN with $\mathrm{poly}(dn)$ Space [14]

In the preprocessing stage, consider a set of $k = \frac{p}{2} \cdot O(\log \log d)$ possible approximation factors $\hat{\mathcal{C}} = \{\hat{c}_i\}_{i=0}^{k}$, where $\mathrm{poly}(d) = \hat{c}_0 \geq \hat{c}_1 \geq \cdots \geq \hat{c}_k = 2^{O(p)}$. First, compute for X an initial NNS data structure A_{init} using [16], which provides approximation $\hat{c}_0 = \mathrm{poly}(d)$ using query time $\mathrm{poly}(d \log n)$ and space $\mathrm{poly}(d)\tilde{O}(n)$.[3] Then, for every data point $x \in X$ and every approximation $\hat{c} \in \hat{\mathcal{C}}$, compute a (scaled) Mazur map $M_{x,\hat{c}} : \ell_p^d \to \ell_2^d$ for the points in the set $B_{\ell_p}(0, \hat{c}r) \cap (X - x)$. Finally, compute for the image points in ℓ_2^d the $(2, r)$-ANN data structure $A_{x,\hat{c}}$ from [18], which uses $\mathrm{poly}(d \log n)$ query time and $\mathrm{poly}(dn)$ space. We have this data structure for each point $x \in X$ and each approximation factor $\hat{c} \in \hat{\mathcal{C}}$, and clearly $|B_{\ell_p}(x, \hat{c}r)| \leq n$, hence the total space requirement is $O(p \cdot \log \log d)n \cdot \mathrm{poly}(dn) = \mathrm{poly}(dn)$.

At query time, given a query point $q \in \ell_p^d$, find a $\hat{c}_0$-approximate solution x_0 using A_{init}. The crucial observation is that since the Mazur map ensures a distortion that depends on the diameter of the point set (Theorem 2.1), the answer from $A_{x_0.\hat{c}_0}$ is a $\hat{c}_1$-approximate solution x_1. Applying this procedure iteratively, the approximation factor decreases even faster than geometrically, roughly as $\hat{c}_i = \hat{c}_{i-1}^{1-2/p}$. Hence, after $k = O(p \log \log c_0)$ iterations we obtain an approximate solution x_k with $\hat{c}_k = 2^{O(p)}$, where the approximation factor does not improve further.

3.2 $(\mathrm{poly}(p), r)$-ANN with $\mathrm{poly}(d)n^{O(\log p)}$ Space [23]

In [23], the image space of the Mazur map is changed from ℓ_2 to ℓ_t for general $1 \leq t < p$. Generalizing the results from [14], a (c_t, r)-ANN data structure in ℓ_t^d with query time $Q(n)$ and space $S(n)$ is used to construct a $(2^{O(p/t)} \cdot c_t, r)$-ANN data structure for ℓ_p^d with query time $O(d) + \frac{p}{t}O(\log \log d)Q(n)$ and space $\frac{p}{t}O(\log \log d)n \cdot S(n)$. Using the above result with $t = p/2$, and applying it recursively to decrease p to 2 (which is actually a double recursion, because we also iterate over the $\hat{c}_i$'s), yields a $(\mathrm{poly}(p), r)$-ANN data structure with query time $\mathrm{poly}(d \log n)$. The caveat is that every application of the recursive step multiplies the space of the data structure by factor n, which yields a data structure with space $\mathrm{poly}(d)n^{O(\log p)}$.

4 $(2^{\tilde{O}(\log p)}, r)$-ANN with $\mathrm{poly}(dn)$ Space

In this section, we give the proof of Theorem 1.2. We first explain the intuition, and for simplicity we restrict this discussion to reducing the space requirement of [14]; reducing the space requirements of [23] is similar in spirit, although more technical.

Revisiting Section 3.1, the final solution x_k is obtained by finding iteratively a sequence of intermediate solutions $x_0, x_1, \ldots, x_{k-1}$. Each iteration $i < k$ makes progress by finding a point x_i and restricting the search region to $B_{\ell_p}(x_i, \hat{c}_i r)$, which has bounded diameter, and thus applying a Mazur map on this region has distortion guarantees. It follows that querying the data structure $A_{x_i,\hat{c}_i}$ (computed over $B_{\ell_p}(x_i, \hat{c}_i r) \cap X$) finds a point x_{i+1} and we can restrict the search region even further, to diameter $\hat{c}_{i+1}r$.

The preprocessing phase prepares for the possibility that each point $x \in X$ will serve (at query time) the $\hat{c}_i$-approximate solution, i.e., the search region will be restricted to $B_{\ell_p}(x, \hat{c}_i r)$. To make progress and restrict the search region even further, a data structure $A_{x,\hat{c}_i}$ is constructed for (the points in) this region. Our key idea in Theorem 1.2 is that, rather

[3] Throughout, the notation $\tilde{O}(f)$ hides factors that are logarithmic in f.

than preparing a *separate* data structure for each search region, the algorithm constructs one *global* collection of data structures that together cover all the possible search regions. For every $\hat{c}_i$, the algorithm constructs a set of ANN data structures computed on a collection of subsets $\mathcal{S} \subseteq 2^X$, such that for every point $x \in X$ there is some $S \in \mathcal{S}$ that contains the search region $B_{\ell_p}(x, \hat{c}_i r) \cap X$. In addition, every $S \in \mathcal{S}$ has diameter at most $\beta \hat{c}_i r$ for some $\beta > 1$. We also want the total number of points in $\mathcal{S}$ (counting repetitions) to be small. The preprocessing algorithm simply stores for every $x \in X$ a reference to a set $S_x \in \mathcal{S}$ with $B_{\ell_p}(x, \hat{c}_i r) \cap X \subseteq S_x$, and at query time, if x serves as a $\hat{c}_i$-approximate solution, the algorithm queries the ANN data structure constructed for S_x. Since S_x has a diameter at most $\beta \hat{c}_i r$, this will still cause the search region's diameter to shrink in the next iteration (although by a slightly smaller factor). Since the total number of points in $\mathcal{S}$ is small, the total memory used by all the ANN data structures will be small too.

It remains to show that the preprocessing phase can indeed find efficiently a collection of subsets of X with the above properties. Fortunately, this was shown to be possible in [11, 12], and has become a fundamental algorithmic tool with numerous applications in distributed computing, network design, routing, graph algorithms, and metric embeddings.

▶ **Definition 4.1** (Sparse Neighborhood Cover [11, 12]). *A* (β, r)-sparse cover *of a metric space* $\mathcal{M} = (X, d_{\mathcal{M}})$ *is a collection of subsets (called clusters)* $\mathcal{S} \subseteq 2^X$*, each of diameter at most* βr*, such that for every* $x \in X$ *there exists* $S \in \mathcal{S}$ *with* $B_{d_{\mathcal{M}}}(x, r) \subseteq S$*. The total number of points* $\sum_{S \in \mathcal{S}} |S|$ *is called the* sparsity *of* $\mathcal{S}$*.*

▶ **Theorem 4.2** ([12]). *There is an algorithm that, given an* n*-point metric space* $\mathcal{M}$ *and parameters* $\beta > 1$ *and* $r > 0$*, outputs a* (β, r)*-sparse cover of* $\mathcal{M}$ *of sparsity* $O(n^{1+1/\beta})$*, and runs in* $O(\beta n^{2+2/\beta})$ *time.*

We are now ready to prove Theorem 1.2, largely following the proof structure of [23, Theorem 1.8].

Proof of Theorem 1.2. Let $X \subset \ell_p^d$ be an n-point dataset for some $p \in (2, \infty)$. For clarity of exposition, we assume that p is a power of 2, which can be easily resolved, see [23, Theorem 1.2]. Also, by an application of Hölder's inequality, we may assume that $p \leq \log d$.

We construct the ANN scheme using a doubly-recursive procedure. The first recursion assumes access to an ANN scheme for ℓ_p^d that achieves approximation factor c_{base}, and provides a new ANN scheme for ℓ_p^d that achieves improved (smaller) approximation c_{new}. This step crucially relies on access to yet another ANN data structure, for ℓ_t^d, for $t = p/2$, that is actually constructed by the same method. This leads to a second recursion, of constructing ANN data structures for intermediate spaces $\ell_p^d, \ell_{p/2}^d, \ldots, \ell_2^d$, where the space ℓ_2^d is known to have ANN data structures with $O(1)$ approximation.

We next describe the first recursion, i.e., how to construct an improved (c_{new}, r)-ANN scheme for ℓ_p^d given a (c_{base}, r)-ANN scheme for ℓ_p^d and a (c_t, r)-ANN scheme for ℓ_t^d, where $t < p$. In the preprocessing phase, use Theorem 4.2 to construct for X a $(\beta, 2c_{\text{base}} r)$-cover $\mathcal{S}$ with sparsity $\tilde{O}(n^{1+1/\beta})$ for $\beta = \log p$. During the construction of $\mathcal{S}$, store for every $x \in X$ a reference to a set $S_x \in \mathcal{S}$ that "covers" it, i.e., $B_{\ell_p}(x, 2c_{\text{base}} r) \cap X \subseteq S_x$, which is guaranteed to exist in a sparse cover. In addition, for every $S \in \mathcal{S}$ designate (arbitrarily) a center point $y \in S$, apply a Mazur map $M^y : \ell_p^d \to \ell_t^d$ scaled down by factor $\frac{p}{t} \cdot (2\beta c_{\text{base}} r)^{p/t-1}$ on $B_{\ell_p}(0, 2\beta c_{\text{base}} r) \cap (X - y)$, and construct for these image points a (c_t, r)-ANN scheme A_S. Finally, construct a (c_{base}, r)-ANN scheme A_{base} for X, and amplify the success probabilities of both data structures to 5/6 by the standard method of independent repetitions. Given a query $q \in \ell_p^d$ that is guaranteed to have $x^* \in X$ with $\|x^* - q\|_p \leq r$, query A_{base}

for the point q and obtain an answer $x_{\text{base}} \in X$. Then find its cluster $S_{x_{\text{base}}}$ and this cluster's designated center y, query $A_{S_{x_{\text{base}}}}$ for the point $M^y(q-y) \in \ell_t^d$, and use its answer $M^y(z_{\text{out}} - y) \in M^y(X-y)$ to output the corresponding $z_{\text{out}} \in X$.

The next claim is analogous to [23, Claim 4.1], and the main difference is using the sparse cover.

▷ Claim 4.3. With probability at least 2/3, we have $\|z_{\text{out}} - q\|_p \le c_{\text{new}} r$, where $c_{\text{new}} = (\frac{p}{t})^{t/p}\ c_t^{t/p}\ (4\beta c_{\text{base}})^{1-t/p}$.

Proof. With probability at least $\frac{5}{6}$, the data structure A_{base} outputs a point x_{base} with $\|x_{\text{base}} - q\|_p \le c_{\text{base}} r$. Let $S_{x_{\text{base}}}$ be the set in the cover referenced by x_{base}, and let $y \in S_{x_{\text{base}}}$ be its designated center point. Since

$$\|x^* - x_{\text{base}}\|_p \le \|x^* - q\|_p + \|q - x_{\text{base}}\|_p \le 2c_{\text{base}} r,$$

we get that $x^* \in B_{\ell_p}(x_{\text{base}}, 2c_{\text{base}} r) \cap X \subseteq S_{x_{\text{base}}}$. Observe that by Theorem 2.1, $\|M^y(x^*) - M^y(q)\|_t \le r$, and thus with probability at least $\frac{5}{6}$, querying $A_{S_{x_{\text{base}}}}$ finds a point $M^y(z_{\text{out}}) \in M^y(S_{x_{\text{base}}}) \subseteq M^y(X)$ with $\|M^y(z_{\text{out}}) - M^y(q)\|_t \le c_t r$. Applying a union bound, we see that with probability at least 2/3, both events hold. In this case, we have by Theorem 2.1, that

$$\frac{t}{p} \cdot (4\beta c_{\text{base}} r)^{1-p/t} \cdot \|z_{\text{out}} - q\|_p^{p/t} \le \|M^y(z_{\text{out}}) - M^y(q)\|_t \le c_t r,$$

and by rearranging, we obtain $\|z_{\text{out}} - q\|_p \le (\frac{p}{t})^{t/p}\ c_t^{t/p}\ (4\beta c_{\text{base}})^{1-t/p} r = c_{\text{new}} r$. ◁

Denote by $\hat{c}_i$ the approximation of the ANN scheme obtained by i applications of Claim 4.3, where the initial ANN scheme is the one from [16], with approximation $\hat{c}_0 = \text{poly}(d)$. We also denote by c_t the approximation of an ANN scheme for ℓ_t, that is constructed by the same method (i.e., recursively), except that for ℓ_2^d we use a $(2,r)$-ANN scheme from [18] with $\text{poly}(d \log n)$ query time and $\text{poly}(dn)$ space and preprocessing time. Using Claim 4.3 with $t = p/2$, and furthermore applying this recursively $k = \lceil \log(\log \hat{c}_0) \rceil = O(\log \log d)$ times, we obtain

$$\hat{c}_k \le \sqrt{8\beta c_{p/2} \hat{c}_{k-1}} \le \sqrt{8\beta c_{p/2} \sqrt{8\beta c_{p/2} \hat{c}_{k-2}}} \le \ldots \le 8\beta c_{p/2} \cdot \hat{c}_0^{1/2^k} \le 16\beta c_{p/2},$$

i.e., this scheme has approximation $c_p = \hat{c}_k \le 16\beta c_{p/2}$. We can amplify the success probability of this scheme to $1 - \frac{1}{3 \log p}$ by the standard method of $O(\log \log p) = O(\log \log \log d)$ independent repetitions. Now by recursion over p for $\log p$ levels, we get that

$$c_p \le (16\beta)^{\log p} = (16 \log p)^{\log p} = p^{4 + \log \log p},$$

and the overall success probability is at least $\frac{2}{3}$ by a union bound.

We are left to analyze the query time of the algorithm, and its space and preprocessing time. Each level of the second recursion makes a total of $k \cdot O(\log \log p) = \tilde{O}(\log \log d)$ calls to an ANN scheme for ℓ_t, for different intermediate values of t. Since the $(2,r)$-ANN for ℓ_2 from [18] has query time $\text{poly}(d \log n)$, and recalling that $p \le \log d$, the overall query time is $\tilde{O}(\log \log d)^{\log p} \cdot \text{poly}(d \log n) = \text{poly}(d \log n)$.

To analyze the space and preprocessing time, we prove the following claim.

▷ Claim 4.4. There exists an absolute constant $D > 2$ such that when the data structure for ℓ_t^d, $t = 2^i$, is computed on m points, it uses total space and preprocessing time $\tilde{O}(\log \log d)^i\ \text{poly}(d) O(m)^{D + i/\log p}$.

Proof. We only analyze the space usage of the data structure; the analysis of the preprocessing time follows similarly, as it takes $O(d)$ time to compute a Mazur map and $O(\log p \cdot m^{2(1+1/\log p)}) \leq O(\log\log d \cdot m^{2(1+1/\log p)})$ time to compute a sparse cover.

The proof proceeds by induction on $i \geq 0$. For $i = 0$, the claim follows because the $(2, r)$-ANN from [18], when computed on m points, uses at most $\mathrm{poly}(d)m^D$ space for some absolute constant $D > 1$. Now, assume the claim holds for $i - 1 \geq 0$. The ANN scheme at level i of the recursion consists of two types of ANN schemes. The first type is an ANN scheme from [16] computed on all m points, which uses $\mathrm{poly}(d)\tilde{O}(m) \leq \mathrm{poly}(d)O(m)^{D(1+1/\log p)}$ space. The second type are multiple ANN schemes at level $i - 1$ that are computed on different subsets of the m points. For every $j = 0, 1, \dots, k = \tilde{O}(\log\log d)$, let $\mathcal{S}^j$ be the sparse cover of sparsity $O(m^{1+1/\log p})$ computed for the points at the j-th level of the first recursion. For every level j of the first recursion and cluster $S \in \mathcal{S}^j$, the algorithm computes a data structure of level $i - 1$ on S. By the induction hypothesis, the space of this data structure is bounded by $\tilde{O}(\log\log d)^i \,\mathrm{poly}(d)O(m)^{D+i/\log p}$. Observe that the function $f : x \mapsto \tilde{O}(\log\log d)^i \,\mathrm{poly}(d)x^{D+i/\log p}$ satisfies that $f(a) + f(b) \leq f(a + b)$ for all $a, b \geq 1$, and, since every data structure is constructed over some $S \in \mathcal{S}^j$ containing at most m points, we can bound the contribution of every point $p \in S$ to the total space usage of the data structure by $\frac{f(m)}{m}$. Since for every j the sparsity of $\mathcal{S}^j$ is at most $O(m^{1+1/\log p})$, the total space usage of all ANN schemes computed over all clusters of $\mathcal{S}^j$ is at most

$$O(m^{1+1/\log p})\frac{f(m)}{m} \leq \tilde{O}(\log\log d)^i \,\mathrm{poly}(d)O(m)^{D+(i+1)/\log p}.$$

Hence, the total space usage is

$$\begin{aligned}&\tilde{O}(\log\log d) \cdot \left(\tilde{O}(\log\log d)^i \cdot \mathrm{poly}(d) \cdot O(m)^{D+(i+1)/\log p}\right) + \mathrm{poly}(d) \cdot \tilde{O}(m)\\ &\leq \tilde{O}(\log\log d)^{i+1} \cdot \mathrm{poly}(d) \cdot O(m)^{D+(i+1)/\log p},\end{aligned}$$

completing the proof. ◁

Finally, we use Claim 4.4 for $m = n$ and $i = \log p$, and obtain that the space usage of the entire ANN data structure is bounded by

$$\tilde{O}(\log\log d)^{\log p} \,\mathrm{poly}(d) \cdot n^{D+\log p/\log p} \leq \mathrm{poly}(d) \cdot n^{D+1} \leq \mathrm{poly}(dn),$$

which completes the proof of Theorem 1.2. ◀

▶ Remark 4.5. Modifying the parameter β of the sparse cover in the proof of Theorem 1.2 from $\log p$ to $\frac{\log p}{\delta}$ for $0 < \delta < 1$ yields a data structure with a slightly larger approximation factor $p^{O(1)+\log(1/\delta)+\log\log p}$, but space requirement that matches that of [18] up to subpolynomial factors in d and an additional n^δ term.

▶ Remark 4.6. The same technique used in the proof of Theorem 1.2, namely applying Theorem 4.2 to construct covers in the preprocessing phase, can also be used to improve the space requirement of ANN for general normed spaces from [8, Theorem 3]. More specifically, for every $0 < \delta < 1$, one can shave an $\Omega(n^{1-\delta})$ factor from the space of their data structure, at the cost of an additional $O(\delta^{-1})$ factor in the approximation.

References

1 A. Andoni and P. Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*, chapter 43, pages 1135–1150. CRC Press, 3rd edition, 2017. doi:10.1201/9781315119601.

2 A. Andoni, P. Indyk, and R. Krauthgamer. Overcoming the l_1 non-embeddability barrier: algorithms for product metrics. In *19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 865–874. SIAM, 2009. doi:10.1137/1.9781611973068.94.

3 Alexandr Andoni. *NN search: the old, the new, and the impossible.* PhD thesis, Massachusetts Institute of Technology, 2009. URL: https://hdl.handle.net/1721.1/55090.

4 Alexandr Andoni, Dorian Croitoru, and Mihai Pătraşcu. Hardness of nearest neighbor under l-infinity. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 424–433. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.89.

5 Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468. IEEE, 2006. doi:10.1109/FOCS.2006.49.

6 Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3287–3318, 2019. doi:10.1142/9789813272880_0182.

7 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Data-dependent hashing via nonlinear spectral gaps. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 787–800. ACM, 2018. doi:10.1145/3188745.3188846.

8 Alexandr Andoni, Assaf Naor, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, pages 159–169. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00024.

9 Alexandr Andoni, Aleksandar Nikolov, Ilya P. Razenshteyn, and Erik Waingarten. Approximate nearest neighbors beyond space partitions. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 1171–1190. SIAM, 2021. doi:10.1137/1.9781611976465.72.

10 Alexandr Andoni and Negev Shekel Nosatzki. Embeddings into similarity measures for nearest neighbor search. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 864–893. IEEE, 2025. doi:10.1109/FOCS63196.2025.00045.

11 B. Awerbuch and D. Peleg. Sparse partitions. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990. doi:10.1109/FSCS.1990.89571.

12 Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing*, 28(1):263–277, 1998. doi:10.1137/S0097539794271898.

13 Yiqiao Bao, Anubhav Baweja, Nicolas Menand, Erik Waingarten, Nathan White, and Tian Zhang. Average distortion sketching. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 277–302. IEEE, 2025. doi:10.1109/FOCS63196.2025.00019.

14 Yair Bartal and Lee-Ad Gottlieb. Approximate nearest neighbor search for ℓ_p-spaces ($2 < p < \infty$) via embeddings. *Theoretical Computer Science*, 757:27–35, 2019. doi:10.1016/j.tcs.2018.07.011.

15 Yoav Benyamini and Joram Lindenstrauss. *Geometric nonlinear functional analysis*, volume 48. American Mathematical Soc., 1998.

16 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discret. Comput. Geom.*, 20(3):359–373, 1998. doi:10.1007/PL00009390.

17 Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *20th annual symposium on Computational geometry*, pages 253–262. ACM, 2004. doi:10.1145/997817.997857.

18 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Comput.*, 8(1):321–350, 2012. doi:10.4086/TOC.2012.V008A014.

19 P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998. `doi:10.1145/276698.276876`.

20 Piotr Indyk. On approximate nearest neighbors under ℓ_∞ norm. *J. Comput. Syst. Sci.*, 63(4):627–638, 2001. `doi:10.1006/JCSS.2001.1781`.

21 Michael Kapralov and Rina Panigrahy. NNS lower bounds via metric expansion for ℓ_∞ and EMD. In *39th International Colloquium on Automata, Languages and Programming, ICALP 2012*, volume 7391 of *Lecture Notes in Computer Science*, pages 545–556. Springer, 2012. `doi:10.1007/978-3-642-31594-7_46`.

22 Robert Krauthgamer and Nir Petruschka. Lipschitz Decompositions of Finite ℓ_p Metrics. In *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66:1–66:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. `doi:10.4230/LIPIcs.SoCG.2025.66`.

23 Robert Krauthgamer, Nir Petruschka, and Shay Sapir. The power of recursive embeddings for ℓ_p metrics. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 2546–2554. IEEE, 2025. `doi:10.1109/FOCS63196.2025.00132`.

24 Deepanshu Kush, Aleksandar Nikolov, and Haohua Tang. Near neighbor search via efficient average distortion embeddings. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 50:1–50:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.SOCG.2021.50`.

25 E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. `doi:10.1137/S0097539798347177`.

26 Assaf Naor and Kevin Ren. Optimal randomized clustering for subsets of L_p when $p > 2$. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 984–995. SIAM, 2026. `doi:10.1137/1.9781611978971.40`.

27 Huy L. Nguyen. Approximate nearest neighbor search in ℓ_p. *CoRR*, abs/1306.3601, 2013. `arXiv:1306.3601`.

28 Daren Yu, Xiao Yu, and Anqi Wu. Making the nearest neighbor meaningful for time series classification. *Proceedings of the 4th International Congress on Image and Signal Processing, CISP 2011*, 5:2481–2485, October 2011. `doi:10.1109/CISP.2011.6100672`.