INFINITY SEARCH: APPROXIMATE VECTOR SEARCH WITH PROJECTIONS ON Q-METRIC SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

An ultrametric space or infinity-metric space is defined by a dissimilarity function that satisfies a strong triangle inequality in which every side of a triangle is not larger than the larger of the other two. We show that search in ultrametric spaces with a vantage point tree has worst-case complexity equal to the depth of the tree. Since datasets of interest are not ultrametric in general, we employ a projection operator that transforms an arbitrary dissimilarity function into an ultrametric space while preserving nearest neighbors. We further learn an approximation of this projection operator to efficiently compute ultrametric distances between query points and points in the dataset. We proceed to solve a more general problem in which we consider projections in q-metric spaces – in which triangle sides raised to the power of q are smaller than the sum of the q-powers of the other two. Notice that the use of learned approximations of projected q-metric distances renders the search pipeline approximate. We show in experiments that increasing values of q result in faster search but lower recall. Overall, search in q-metric and infinity metric spaces is competitive with existing search methods.

1 INTRODUCTION

Given a dataset of vector embeddings, a dissimilarity function and a query, nearest neighbor search refers to the problem of finding the point in the dataset that is most similar to the query, or, in its stead, a vector that is among the most similar (Indyk and Motwani, 1998a). Exact search requires comparing against all points in the dataset in the worst case (Murtagh and Contreras, 2012) but it is well known that this search complexity can be reduced in the particular case of metric dissimilarities. In this case we can organize data in a vantage point (VP) tree that we can search with a smaller average number of comparisons (Yianilos, 1993). This happens because we can use triangle inequalities to prune *some* branches of the VP tree (Yianilos, 1993).

In this paper we observe that metric spaces are a particular case of what we term here as q-metric spaces. These spaces satisfy more restrictive triangle inequalities in which the q-th power of each side of a triangle is not larger than the sum of the qth powers of the other two. The advantage of q-metric spaces is that the inequalities that make pruning of metric trees in standard metric spaces possible (Section 2.1) become more lenient (Section 2.2). This fact substantiates the first contribution of our paper:

(C1) We demonstrate empirically that search in a q-metric space requires a smaller number of comparisons than search in a standard metric space (Section 5).

The limit of growing q yields ∞-metric spaces in which each side of a triangle is not larger than the maximum of the other two (Section 2). The value of ∞-metric spaces is that the conditions that allow pruning of VP trees are always met (Section 2.3). This fact leads to our second contribution:

(C2) The number of comparisons needed to find a nearest neighbor of a query in an ∞-metric space is, at most, the depth of the VP tree (Theorem 1). We further demonstrate empirically that, as it would be expected from this observation, search in ∞-metric spaces requires a number of comparisons close to the base-2 logarithm of the dataset's cardinality (Section 2.3).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We point out that ∞-metric spaces are often called ultrametric spaces and that the ∞-triangle inequality is often called the strong triangle inequality (Dovgoshey, 2025). Ultrametric spaces are equivalent to the dendrograms used in hierarchical clustering (Draganov et al., 2025), a fact that may help to understand why search in ultrametric spaces exhibits complexity close to logarithmic.

Because of (C1) we prefer to solve search problems in q-metric spaces with larger q. Because of (C2) we most prefer to solve search in ∞-metric spaces, thus the title of this paper. However, the data is what the data is and most problems in vector search involve dissimilarity functions that are not even metric (Zezula et al., 2006). We thus seek projection operators to map datasets into general q-metric or ∞-metric spaces. We propose to adopt the *canonical projections* developed first for hierarchical clustering (Carlsson et al., 2014; Smith et al., 2016; Carlsson et al., 2017) and generalized later to metric representations of network data (Segarra et al., 2019; 2020).

These canonical projections generate q-metric spaces by computing shortest q-norm paths linking any pair of points (Section 3). In particular, they generate ∞-metric spaces by computing shortest ∞-norm paths [1]. These projections are deemed canonical because they are the only ones that abide to the (very justifiable) Axioms of Projection and Transformation (Segarra et al., 2019). The Axiom of Projection states that the projection of a q-metric space should be the same q-metric space. The Axiom of Transformation states that the projection of a q-metric space must respect the partial ordering of original spaces. I.e., if the dissimilarities of a dataset dominate *all* the dissimilarities of another, the q-metric distances of the projected datasets must satisfy the same relationship. To strengthen the case for using canonical projections, we provide a third contribution:

(C3) The canonical projections respect the nearest neighbors of any query (Section 3.1): A point that is a nearest neighbor of a query with respect to the original dissimilarity function is also a nearest neighbor with respect to its canonical q-metric or ∞-metric projections.

To the extent that the Axioms of Projection and Transformation are reasonable, the canonical projections are the only reasonable methods that we can use to generate a q-metric space Segarra et al. (2019). Regardless, (C3) states that the canonical projections respect nearest neighbors, which is all we need for search.

Contributions (C1)-(C3) dictate that we can project datasets into q-metric and ∞-metric spaces with a canonical projection to search with low complexity. The hitch is that canonical projections require computation of q-shortest paths or ∞-shortest paths. This can be precomputed for the data but doing so for a query requires that we first compute distances between the query and *all* the points in the dataset. This defeats the purpose of using q-metric or ∞-metric distances to reduce search complexity. We work around this problem with our fourth contribution:

(C4) We learn a map from original embeddings to auxiliary embeddings whose Euclidean distances estimate q-metric or ∞-metric distances. We train this map on the dataset and generalize it to queries (Section 4).

Combining (C1)-(C4) we propose to use canonical q-metric and ∞-metric projections of a dataset to generate spaces where we expect nearest neighbor search to be more efficient. We incur this cost once and reutilize it for all subsequent searches. We then use the learned embedding in (C4) to estimate q-metric distances between queries and points in the dataset. The use of approximate computation of q-metric and ∞-metric projections renders the resulting search algorithm approximate. Numerical experiments indicate that this approach is competitive with state-of-the-art approximate search algorithms in terms of search complexity and recall (Section 5). We refer the reader to Appendix A for a discussion of related work.

2 Nearest Neighbor Search in q-Metric Spaces

We are given a set X containing m data vectors $x \in \mathbb{R}^n$ along with a nonnegative dissimilarity function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ so that smaller values of $d(x, y)$ represent more similarity between data vectors x and y. The set X along with the set D containing all dissimilarities $d(x, y)$ for all

[1] We emphasize that the q-norm of a path is unrelated to the q-norm of the vector embeddings. As all norms do, the q-norm of the vector embeddings satisfies the *standard* triangle inequality.

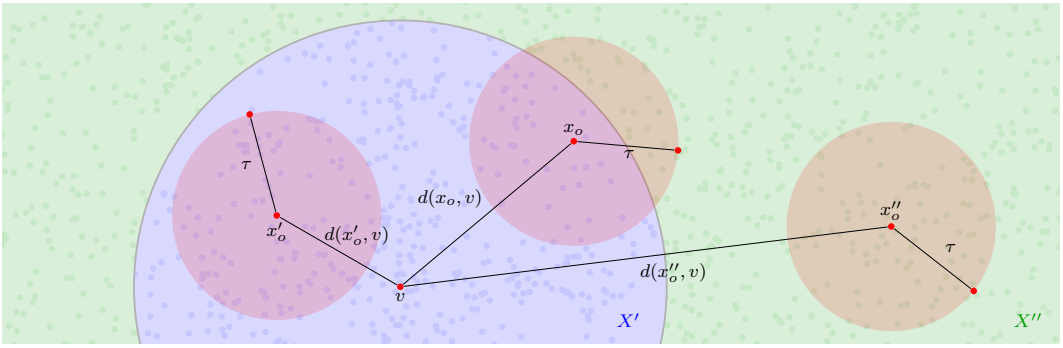


Figure 1: Search in metric spaces requires fewer comparisons than search in arbitrary spaces because for *some* queries – such as x'_o and x''_o – the triangle inequality allows us to restrict comparisons to subsets of the dataset X. Queries, such as x_o, for which triangle inequality bounds are inconclusive, also exist (6). This latter eventuality is impossible when the strong triangle inequality (3) holds, leading to a more efficient search in ultrametric spaces (Theorem 1).

points $x, y \in X$ defines a fully connected weighted graph $G = (X, D)$. We assume here that $d(x, y) = d(y, x)$ for all $x, y \in \mathbb{R}^n$, which, in particular, implies that the graph G is symmetric. Examples of dissimilarity functions used in vector search are the Euclidean distance, Manhattan (1-norm) distance, cosine dissimilarity, and dissimilarities based on the Jaccard index (Zezula et al., 2006); see F.4.

We are further given a vector x_o which is not necessarily an element of X but for which it is possible to evaluate the dissimilarity function $d(x_o, x)$ for all $x \in X$. The nearest neighbors of x_o in the set X are the elements that are most similar to x_o,

$$\hat{x}_o \equiv \underset{x \in X}{\operatorname{argmin}}\, d(x, x_o). \tag{1}$$

The problem of finding *a* vector $\hat{x}_o$ is termed nearest neighbor search, the vector x_o is called a query and the vector $\hat{x}_o$ is the query's answer (Wang et al., 2014). It may be that there are several nearest neighbors. In such case we overload $\hat{x}_o$ to denote an arbitrary nearest neighbor and also the set of all nearest neighbors. We make the distinction clear when needed.

We evaluate the performance of a search algorithm by the number $c(x_o)$ of comparisons against elements of X that are needed to find $\hat{x}_o$. Without further assumptions on the dissimilarity function d, we have $c(x_o) = m$ because we need to compare the query x_o to all points in X. This number of comparisons can be reduced if we assume that d has some metric structure (Murtagh and Contreras, 2012). For instance, it may be that d is a proper metric or pseudometric that satisfies the triangle inequality so that for any three points $x, y, z \in \mathbb{R}^n$ we have that

$$d(x, y) \leq d(x, z) + d(y, z). \tag{2}$$

Alternatively, we may consider ultrametric dissimilarity functions d (Simovici et al., 2017). In this case, triplets of points $x, y, z \in \mathbb{R}^n$ satisfy the strong triangle inequality,

$$d(x, y) \leq \max\Big[\, d(x, z),\, d(y, z) \,\Big]. \tag{3}$$

In this paper, we also have an interest in q-metric spaces (Greenhoe, 2016). These spaces arise when the dissimilarity function is such that any three points $x, y, z \in \mathbb{R}^n$ satisfy the q-triangle inequality[2],

$$d^q(x, y) \leq d^q(x, z) + d^q(y, z), \tag{4}$$

for some given $q \geq 1$. For $q = 1$, (4) reduces to (2) and as $q \to \infty$, (4) approaches (3). Thus, we can think of q-metric spaces as interpolations between regular metric spaces that sastify the standard triangle inequality and ultrametric spaces that satisfy the strong triangle inequality. Henceforth, we may refer to (2) as the 1-triangle inequality and to (3) as the ∞-triangle inequality.

[2]Notice that the q-norm $||x||_q$ for $x \in \mathbb{R}^n$ satisfies the regular triangle inequality, *not* the q-triangle inequality.

2.1 VANTAGE POINT TREES AND METRIC STRUCTURE

A classical structure that exploits metric structure to simplify search is a vantage point (VP) tree Yianilos (1993). As we show in Figure 1, a VP tree is constructed from the partition of the set X into a vantage point v and points other than v that are closer or farther apart from v than a threshold μ

$$X' := \Big\{ x \in X \,:\, d(v,x) < \mu \,,\, x \neq v \Big\}, \quad X'' := \Big\{ x \in X \,:\, \mu \leq d(v,x) \Big\}. \tag{5}$$

A VP tree is built by recursive partition of the inside set X' and the outside set X'' using corresponding vantage points and radiuses. I.e., we select a vantage point $v' \in X'$ and a radius μ' to divide X' into inside and outside sets and do the same for X''. We then partition the new group of inside and outside sets and repeat until we reach indivisible sets with only one point; see Appendices C and D for details.

When coupled with metric structure, VP trees may reduce the number of comparisons $c(x_o)$ needed to find a nearest neighbor. This is because comparing the query x_o to a vantage point v allows us to guarantee that to find a nearest neighbor it suffices to explore either the inside or outside set. Indeed, suppose that an upper bound τ on the distance between query and nearest neighbor is available and consider for the query x'_o in Figure 1. It is important for forthcoming discussions that $\tau' \leq d(v, x'_o)$ because we can otherwise set $\tau' = d(v, x'_o)$. If it happens that x'_o, τ and μ are such that

$$d(v, x'_o) + \tau < \mu, \tag{CI}$$

there is no point in the outside set X'' with a distance to x'_o smaller than τ. Thus, we can discard the outside set X'' and search only the inside set X' – see Proposition 3 for a formal proof. Likewise, the query x''_o and the nearest neighbor distance upper bound τ in Figure 1 are such that

$$\mu + \tau \leq d(v, x''_o). \tag{CO}$$

If this condition holds we know that there is no point in the inside set X' with a distance to x'_o smaller than τ. Thus, we can discard the inside set X' and search only the outside set X'' – see Proposition 3 for a formal proof. While this argument seems to indicate logarithmic complexity of nearest neighbor search in metric spaces, this is not quite so. The reason is that we may have queries such as x_o in Figure 1 for which neither condition is true. That is, there may be points x_o such that,

$$\mu - \tau \;\leq\; d(v, x_o) \;<\; \mu + \tau, \tag{6}$$

If $d(x_o, v)$ is such that (6) holds, the nearest neighbor $\hat{x}_o$ may be in X' or X'' and we therefore do not have a reduction in the number of comparisons needed to find $\hat{x}_o$.

2.2 VANTAGE POINT TREES AND q-METRIC STRUCTURE

The argument we build in Figure 1 can be generalized to q-triangle inequality comparisons in q-metric spaces. I.e., we consider inequalities analogous to those in (CI) and (CO) in which respective quantities are raised to the power of q,

$$d^q(v, x'_o) + \tau^q < \mu^q \quad (q\text{-CI}) \qquad\qquad \mu^q + \tau^q \leq d^q(v, x''_o) \quad (q\text{-CO})$$

We show in Proposition 3 in Appendix C.2 that if we are searching in a q-metric space we can prune branches of VP trees in a similar way to how we prune branches in metric spaces. If Condition (q-CI) holds, we can discard the outside set X'' and focus the search in the inside set X'. If Condition (q-CO) holds, we can discard the inside set X' and focus the search in the outside set X''.

As is the case of metric spaces, it is not always possible to prune VP trees using conditions (q-CI) and (q-CO). There may be queries x_o for which

$$\mu^q - \tau^q \;\leq\; d^q(v, x_o) \;<\; \mu^q + \tau^q, \tag{7}$$

that require exploration of the inside and outside sets. Observe that for given μ and τ the range of dissimilarity values $d(v, x_o)$ that satisfy (7) for $q > 1$ is larger than the range of dissimilarity values $d(v, x_o)$ that satisfy (6). Furthermore, the range of dissimilarity values $d(v, x_o)$ that satisfy (7) shrinks monotonically with increasing q. One then would expect that pruning of VP trees in q-metric spaces is more efficient for larger q. This expectation is confirmed by experiments (Section F).

2.3 VANTAGE POINT TREES AND ∞-METRIC STRUCTURE

As q grows, the q-metric dissimilarity function approaches an ultrametric dissimilarity function and Conditions (q-CI) and (q-CO) reduce to conditions that involve maxima instead of sums of q-powers ,

$$\max\big(\, d(v, x_o),\, \tau\,\big) < \mu \quad (\infty\text{-CI}) \qquad\qquad \max\big(\,\mu,\, \tau\,\big) \leq d(v, x_o'') \quad (\infty\text{-CO})$$

These conditions can be further simplified if we recall that $\tau \leq d(v, x_o')$ in (∞-CI) and that $\tau \leq d(v, x_o'')$ in (∞-CO). These observations imply that the presence of τ in these inequalities is moot as conditions (∞-CI) and (∞-CO) can be reduced to

$$d(v, x_o') \;<\; \mu, \qquad \mu \;\leq\; d(v, x_o''). \tag{8}$$

We show in Proposition 4 in Appendix C.3 that if we are searching in an ∞-metric space we can prune branches of VP trees using these two conditions. For queries x_o' such that $d(v, x_o') < \mu$ [cf. (∞-CI)] we can restrict the search for a nearest neighbor to the inside set X'. For queries x_o'' such that $\mu \leq d(v, x_o'')$ [cf. (∞-CO)] we can restrict the search for a nearest neighbor to the outside set X''.

Remarkably, the conditions in (8) are mutually exclusive. Every query must satisfy one and exactly one of these two conditions. This fact implies that in each vantage point comparison we discard one branch of the VP tree and that the cost of finding a nearest neighbor is upper bound by the depth of the depth of the VP tree. We summarize this claim in the following theorem

Theorem 1. *Consider a dataset X, a query x_o, a dissimilarity function d satisfying the strong triangle inequality* (3) *and a vantage point tree $T(X)$ constructed by the recursive partition of X into vantage points and their corresponding inside and outside sets [cf.* (5)*]. The number of comparisons $c(x_o)$ needed to find a nearest neighbor of x_o in the set X is bounded by the depth $h(T(X))$ of the VP tree $T(X)$,*

$$c(x_o) \leq h(T(X)). \tag{9}$$

Proof. See Appendix C. □

When we split the set X into a vantage point v and corresponding inside and outside sets, we can choose the radius μ as the median of the distances

$$\mu \;=\; \underset{x \in X,\, x \neq v}{\text{median}}\; d(v, x). \tag{10}$$

If there is only one point $x \in X$ that attains the median, this choice results in a balanced assignment of points to the inside and outside sets. They each contain the same number of points plus or minus one – to account for odd numbers of points to be assigned. If this happens throughout the recursive construction of the VP tree, the resulting tree is said to be balanced and has a depth equal to the ceiling of the base-2 logarithm of the number of points in the original dataset. I.e. the depth of a balanced tree is $h(T(X))) = \lceil \log_2 m \rceil$. According to Theorem 1, this quantity is also a bound for the worst case number of comparisons for any query in the case of ultrametric dissimilarity functions.

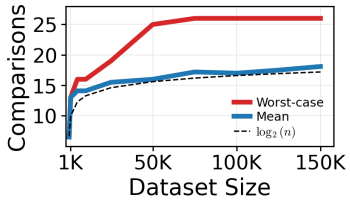


Figure 2: **VP Tree search complexity on a ∞-metric space**. The worst-case bound in comparisons corresponds to the depth of the tree.

In practice, we do observe that several points may achieve the median in (10). As per (5), all of these points are assigned to the outside set X'' resulting in unbalanced VP trees. While this yields trees with depth larger than $h(T(X))) = \lceil \log_2 m \rceil$, we still observe a logarithmic grow in the depth of the VP tree. A typical empirical evaluation is shown in Figure 2. We also observe that the mean number of comparisons stays close to the ideal depth $h(T(X))) = \lceil \log_2 m \rceil$ suggesting that the path to most leaves of the VP tree is close to this ideal value.

2.4 APPROXIMATE SEARCH WITH q-METRIC SPACES

It follows from Theorem 1 that, given a choice, we would like to solve nearest neighbor search in ultrametric spaces. Alas, dissimilarity metrics of interest do not satisfy the strong triangle inequality.

Some are not even metric. Due to this mismatch, we propose here an approach to *approximate* nearest neighbor search based on the development of projection and embedding operators to compute and approximate q-metric distances:

Projection Operator. The projection operator P_q maps dissimilarities $d(x, x') \in D$ that do not necessarily satisfy the q-triangle inequality into distances $d_q(x, x') \in D_q$ that do satisfy the q-triangle inequality (Section 3).

Embedding Operator. The embedding operator Φ_q is a learned map from vectors $x \in \mathbb{R}^n$ into vectors $x_q \in \mathbb{R}^s$ such that 2-norms $\|x_q - x'_q\|_2$ approximate q-distances $d_q(x, x')$. The map is trained on the dataset X and applied to queries x_o (Section 4).

We use the projection operator P_q to process the dataset $G = (X, D)$ to produce a graph $G_q = (X, D_q)$ such that any three points $x, y, z \in X$ satisfy the q-triangle inequality. This is a one-time preprocessing cost that we leverage for all queries. We use the learned embedding operator Φ_q to approximate the q-metric distances between a query x_o and points $x \in X$ as $d_q(x_o, x) \approx \|x_{oq} - x_q\|_2$ with $x_{oq} = \Phi_q(x_o)$. This is needed because computing the true q-metric distance $d_q(x_o, x)$ requires comparing x_o to all points in X (Section 3). Approximating q-metric distances with the embedding operator E_q renders this search methodology approximate. Experiments in Section 5, show that we are competitive with existing approximate search methods.

3 Projecting Dissimilarity Functions on q-Metric Spaces

We seek to design a *q-metric projection* $P_q : (X, D) \to (X, D_q)$ where, for arbitrary input dissimilarities D, the distances in D_q satisfy the q-triangle inequality given in (4). Observe that finding a feasible projection is trivial. For example, consider a projection that assigns all distances in D_q to be 1, regardless of the input D. In this case, the image space (X, D_q) becomes an ultrametric (hence, a valid q-metric for all q). However, performing nearest neighbor search in (X, D_q) would be a poor proxy for searching in (X, D), as all distance information was lost in the projection. Motivated by this example, we build on the work of Segarra et al. (2019), which imposes conditions on P_q to ensure that meaningful distance information is preserved through the projection. In their work, these requirements were formalized as the following two axioms:

(A1) Axiom of Projection. The q-metric graph $G_q = (X, D_q)$ is a fixed point of the projection, i.e.,

$$P_q(G_q) = G_q. \tag{A1}$$

(A2) Axiom of Transformation. Consider any two graphs $G = (X, D)$ and $G' = (X', D')$ and a dissimilarity reducing map $\varphi : X \to X'$ such that $D(x, y) \geq D'(\varphi(x), \varphi(y))$ for all $x, y \in X$. Then, the output q-metric graphs $(X, D_q) = P_q(G)$ and $(X', D'_q) = P_q(G')$ satisfy, for all $x, y \in X$,

$$D_q(x, y) \geq D'_q(\varphi(x), \varphi(y)). \tag{A2}$$

Axiom (A1) is natural in our setting: if the graph under study already satisfies the q-triangle inequality, then the projection should introduce no distortion and simply return the same graph. Axiom (A2) enforces a notion of monotonicity: if the distances in one graph dominate those in another, this dominance should be preserved under projection. Though seemingly lax, Axioms (A1) and (A2) impose significant structure on the projection P_q. In fact, we leverage the fact that *there exists a unique projection P_q that satisfies Axioms (A1) and (A2)* Segarra et al. (2019). Moreover, we show that *this projection preserves nearest neighbors*. To formally state these results, we introduce the canonical q-metric projection.

Canonical q-metric projection $P_q^\star$. Consider a graph equipped with a dissimilarity $G = (X, D)$ and let C_{xy} denote the set of all paths from x to y, where $x, y \in X$. For a given path $c = [x = x_0, x_1, \ldots, x_l = y] \in C_{xy}$, we define its q-length as $\ell_q(c) = \|[d(x, x_1), \ldots, d(x_{l-1}, y)]\|_q$. That is, the q-length of a path is the q-norm of the vector containing the dissimilarities along the path's edges. We define the canonical q-metric projection $(X, D_q) = P_q^\star(X, D)$ as

$$d_q(x, y) = \min_{c \in C_{xy}} \ell_q(c). \tag{11}$$

In words, the canonical q-metric projection computes the all-pairs shortest paths using the q-norm as the path cost function. A first observation is that $P_q^\star$ is a valid q-metric projection; i.e., that the output graph is guaranteed to satisfy the q-triangle inequality (see Appendix E). More importantly, as it was already proved in Segarra et al. (2019), $P_q^\star$ is uniquely characterized by Axioms (A1) and (A2).

Theorem 2 (Existence and uniqueness)**.** *The canonical projection $P_q^\star$ satisfies Axioms (A1) and (A2). Moreover, if a q-metric projection P_q satisfies Axioms (A1) and (A2), it must be that $P_q = P_q^\star$.*

As we discuss next, Theorem 2 has direct practical applications for nearest-neighbor search.

3.1 Search in Projected q-metric Spaces

Since the Axiom of Transformation (A2) is satisfied, we expect $P_q^\star$ to preserve up to some extent the distance ordering of the original graph, but this was never proved. In particular, we now show that the nearest neighbors are preserved by this projection. To formalize this, consider the graph $H = (Y, E)$ whose node set $Y = X \cup \{x_o\}$ includes the dataset X and the query x_o. The dissimilarity set E consists of the original dissimilarities D as well as the dissimilarities between the query and each data point, i.e., $E(x_o, x) = d(x_o, x)$ for all $x \in X$. We now apply the canonical projection to this extended graph, obtaining $(Y, E_q) = P_q^\star(H)$, where E_q includes projected distances of the form $E_q(x, x_o)$ between each data point $x \in X$ and the query x_o. We next state that this projection preserves the identity of the nearest neighbor of x_o.

Proposition 1. *Given a graph $G = (X, D)$ and a query x_o, define $H = (Y, E)$ with $Y = X \cup \{x_o\}$ and let $(Y, E_q) = P_q^\star(H)$ be the projected graph. Then, the set of nearest neighbors satisfies*

$$\hat{x}_o \;\equiv\; \operatorname*{argmin}_{x \in X} E(x, x_o) \;\subseteq\; \operatorname*{argmin}_{x \in X} E_q(x, x_o). \tag{12}$$

Moreover, for the case where $q < \infty$, the result holds with an equality.

Proposition 1 shows that the canonical projection $P_q^\star$ in (11) preserves the nearest neighbor. This is enticing because we have argued that search in q-metric spaces is easier (Figure 1) and proved that search in ∞-metric spaces requires a logarithmic number of comparisons (Theorem 1). However, Equation (12) does not immediately help us solve the original nearest neighbor problem in (1). The proposition is stated for the projected dissimilarity set E_q, which requires access to the full set E, including all dissimilarities $E(x_o, x) = d(x_o, x)$ for every $x \in X$. Thus, computing E_q requires comparing x_o with all points in X, which defeats the purpose of reducing the number of comparisons $c(x_o)$. A broader analysis of this projection complexity is available in Appendix E.3. In the following section, we address the projection cost challenge by learning an embedding operator that enables us to *approximate* the values $E_q(x, x_o)$ without computing all pairwise distances explicitly.

4 Embedding Operator: Learning to Approximate q-Metrics

We approximate q-metrics $E_q(x, x_o)$ with the 2-norm of a parameterized embedding $\Phi(x; \theta)$. Formally, we want to find a function $\Phi(x; \theta) : \mathbb{R}^n \to \mathbb{R}^s$ such that for any $x_o \in \mathbb{R}^n$ and $x \in X$:

$$E_q(x, x_o) = \big\|\Phi(x; \theta) - \Phi(x_o; \theta)\big\| \tag{13}$$

where $\|\cdot\|$ denotes the 2-norm of a vector. We fit the parameter θ to approximate distances $D_q(x, y)$ given by the canonical projection $D_q = P_q^\star(D)$, by minimizing a quadratic loss,

$$\ell_{\mathrm{D}}(x, y) = \Big[\, D_q(x, y) - \big\|\Phi(x; \theta) - \Phi(y; \theta)\big\| \,\Big]^2. \tag{14}$$

The loss is inspired in type of Stress functional used in metric Multidimensional Scaling (mMDS). In practice, we also enforce neighborhood accuracy by minimizing the contrastive loss

$$\ell_{\mathrm{C}}(x) = \tfrac{1}{\tau}\|\Phi_\theta(x) - \Phi_\theta(x^+)\|_2^2 + \log \sum_{y \in X \setminus \{x\}} e^{-\|\Phi_\theta(x) - \Phi_\theta(y)\|_2^2/\tau}, \tag{15}$$

where x^+ is sampled from the k nearest neighbors of x under D and $\tau > 0$ is a temperature. We minimize the combined objective over the dataset X:

$$\theta^\star = \operatorname*{argmin}_\theta \sum_{x, y \in X} \ell_{\mathrm{D}}(x, y) + \mu \sum_{x \in X} \ell_{\mathrm{C}}(x). \tag{16}$$

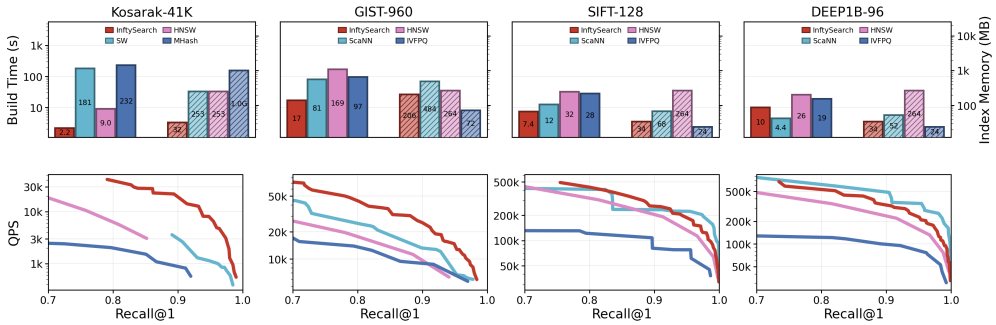


Figure 3: **Infinity Search is competitive in approximate nearest neighbor search.** We learn approximate infinity embeddings on four real-world datasets and perform nearest-neighbor search using VP-Trees. When compared with efficient approximate nearest neighbor methods, Infinity Search achieves competitive query throughput (bottom), with particularly strong performance on high-dimensional datasets such as KOSARAK and GIST. Because trees can be constructed efficiently, Infinity Search leads in index-construction times (top-left) in three of the four datasets and ranks second in memory usage (top-right).

Further notice that although we train θ over the dataset X we expect it to generalize to estimate the q-metric $E_q(x, x_o)$. Our numerical experiments in the next section indicate a successful generalization. For additional details on training see Appendix F.

5 Experiments

In this section, we evaluate our end-to-end approach against exact and approximate vector search methods, including HNSW (Malkov and Yashunin, 2020), IVF-PQ (Jégou et al., 2011), and the more recent ScaNN (Guo et al., 2020). We use 1M points from four datasets spanning a range of dimensionalities and similarity metrics. For cosine similarity, we use DEEP1B, with 96-dimensional representations, and SIFT, with 128-dimensional vectors. For Euclidean distance, we use the 960-dimensional GIST representations. Finally, for Jaccard similarity, we use KOSARAK, which contains sparse web-page interactions in a 41K-dimensional space. KOSARAK presents two additional challenges: it is high-dimensional and relies on the less conventional Jaccard similarity metric. We therefore also consider methods designed to operate effectively in these settings, such as SW-graph (Malkov et al., 2014), which is a modification of HNSW, and MinHash-LSH Forest (Indyk and Motwani, 1998b), which is specifically designed for Jaccard similarity.

In this setup, we evaluate methods based on the tradeoff between search accuracy and computational efficiency. Efficiency is measured along three dimensions: index construction time, reported in seconds (s), search throughput, reported as queries per second (QPS) and index size in megabytes (MB). Accuracy is evaluated as Recall@1, defined as the fraction of queries for which the true nearest neighbor is retrieved. Our method learns the embedding offline, prior to index construction, and computes the query projection before search; therefore, we treat both operations as preprocessing steps. Additional results for retrieving $k \in \{5, 10\}$ neighbors, as well as a broader comparison with the algorithms included in ANN-Benchmarks (Aumüller et al., 2020), are provided in Appendix F.

Infinity Search is competitive in approximate nearest neighbor search. Across the evaluated datasets, our method achieves performance comparable to the baseline methods for nearest-neighbor search. Each point on the throughput–accuracy curve in Figure 3 represents a different hyperparameter setting in the benchmark, showing that Infinity Search achieves speed gains with minimal accuracy trade-offs, thereby satisfying (C1). These gains are particularly pronounced on high-dimensional embeddings, such as KOSARAK-41K and GIST960. KOSARAK is high-dimensional (41K), sparse, and evaluated using Jaccard similarity, making it unsupported by many ANN implementations. Under this setting, Infinity Search achieves its strongest performance, which highlights the ability of our method to accommodate arbitrary dissimilarities, while scaling to higher-dimensional data.

Infinity Search achieves the fastest index construction on three of the four datasets, ranking second on DEEP1B. We attribute this efficiency to the VP-Tree index our method builds upon, whose simple

structure requires only a handful of computations to construct. Infinity Search also ranks second in index memory, just behind IVF-PQ, which requires no additional index structure. The reader will find a more detailed analysis of this construction phase in Appendix C.

$\lceil \log_2(n) \rceil = 20$	SIFT-128	DEEP1B-96	GIST-960	KOSARAK-41K
Mean / Max	20.75 / 28	20.90 / 29	20.92 / 28	21.07 / 32

Table 1: **VP-Tree depth remains close to logarithmic.** We build VP-Trees over approximate infinity embeddings on large real-world datasets. Across datasets, the resulting trees exhibit near-logarithmic depth, close to the optimal value of a balanced tree.

VP-Tree depth remains close to logarithmic. Theorem 1 claims that nearest neighbor search can be solved with a number of comparisons equal to the depth of a VP-Tree. However, this depth need not be close to its optimal logarithmic value, which is attained only by balanced trees. Table 1 shows the depths of VP-Trees constructed on real-world datasets, demonstrating that approximate balanced trees can be built with near-logarithmic depth even on millions of points.

Learning error falls below the pruning threshold. When $q = \infty$, the canonical projection $P^{\star}_{\infty}$ maps points into an ∞-metric space. Computing this projection directly is impractical when fast search is required. In Section 5, we propose a learning approach that obtains approximate ∞-metric embeddings by minimizing their pairwise distance to the canonical projections. If the approximation error is large, VP-Tree pruning may discard true nearest neighbors, impairing recall. Conversely, if the approximation error remains below the distance threshold μ between a vantage point and its children, tree traversal is expected to remain accurate under the approximate embeddings.

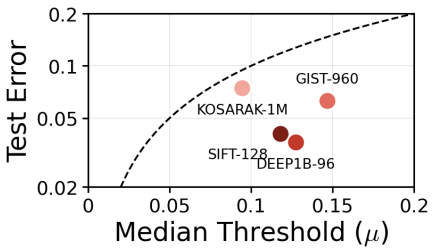


Figure 4: **Learning error falls below pruning threshold.** When trained on 1M DEEP1B points, embeddings exhibit errors that fall below the minimum threshold across vantage points and the average tree traversal exhibits no pruning errors.

In Figure 4, we compare the minimum margin in the tree with the average test error of the learned embeddings. Across datasets, the approximation error remains sufficiently small relative to the minimum threshold of the tree, allowing effective pruning while preserving search accuracy. We note that, since the reported error is an average quantity, perfect traversal cannot be guaranteed for every query. Nevertheless, the results suggest that the learned embeddings provide sufficiently accurate distances for reliable search.

Infinity Search scales. In Section 2, Figure 2 reported the growth in the number of comparisons under exact ∞-metric embeddings, obtained by applying the canonical projection $P^{\star}_{\infty}$. Since this projection is computationally expensive, the scaling law was originally tested only up to 150K points, showing a near-logarithmic average depth. By approximating the canonical projection, we can leverage the near-∞-metric structure at larger scales. Figure 5 shows the scaling of the number of comparisons for learned embeddings on up to 5M points. We highlight that the scaling law observed for the approximate embeddings closely resembles that of exact projections. Furthermore, the gap between the maximum number of comparisons and the logarithmic optimum appears to remain constant across scales. We defer the conclusion to Appendix B.

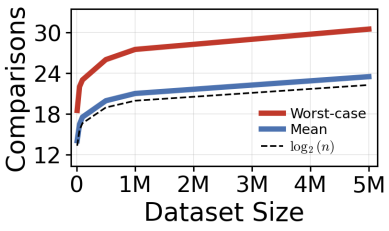


Figure 5: **Infinity Search scales.** We train Infinity Search on a subset of 40K points from DEEP1B and then use it to search over up to 5M points, where our method exhibits a favorable scaling behavior.

AI USE STATEMENT

In this work, we used generative AI tools for implementing experimental methods and assist in the writing of proofs.

We have not used generative AI tools help to develop theoretical models or conceptual frameworks or formulate mathematical claims. We have neither used AI tools to generate synthetic data sets, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, assist with translation, clean and reformat datasets, support qualitative and thematic data analysis or interpret results.

Additionally, we used generative AI tools for suggest experimental parameters, create or edit software code and identify relevant literature. We have reviewed all AI-assisted work. For example, AI-proposed code has been manually tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Given the nature of the manuscript, we do not believe an additional ethic statement is needed.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, a ready-to-run script has been created to reproduce the main experiments of the manuscript. Moreover, this script launches a Docker container, further ensuring isolation of the results. In addition, the script has been made as simple and self-contained as possible and includes comments that improve its interpretability. A full description of the hardware and environment used to generate the experimental results is also included in the Experimental section.

Regarding theory, proofs of the theoretical statements and their explanations are also provided in the Appendix.

REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Aumüller, M., Bernhardsson, E., and Faithfull, A. (2020). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374.

Babenko, A. and Lempitsky, V. (2015). The inverted multi-index. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(6):1247–1260.

Babenko, A. and Lempitsky, V. (2016). Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2055–2063.

Beygelzimer, A., Kakade, S., and Langford, J. (2006). Cover trees for nearest neighbor. In *ICML*.

Carlsson, G., Mémoli, F., Ribeiro, A., and Segarra, S. (2017). Admissible hierarchical clustering methods and algorithms for asymmetric networks. *IEEE Transactions on Signal and Information Processing over Networks*, 3(4):711–727.

Carlsson, G., Mémoli, F., Ribeiro, A., and Segarra, S. (2014). Hierarchical quasi-clustering methods for asymmetric networks. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 352–360, Bejing, China. PMLR.

Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., and Wang, J. (2021). Spann: Highly-efficient billion-scale approximate nearest neighbor search.

Dovgoshey, O. (2025). Totally bounded ultrametric spaces and locally finite trees. *arXiv preprint arXiv:2502.04228*.

Draganov, A., Weber, P., Jørgensen, R. S. M., Beer, A., Plant, C., and Assent, I. (2025). I want 'em all (at once) – ultrametric cluster hierarchies. *arXiv preprint arXiv:2502.14018*.

Echihabi, K., Zoumpatianos, K., and Palpanas, T. (2021). High-dimensional similarity search for scalable data science. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2369–2372.

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345.

Greenhoe, D. J. (2016). Properties of distance spaces with power triangle inequalities. *arXiv preprint arXiv:1610.07594*.

Groisman, P., Jonckheere, M., and Sapienza, F. (2022). Nonhomogeneous Euclidean first-passage percolation and distance learning. *Bernoulli*, 28(1):255 – 276.

Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. (2020). Accelerating large-scale inference with anisotropic vector quantization. In Daumé III, H. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR.

Indyk, P. and Motwani, R. (1998a). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*.

Indyk, P. and Motwani, R. (1998b). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–613. ACM.

Iwasaki, M. and Miyazaki, D. (2018). Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. *ArXiv*, abs/1810.07355.

Jääsaari, E., Hyvönen, V., and Roos, T. (2024). Lorann: Low-rank matrix factorization for approximate nearest neighbor search. *Advances in Neural Information Processing Systems*, 37:102121–102153.

Jégou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Kalantidis, Y. and Avrithis, Y. (2014). Locally optimized product quantization for approximate nearest neighbor search. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2329–2336.

Li, C., Zhang, M., Andersen, D. G., and He, Y. (2020a). Improving approximate nearest neighbor search through learned adaptive early termination. In *Proc. SIGMOD*, pages 2539–2554.

Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., and Lin, X. (2020b). Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488. Publisher Copyright: © 1989-2012 IEEE.

Malkov, Y., Ponomarenko, A., Logvinov, A., and Krylov, V. (2014). Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68.

Malkov, Y. A. and Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Muñoz, L. A. B. (2012). Understanding (dis)similarity measures. *CoRR*, abs/1212.2791.

Murtagh, F. and Contreras, P. (2012). The future of search and discovery in big data analytics: Ultrametric information spaces.

Newman, D. (2008). Bag of Words. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZG6P.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175.

Pan, J. J., Wang, J., and Li, G. (2024). Survey of vector database management systems. *The VLDB Journal*, 33(5):1591–1615.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Rácz, B., Bodon, F., and Schmidt-Thieme, L. (2005). Benchmarking frequent itemset mining algorithms: from measurement to analysis. In Goethals, B., Nijssen, S., and Zaki, M. J., editors, *Proceedings of ACM SIGKDD International Workshop on Open Source Data Mining (OSDM'05)*, pages 36–45, Chicago, IL, USA.

Segarra, S., Carlsson, G., Memoli, F., and Ribeiro, A. (2019). Metric representations of network data. *arXiv preprint arXiv:1911.00164*.

Segarra, S., Roddenberry, T. M., Mémoli, F., and Ribeiro, A. (2020). Metric representations of networks: A uniqueness result. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5345–5349.

Simovici, D. A., Vetro, R., and Hua, K. (2017). *Ultrametricity of Dissimilarity Spaces and Its Significance for Data Mining*, pages 141–155. Springer International Publishing, Cham.

Smith, Z., Chowdhury, S., and Mémoli, F. (2016). Hierarchical representations of network data with optimal distortion bounds. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 1834–1838. IEEE.

Sun, P., Simcha, D., Dopson, D., Guo, R., and Kumar, S. (2023). Soar: improved indexing for approximate nearest neighbor search. *Advances in Neural Information Processing Systems*, 36:3189–3204.

Uhlmann, J. K. (1991). Satisfying general proximity / similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179.

Ukey, N., Yang, Z., Li, B., Zhang, G., Hu, Y., and Zhang, W. (2023). Survey on exact knn queries over high-dimensional data space. *Sensors*, 23(2):629.

Wang, J., Shen, H. T., Song, J., and Ji, J. (2014). Hashing for similarity search: A survey.

Warshall, S. (1962). A theorem on boolean matrices. *J. ACM*, 9(1):11–12.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yang, K., Wang, H., Xu, B., Wang, W., Xiao, Y., Du, M., and Zhou, J. (2021). Tao: A learning framework for adaptive nearest neighbor search using static features only. *arXiv preprint arXiv:2110.00696*.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM-SIAM Symposium on Discrete Algorithms*.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity Search: The Metric Space Approach*. Springer.

A Related Work

The literature on vector-based nearest–neighbour search is vast and spans many decades. We briefly review the families that are most germane to our contributions and direct the reader to existing surveys (Echihabi et al., 2021; Ukey et al., 2023; Li et al., 2020b) for a broader analysis.

For data sets endowed with a proper metric, pruning rules derived from the standard triangle inequality have long been exploited through pivot- or vantage-point trees, k-d trees, ball trees, and their variants (Yianilos, 1993; Uhlmann, 1991; Beygelzimer et al., 2006). Although these methods can yield exact results, they typically struggle with dimensionality and rely on existing metric structure of the date have been largely ouperformed by inverted file indexes (Babenko and Lempitsky, 2015), hashing (Indyk and Motwani, 1998a; Wang et al., 2014), quantisation (Kalantidis and Avrithis, 2014) and graph based methods (Malkov and Yashunin, 2020) such as HNSW (Malkov and Yashunin, 2020). Approaches—such as SPANN (Chen et al., 2021), and ONNG (Iwasaki and Miyazaki, 2018)— have optimized memory layout, link degree, or storage hierarchy to scale to billion-point datasets.

While many of these methods can offer strong performance, recent works increasingly seek to embed learning into the indexing process for data-aware improvements. ScaNN (Guo et al., 2020) jointly optimizes clustering and quantization, SOAR adds orthogonality-amplified residuals to further boost ScaNN's efficiency (Sun et al., 2023). Learned policies can also steer the search itself: AdaptNN predicts early-termination thresholds online (Li et al., 2020a), and Tao learns to set them *before* query execution using only static local-intrinsic-dimension features (Yang et al., 2021). On the other hand, LoRANN replaces PQ scoring with a supervised low-rank regressor that outperforms traditional IVF on billion-scale data (Jääsaari et al., 2024).

Industrial adoption of large language models has driven the emergence of "vector DBMSs", see (Pan et al., 2024) and references therein. Despite many developments in compression, partitioning, graph navigation, and hybrid attribute-vector query planning, most if not all systems index the *original* dissimilarities - or an approximation – and thus inherit their metric limitations. In contrast, we propose to pre-process data to obtain a q-metric space whose structure is provably more favorable for search.

B Conclusion

Driven by the insight that search in ultrametric spaces remains close to logarithmic, in this paper we leverage a projection that endows a set of vectors with arbitrary dissimilarities with q-metric structure, while preserving the nearest neighbor. This projection of a dataset is computed as the shortest path in a graph using the q-norm as path length. To address the challenge of computing q-metric distances for queries efficiently, we developed a learning approach that embeds vectors into a space where Euclidean distances approximate q-metric distances. Our experiments real-world datasets, which included high-dimensional sparse data with less common dissimilarity functions, demonstrated that our method is competitive with strong approximate search algorithms at low scale.

C Vantage Point Trees

To search in q-metric and ∞-metric spaces we adopt vantage point (VP) trees (Yianilos, 1993). To explain VP trees consider a set of points X and a dissimilarity function d. Select a vantage point $v \in X$ and an associated radius μ. Proceed then to partition the space X into v itself and sets X' and X'' containing points (other than v) that are inside and outside the ball of radius μ centered at v, respectively,

$$\begin{aligned} X' &:= \Big\{ x \in X \,:\, d(v,x) < \mu \,,\, x \neq v \Big\}, \\ X'' &:= \Big\{ x \in X \,:\, \mu \leq d(v,x) \qquad\quad \Big\}. \end{aligned} \tag{17}$$

A VP tree is built by recursive partition of the sets X' and X'' using corresponding vantage points and radiuses. I.e., we select a vantage point $v' \in X'$ and a radius μ' to divide X' into inside and outside sets and do the same for X''. We then partition the new group of inside and outside sets and repeat until we reach indivisible sets with only one point (see Section D and Figure 1 for details). We

emphasize, as this will be important later, that the border points at which $\mu = d(v, x)$ are assigned to the outside set X''.

VP trees were developed for efficient search of metric spaces (Section C.1). We will see here that they can also be used for efficient search of q-metric (Section C.2) and ∞-metric (Section C.3) spaces. We will further see that we expect exploration of VP trees to be more efficient in q-metric spaces with larger q (Section C.2). In particular, when $q = \infty$ we will show that we are guaranteed to find a nearest neighbor $\hat{x}_o$ of any query x_o with a number of comparisons that is, at most, equal to the depth of the VP tree (Section C.3).

C.1 Vantage Point Trees in Metric Spaces

The inside and outside sets X' and X'' reduce the complexity of finding the nearest neighbor of a query x_o when the dissimilarity function $d(x, y)$ corresponds to a (standard) metric space (Figure 1). This fact is well known, see e.g., Yianilos (1993), but we repeat the relevant facts here to facilitate understanding of the material in Sections C.2 and C.3.

Start by assuming that we have available an upper bound $\tau \geq d(x_o, \hat{x}_o)$ on the distance between the query x_o and a nearest neighbor $\hat{x}_o$. This upper bound is typically a distance to an earlier vantage point and must satisfy $\tau \leq d(v, x_o)$ because we can otherwise set $\tau = d(v, x_o)$ as an upper bound on $d(x_o, \hat{x}_o)$. We can then show that the following proposition holds.

Proposition 2. *Consider a set of points X, a vantage point $v \in X$, and a radius μ to define inside and outside sets X' and X'' as in* (17)*. We are given a query x_o and an upper bound $d(v, x_o) \geq \tau \geq d(x_o, \hat{x}_o)$ on the nearest neighbor distance $d(x_o, \hat{x}_o)$. If the dissimilarity function $d(x, y)$ satisfies the triangle inequality* (2)*, we have that*

- **(CI)** *If $d(v, x_o) + \tau < \mu$, then, either v is a nearest neighbor or there exist a nearest neighbor in the inside set; i.e., $\hat{x}_o \cap (X' \cup v) \neq \emptyset$.*
- **(CO)** *If $\mu + \tau \leq d(v, x_o)$, then, there exist a nearest neighbor in the outside set; i.e., $\hat{x}_o \cap X'' \neq \emptyset$.*

Proof. We show first that if the condition $d(v, x_o) + \tau < \mu$ in (CI) holds, there is no point x'' in the outside set X'' that can be a bearest neighbor of x_o. To see that this is true, recall that for any point $x'' \in X''$ we have $\mu \leq d(v, x'')$ and apply the triangle inequality to the triangle (v, x_o, x'') to write

$$\mu \;\leq\; d(v, x'') \;\leq\; d(v, x_o) + d(x_o, x''). \tag{18}$$

Because we are assuming that the condition in (CI) holds, we also know that $d(v, x_o) + \tau < \mu$. This inequality is compatible with (18) if and only if,

$$\tau < d(x_o, x''). \tag{19}$$

Further recall that according to the definition of the upper bound τ we must have $d(x_o, \hat{x}_o) \leq \tau$. Consequently,

$$d(x_o, \hat{x}_o) \leq \tau < d(x_o, x''). \tag{20}$$

This means that all points $x'' \in X''$ have distances to x_o larger than the distance to its nearest neighbor $\hat{x}_o$. Thus, there is no nearest neighbor in X'' and we must therefore have that either v or some other point in X' is a nearest neighbor of x_o.

Analogously, we show that if the condition $\mu + \tau \leq d(v, x_o)$ in (CO) holds, we must have $d(x_o, \hat{x}_o) < d(x_o, x')$ for any point x' in the inside set X'. To see that this is true, apply the triangle inequality to the triangle (v, x', x_o) and recall that for any point $x' \in X'$ we must have $d(v, x') < \mu$ to write

$$d(v, x_o) \;\leq\; d(v, x') + d(x', x_o) \;<\; \mu + d(x', x_o). \tag{21}$$

Because we are assuming that the condition in (CO) holds we also know that $\mu + \tau \leq d(v, x_o)$. This inequality is compatible with (21) if and only if,

$$\tau < d(x_o, x'). \tag{22}$$

Further recall, as we did in (20), that the definition of the upper bound τ implies that $d(x_o, \hat{x}_o) \leq \tau$. We can thus write

$$d(x_o, \hat{x}_o) \leq \tau < d(x_o, x'). \tag{23}$$

This means that there is no nearest neighbor in X'. We therefore must have that either v is a nearest neighbor or that we have a nearest neighbor in X''. In this case we know that v is not a nearest neighbor of x_o because of condition (CO). Indeed, because $\mu > 0$ we have that $d(x_o, \hat{x}_o) \leq \tau < \mu + \tau \leq d(v, x_o)$. □

Because of (CI) and (CO) we can sometimes eliminate entire branches of VP trees during the search for a nearest neighbor of a query x_o. If the condition $d(v, x_o) + \tau < \mu$ in (CI) holds, we know that either v is a nearest neighbor or that we can find a nearest neighbor in X'. We can therefore discard all points in X'' and focus the search on the inside set X'. Likewise, if the condition $\mu + \tau \leq d(v, x_o)$ in (CO) holds, we know that a nearest neighbor $\hat{x}_o$ can be found in X''. We can therefore discard all points in X' and focus the search on the outside set X''. If we do this recursively, each comparison with a vantage point has the potential of halving the number of points that remain in contention as nearest neighbor candidates (see Section D and Algorithm 2 for details).

This potential to discard points from consideration as nearest neighbors is not always realized because it can be that neither of the conditions in (CI) or (CO) hold true. Indeed, for queries x_o such that the distance to the vantage point satisfies

$$\mu - \tau \;\leq\; d(v, x_o) \;<\; \mu + \tau, \tag{24}$$

neither of the conditions in (CI) or (CO) hold true and we must explore both sets X' and X'' further. We will see next that this limitation is mitigated in q-metric spaces (Section C.2) and eliminated in ∞-metric spaces (Section C.3).

C.2 Vantage Point Trees in q-Metric Spaces

The partition of a set X into the inside and outside sets defined in (17) for a vantage point v and radius μ is independent of the structure of the dissimilarity function $d(x, y)$. The partition is valid even if $d(x, y)$ is not metric. Proposition 2 *does* require $d(x, y)$ to be metric and shows that in such a case comparisons of a query x_o with a vantage point v *may* confine the search for the nearest neighbor $\hat{x}_o$ to either the inside set X' or the outside set X''.

We show here that an analogous result holds for q-metric dissimilarity functions. The difference is that the conditions in (CI) and (CO) compare q-powers of the respective quantities. This modification matches the use of q-powers of distances in the q-triangle inequality (4).

Proposition 3. *Consider a set of points X, a vantage point $v \in X$, and a radius μ to define inside and outside sets X' and X'' as in* (17)*. We are given a query x_o and an upper bound $d(v, x_o) \geq \tau \geq d(x_o, \hat{x}_o)$ on the nearest neighbor distance $d(x_o, \hat{x}_o)$. If the dissimilarity function $d(x, y)$ satisfies the q-triangle inequality* (4)*, we have that*

(q-CI) *If $d^q(v, x_o) + \tau^q < \mu^q$, then, either v is a nearest neighbor or there exist a nearest neighbor in the inside set; i.e., $\hat{x}_o \cap (X' \cup v) \neq \emptyset$.*

(q-CO) *If $\mu^q + \tau^q \leq d^q(v, x_o)$, then, there exist a nearest neighbor in the outside set; i.e., $\hat{x}_o \cap X'' \neq \emptyset$.*

Proof. The proof is a ready extension of the proof of Proposition 2. Thus, we show first that if the condition $d^q(v, x_o) + \tau^q < \mu^q$ in (q-CI) holds, we must have $d(x_o, \hat{x}_o) < d(x_o, x'')$ for any point x'' in the outside set X''. To see that this is true, recall that for any point $x'' \in X''$ we must have $\mu \leq d(v, x'')$ and, consequently, we must also have $\mu^q \leq d^q(v, x'')$. Writing this latter inequality followed by the q-triangle inequality (4) to the triangle (v, x_o, x''), we can write,

$$\mu^q \;\leq\; d^q(v, x'') \;\leq\; d^q(v, x_o) + d^q(x_o, x''). \tag{25}$$

Because we are assuming that the condition in (q-CI) holds we also know that $d^q(v, x_o) + \tau^q < \mu^q$. This inequality is compatible with (18) if and only if,

$$\tau^q < d^q(x_o, x''). \tag{26}$$

This inequality implies that we must have $\tau < d(x_o, x'')$ for any $x'' \in X''$. To complete the claim recall that since τ is an upper bound on the nearest neighbor distance we know that $d(x_o, \hat{x}_o) \leq \tau$. Combining this inequality with (26) yields

$$d(x_o, \hat{x}_o) \leq \tau < d(x_o, x''). \tag{27}$$

We conclude that there is no nearest neighbor in X''. We therefore must have that either v is a nearest neighbor or there is a nearest neighbor in X'.

Analogously, we show that if the condition $\mu^q + \tau^q \leq d^q(v, x_o)$ in (q-CO) holds, we must have $d(x_o, \hat{x}_o) < d(x_o, x')$ for any point x' in the inside set X'. To see that this is true recall that since $d(v, x') < \mu$ for any point $x' \in X'$ we must also have that $d^q(v, x') < \mu^q$. Write now the q-triangle inequality for the triangle (v, x', x_o) and use the fact that $d^q(v, x') < \mu^q$ to conclude that

$$d^q(v, x_o) \;\leq\; d^q(v, x') + d^q(x', x_o) \;<\; \mu^q + d^q(x', x_o). \tag{28}$$

Because we are assuming that the condition in (q-CO) holds we also know that $\mu^q + \tau^q \leq d^q(v, x_o)$. This inequality is compatible with (21) if and only if,

$$\tau^q < d^q(x_o, x'). \tag{29}$$

We therefore must have that $\tau < d(x_o, x') > \tau$ for any $x' \in X'$. Since we also know that $d(x_o, \hat{x}_o) \leq \tau$, we conclude that

$$d(x_o, \hat{x}_o) \leq \tau < d(x_o, x'). \tag{30}$$

Thus, there cannot be a nearest neighbor of x_o in X'. It follows that we must either have a nearest neighbor in X'' or v itself is a nearest neighbor. We know that the latter is not possible because since $\mu > 0$ and condition (q-CO) holds, we have that $d^q(x_o, \hat{x}_o) \leq \tau^q < \mu^q + \tau^q \leq d^q(v, x_o)$, which in turn implies $d^q(x_o, \hat{x}_o) < d(v, x_o)$. □

Because of (q-CI) and (q-CO) it may be possible to prune branches of the VP tree when searching for the nearest neighbor of a query x_o when the dissimilarity function $d(x, y)$ satisfies the q-triangle inequality. If the condition $d^q(v, x_o) + \tau^q < \mu^q$ in (q-CI) holds, we prune the outside set X'' and focus the search on the inside set X'. If the condition $\mu^q + \tau^q \leq d^q(v, x_o)$ in (q-CO) holds, we prune the inside set X' and focus the search on the outside set X''. This is analogous to the pruning of VP trees when the dissimilarity function $d(x, y)$ satisfies the regular triangle inequality – which is also particular case with $q = 1$.

As is the case of standard triangle inequalities, pruning is not always possible. If the distance $d(v, x_o)$ between the query and the vantage point is such that

$$\mu^q - \tau^q \;\leq\; d^q(v, x_o) \;<\; \mu^q + \tau^q, \tag{31}$$

neither of the conditions in (q-CI) or (q-CO) hold. We must therefore explore both, the inside set X' and the outside set X'' as we do in the case of standard triangle inequalities when (24) holds – this equation being a particular case of (31) corresponding to $q = 1$.

A remarkable fact is that (31) is stricter than (24) for any $q > 1$. I.e., for given μ and τ the range of values between $\mu^q - \tau^q$ and $\mu^q + \tau^q$ is smaller than the range of values between $\mu - \tau$ and $\mu + \tau$. This observation suggests that pruning of VP trees is more effective when data and queries are samples of a q-metric space than when they are samples of a standard metric space. Our experiments corroborate that this is true (see Section F.2, Figures 14a and 14b).

A second remarkable fact is that the range of values between $\mu^q - \tau^q$ and $\mu^q + \tau^q$ for given μ and τ shrinks with larger q and becomes empty as q approaches infinity. This observation indicates that pruning of a VP tree is likely to be always possible when the strong triangle inequality holds. We explain in the next section that this is indeed true.

C.3 Vantage Point Trees in ∞-Metric Spaces

When the dissimilarity function $d(x, y)$ satisfies the strong triangle inequality (3), the analogous of the conditions in (CI) and (q-CI) is

$$\max\big(\, d(v, x_o),\, \tau \,\big) \;<\; \mu. \tag{32}$$

Observe however that the upper bound τ on the distance to the nearest neighbor of the query x_o is not larger than $d(v, x_o)$ by construction. I.e., $d(v, x_o) \geq \tau \geq d(x_o, \hat{x}_o)$. Thus, the maximum in (32) is attained by $d(v, x_o)$, from where it follows that (32) simplifies to,

$$d(v, x_o) \;<\; \mu. \tag{33}$$

Likewise, the analogous of the conditions in (CO) and (q-CO) is the inequality,

$$\max\big(\,\mu,\,\tau\,\big)\;\leq\;d(v,x_o). \tag{34}$$

Again, we know that by construction we have $d(v,x_o)\geq\tau$. Thus (34) holds if and only if,

$$\mu\;\leq\;d(v,x_o). \tag{35}$$

The conditions in (33) and (35) are exact complements. One and only one of them holds for any value of $d(v,x_o)$. It remains for us to show that they are valid conditions for pruning a VP tree when the dissimilarity function $d(x,y)$ satisfies the strong triangle inequality. The following proposition claims that this is true.

Proposition 4. *Consider a set of points X, a vantage point $v\in X$, and a radius μ to define inside and outside sets X' and X'' as in* (17)*. We are given a query x_o. If the dissimilarity function $d(x,y)$ satisfies the ∞-triangle inequality* (4)*, we have that*

(∞-CI) *If $d(v,x_o)<\mu$, then, either v is a nearest neighbor or there exist a nearest neighbor in the inside set; i.e., $\hat{x}_o\cap(X'\cup v)\neq\emptyset$.*

(∞-CO) *If $\mu\leq d(v,x_o)$, then, either v is a nearest neighbor or there exist a nearest neighbor in the outside set; i.e., $\hat{x}_o\cap(X''\cup v)\neq\emptyset$.*

Proof. Consider first the case in which the condition $d(v,x_o)<\mu$ in (∞-CI) holds and let x'' be an arbitrary element of the outside set X''. As per the definition of the outside set, any point $x''\in X''$ is such that $\mu\leq d(v,x'')$. Writing this latter inequality followed by the ∞-triangle inequality (3) applied to the triangle (v,x_o,x''), we can write,

$$\mu\;\leq\;d(v,x'')\;\leq\;\max\big(\,d(v,x_o),\,d(x_o,x'')\,\big). \tag{36}$$

Because we are assuming that the condition $d(v,x_o)<\mu$ in (∞-CI) holds, we can also conclude that the maximum in the right hand side of (36) must be achieved by $d(x_o,x'')$. This is the only way in which the maximum can equal or exceed μ when $d(v,x_o)$ is strictly smaller than μ. We have therefore proven that for any point $x''\in X''$,

$$\mu\leq d(x_o,x''). \tag{37}$$

We also know that the distance $d(x_o,\hat{x}_o)$ between the query x_o and a nearest neighbor $\hat{x}_o$ cannot exceed the distance $d(v,x_o)$ between the query x_o and the vantage point v. Given that condition (∞-CI) is assumed to hold we also have that $d(v,x_o)<\mu$ and we can therefore write,

$$d(x_o,\hat{x}_o)\leq d(v,x_o)<\mu. \tag{38}$$

Contrasting (37) with (38) we conclude that it is impossible for a nearest neighbor to belong to the outside set X''. We must therefore have that either v is a nearest neighbor or there is a nearest neighbor in X'. Notice that this proof is analogous to the corresponding proofs of claims (CI) and (q-CI) of Propositions 2 and 3.

Consider now the case in which the condition $\mu\;\leq\;d(v,x_o)$ in (∞-CI) holds and let x' be an arbitrary element of the inside set X'. Further consider the distance $d(v,x_o)$ and apply the ∞-triangle inequality (3) to the triangle (v,x',x_o) to write,

$$d(v,x_o)\;\leq\;\max\big(\,d(v,x'),\,d(x',x_o)\,\big). \tag{39}$$

Suppose that the maximum in (39) is attained by $d(v,x')$. Under this assumption we have that $d(v,x_o)\;\leq\;d(v,x')$. Furthermore, because we assume that $x'\in X'$, we have according to (17), that $d(v,x')<\mu$. These facts together yield,

$$d(v,x_o)\;\leq\;d(v,x')\;<\;\mu \tag{40}$$

This is contradiction with the condition $\mu\;\leq\;d(v,x_o)$ in (∞-CI). Thus, it is impossible for the maximum in (39) to be attained by $d(v,x')$. The maximum must be therefore attained by $d(x',x_o)$ leading to the conclusion that,

$$d(v,x_o)\;\leq\;d(x',x_o). \tag{41}$$

Algorithm 1: Vantage Point (VP) Tree Construction

Input: Data set X and dissimilarity function d.

Output: Node $[p, \mu, \text{inside}, \text{outside}]$.

Function BuildVPTree
  if $X \neq \emptyset$ **then**
    Choose random vantage point $v \in X$
    node.$p \leftarrow v$
    node.$\mu \leftarrow \text{MEDIAN}_{x \in X}(d(x, v))$
    $X' \leftarrow \{x \in X : d(x, v) < \mu,\ x \neq v\}$
    $X'' \leftarrow \{x \in X : d(x, v) \geq \mu\}$
    node.inside $\leftarrow$ BuildVPTree(X', d)
    node.outside $\leftarrow$ BuildVPTree(X'', d)
    return node
  else
    return *null*
  end if

Figure 6: Construction of a Vantage Point (VP) tree. Nodes are of the form $[p, \mu, \text{inside}, \text{outside}]$. Here, p is a vantage point, μ its associated radius, "inside" is a node representing the inside set X' and "outside" is a node representing the outside set X''. All nodes $[v', \mu', \text{inside}', \text{outside}']$ in the inside subtree satisfy $d(v, v') < \mu$, and all nodes $[v'', \mu'', \text{inside}'', \text{outside}'']$ in the outside subtree satisfy $\mu \leq d(v, v'')$.

Since the nearest neighbor distance is $d(x_o, \hat{x}_o) \leq d(v, x_o)$, there is no point in X' whose distance to the query x_o is smaller than the distance between x_o and the vantage point v. There are now two possibilities: (i) The vantage point v is a nearest neighbor. (ii) The vantage point v is *not* a nearest neighbor. If v is a nearest neighbor the claim in (∞-CO) holds true as the vantage point v is contained in the nearest neighbor set. If v is *not* a nearest neighbor, (41) implies that there is no nearest neighbor in X'. The nearest neighbors must therefore be in X''.

Observe that this proof is *not* analogous to the proof of claims (CO) and (q-CO) of Propositions 2 and 3. Indeed, the claims itself are different. The reason for having a different argument is that in the case of ∞-metric spaces the inequality in (41) does not prevent the existence of nearest neighbors in X'. It implies the (weaker) claim that this is possible only if v itself is a nearest neighbor. This difference does not affect our ability to prune the set X' if our goal is to find *a* nearest neioghbor of x_o. □

Because of (∞-CI) and (∞-CO) each comparison with a vantage point in an ∞-metric space focuses the search on a branch of the VP tree. If the condition in (∞-CI) holds, we continue exploration of the inside branch X' and discard the outside set X''. We know that no nearest neighbors are in this set. If the condition in (∞-CO) holds, we continue exploration of the outside branch X'' and discard the inside set X''. In this case we know that there may be some nearest neighbors in X' if more than one exists, but none of them are closer to x_o than the vantage point v.

The remarkable property of ∞-metric spaces is that the conditions in (∞-CI) and (∞-CO) are complementary. One and exactly one of them hold. It follows from this observation that to find the nearest neighbor of a query x_o for dissimilarity functions that satisfy the strong triangle inequality is bounded by the depth of the VP tree.

Theorem 1. *Consider a dataset X, a query x_o, a dissimilarity function d satisfying the strong triangle inequality* (3) *and a vantage point tree $T(X)$ constructed by the recursive partition of X into vantage points and their corresponding inside and outside sets [cf.* (5)*]. The number of comparisons $c(x_o)$ needed to find a nearest neighbor of x_o in the set X is bounded by the depth $h(T(X))$ of the*

VP tree $T(X)$*,*

$$c(x_o) \leq h(T(X)). \tag{42}$$

Proof. Consider a VP-tree, constructed on X using Algorithm 1. Proposition 4 ensures that making one single comparison -38 or 37- for each vantage point will discard a set of points -either X' or X''- not containing the nearest neighbor of x_0. Notice that, by construction in 1, the maximum number of tims this can be done is limited by the recursion depth, which is indeed, the tree depth.

□

D Algorithms to Construct and Search Vantage Point Trees

Given a set of points X, Figure1 depicts how data can be organized to enable efficient search. As a concrete example, Algorithm 1 illustrates how a VPTree can be constructed by recursively partitioning a set of points X. This construction begins by randomly selecting a vantage point v (Step 3). Fixing v as one argument of the dissimilarity function d, allows to compute the median μ from the induced set distances $\{d(x, v)\}_{x \in X}$ (Step 5). This median serves as a radius that partitions X into an *inside* subset X' (Step 6) and *outside* subset X'' (Step 7). The function then recurses on subsets X' and X'', until no further split is possible, at which point the recursion halts and a leaf node is created (Step 12). This process results in a nested list of nodes $[p, \mu, inside, outside]$ storing a vantage point, the corresponding radius and the subsequent inside and outside nodes.

Algorithm 2: Searching Phase of the q-VPTree

Input: Query point x_o, parameter q and a node (initialized to the root). Nearest neighbor $\hat{x}_o$ is set to *null* and its distance τ to the query is initialized as ∞.

Output: Nearest neighbor $\hat{x}_o$ and its distance τ, after searching the subtree rooted at a node.

1: **Function** SearchVPTree$_q$(node, $\tau, \hat{x}_o, x_o$)
2: **if** node $\neq$ *null* **then**
3: $d \leftarrow d(x_o, \text{node}.p)$
4: **if** $d < \tau$ **then**
5: $\tau \leftarrow d$
6: $\hat{x}_o \leftarrow \text{node}.p$
7: **end if**
8: **if** $d^q < \text{node}.\mu^q - \tau^q$ **then**
9: $\hat{x}_o, \tau \leftarrow$ SearchVPTree(node.$inside, \tau, \hat{x}_o, x_o$)
10: **else if** $d^q \geq \text{node}.\mu^q + \tau^q$ **then**
11: $\hat{x}_o, \tau \leftarrow$ SearchVPTree(node.$outside, \tau, \hat{x}_o, x_o$)
12: **else**
13: $\hat{x}_o, \tau \leftarrow$ SearchVPTree(node.$inside, \tau, \hat{x}_o, x_o$)
14: $\hat{x}_o, \tau \leftarrow$ SearchVPTree(node.$outside\tau, \hat{x}_o, x_o$)
15: **end if**
16: **end if**
17: **return** $\hat{x}_o, \tau$

Figure 7: Nearest neighbour search in a q-VP tree. Each node is of the form $[p, \mu, \text{inside}, \text{outside}]$, where p is a vantage point and μ its associated radius. Given a query x_o, the algorithm maintains the current best candidate $\hat{x}_o$ and its distance $\tau = d(x_o, \hat{x}_o)$. At each visited node, it updates $(\hat{x}_o, \tau)$ and may prune inside or outside subtrees using the q-triangle inequality.

Notice, that the partition of X into inside and outside sets as defined in (17) is independent of the structure of the dissimilarity d. This means, we can build the sets X' and X'' for any dissimilarity dunction d. In particular, we can build these sets when d represents distances of points in a q-metric or ∞-metric space.

The specific properties of q-metric and ∞-metric spaces are leveraged when searching for nearest neighbors. This search phase is illustrated in Algorithm 2 for the q-metric case and Algorithm 3 for ∞-metric spaces. Given a node $[v, \mu, \texttt{inside}, \texttt{outside}]$, we first compute the distance from the query x_o to the node's vantage point p (Step 3). If this improves the current best distance τ (Step 4), then the candidate nearest neighbor $\hat{x}_o$ is updated (Steps 5–6). Next, the pruning rules of Propositions 3 and 4 are used to decide whether to visit the `inside` child (Step 8) or the `outside` child (Step 10), without discarding the true nearest neighbor. Although both algorithms share the same initial steps, their pruning differs. In the q-metric case, Algorithm 2 may need to visit both children (Step 12); this additional step is not present in Algorithm 3. The procedure recurses until a leaf is reached; it then returns the current best neighbor $\hat{x}_o$ and its distance to x_o, namely τ (Step 14).

Algorithm 3: Searching Phase of the ∞-VPTree

Input: Query point x_o, parameter q, and a node (initialized to the root). Nearest neighbor $\hat{x}_o$ is set to *null* and its distance τ to the query is initialized as ∞.

Output: Nearest neighbor $\hat{x}_o$ and its distance τ, after searching the subtree rooted at a node.

1: **Function** SearchVPTree$_q$(node, $\tau, x_o, \hat{x}_o$)$v \neq$ *null*
2: $d \leftarrow d(x_o, \text{node}.p)$
3: **if** $d < \tau$ **then**
4: $\tau \leftarrow d$
5: $\hat{x}_o \leftarrow \text{node}.p$
6: **end if**
7: **if** $d < \text{node}.\mu$ **then**
8: $\hat{x}_o, \tau \leftarrow$ SearchVPTree(*inside*, $\tau, \hat{x}_o, x_o$)
9: **else**
10: $\hat{x}_o, \tau \leftarrow$ SearchVPTree(*outside*, $\tau, \hat{x}_o, x_o$)
11: **end if**
12: **return** $\hat{x}_o, \tau$

Figure 8: Nearest-neighbour search in an ∞-VP tree. Each node is of the form $[p, \mu, \text{inside}, \text{outside}]$, where p is a vantage point and μ its associated radius. Given a query x_o, the algorithm maintains the current best candidate $(\hat{x}_o, \tau)$. At each visited node, it updates $(\hat{x}_o, \tau)$ and prunes to a single branch using the strong triangle inequality.

Remark 1 (Balanced VP trees)**.** A balanced VP node is one in which the number of points in the inside and outside sets differ by at most one, $||X'| - |X''|| \leq 1$. A balanced VP tree is one in which all nodes are balanced. If the VP tree is balanced its depth satisfies $h \leq \log_2(|X|) + 1$ and is the lowest possible among all VP trees of the set X. This is accomplished in Step 5 of the algorithm in Algorithm 1 where we set the radius to $\mu = \text{median}_X\, d(v, x)$. This is a good choice for a balanced tree because precisely half of the points in X are on each side of the median. The tree is not perfectly balanced, however, because we may have several points $x \in X$ with $d(v, x) = \mu$, which, according to (17), must be assigned to the outside set X''. Having several points with $d(v, x) = \mu$ almost never happens in metric and q-metric spaces but it does sometimes happen in ∞-metric spaces. Because of this, the depth of VP trees of samples of an ∞-metric space is (slightly) larger than the depth $h \leq \log_2(|X|) + 1$ of a balanced tree; see Figure 9. This figure shows that the worst-case complexity -namely, the tree depth- lies within a constant slack of logarithmic scaling, the latter matching the expected number of comparisons.

In this settings where logarithmic depth is reached, Theorem 1 guarantees logarithmic search complexity. Remarkably, this is not merely an upper bound: it is an exact characterization. That is, for every query, the search requires exactly a logarithmic number of comparisons -up to a $+1$ slack-, so that best- and worst-case coincide.

Remark 2. Note that in Step 2 we select v arbitrarily. Proper choice of v may have a significant effect in practice and the use of heuristic to select good vantage points is common Yianilos (1993). Commonly, the vantage point v is set to maximize the spread of the induced distances. In the original VP-tree work Yianilos (1993), this was done by considering higher moments of the distance distribution, such as the variance.

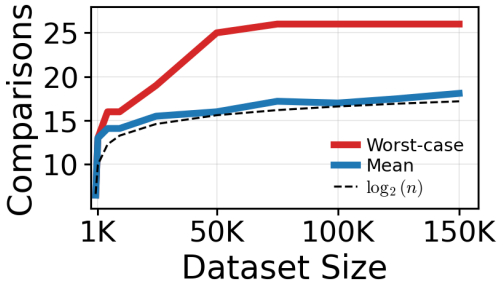


Figure 9: VP Tree search complexity on a ∞-metric space with $n \in \{100, 500, 1\text{K}, 5\text{K}, 10\text{K}, 50\text{K}, 100\text{K}, 150\text{K}\}$ points. The worst-case bound in comparisons corresponds to the depth of the tree.

E PROJECTION

In this Appendix, we expand on the theoretical properties exposed in Section 3. The proofs rely on the axioms (A1) and (A2), that are imposed on the Canonical Projection. These two requirements are enough to imbue the projection with beneficial properties on Nearest Neighbor search.

E.1 $P_q^\star$ EXISTS, IS UNIQUE, AND SATISFIES THE q-TRIANGLE INEQUALITY

The results in Section E.1 were presented in (Segarra et al., 2019) and are included here for completeness.

For a given q, the Canonical Projection $P_q^\star$ as defined in (11), is guaranteed to satisfy the q-triangle inequality. Moreover, it meets axioms of Projection A1 and Transformation A2.

Lemma 1 (Satisfaction of q-triangle inequality)**.** *The canonical Projection as defined in 11, satisfies the q-triangle inequality.*

Proof. To verify the q-triangle inequality, let $c_{xx'}$ and $c_{x'x''}$ be paths that achieve the minimum in (11) for $d_q(x, x')$ and $d_q(x', x'')$, respectively. Then, from the definition in (11), it follows that:

$$\begin{aligned} d_q(x, x'')^q &= \min_{c_{xx''}} \|c\|_q^q, \leq \|c_{xx'} \oplus c_{x'x''}\|_q^q, \\ &= \|c_{xx'}\|_q^q + \|c_{x'x''}\|_q^q \\ &= d_q(x, x')^q + d_q(x', x'')^q, \end{aligned} \tag{43}$$

where the inequality holds because the concatenated path $c_{xx'} \oplus c_{x'x''}$ is a valid (though not necessarily optimal) path between x and x'', while $d_q(x, x'')$ minimizes the q-norm across all such paths. □

Theorem 1 (Existence and uniqueness)**.** The canonical projection $P_q^\star$ satisfies Axioms A1 and A2. Moreover, if a q-metric projection P_q satisfies Axioms A1 and A2, it must be that $P_q = P_q^\star$.

Proof. We first prove that d_q is indeed a q-metric on the node space X. That $d_q(x, x') = d_q(x', x)$ follows from the fact that the original graph G is symmetric, and that the norms $\|\cdot\|_q$ are symmetric in their arguments for all q. Moreover, $d_q(x, x') = 0$ if and only if $x = x'$, due to the positive definiteness of the q-norms.

To see that the Axiom of Projection A1 is satisfied, let $M = (X, d) \in \mathcal{M}_q$ be an arbitrary q-metric space, and denote $(X, d_q) = P_q^\star(M)$, the output of applying the canonical q-metric projection. For any pair $x, x' \in X$, we have:

$$d_q(x, x') = \min_{c \in C_{xx'}} \|c\|_q \leq \|[x, x']\|_q = d(x, x'), \tag{44}$$

for all q, where the inequality comes from choosing the trivial path $[x, x']$ consisting of a single edge from x to x'.

Let $c^\star_{xx'} = [x = x_0, x_1, \ldots, x_\ell = x']$ be the path that achieves the minimum in the above expression. Using Lemma 1 we know d_q satisfies the q-triangle inequality:

$$d(x, x') \leq \left(\sum_{i=0}^{\ell-1} d(x_i, x_{i+1})^q \right)^{1/q} = \|c^\star_{xx'}\|_q = d_q(x, x'). \tag{45}$$

Substituting this into the previous inequality, we find that:

$$d(x, x') = d_q(x, x'). \tag{46}$$

Since x and x' were arbitrary, we conclude $d \equiv d_q$, hence $P^\star_q(M) = M$, as required.

To show that the Axiom of Transformation A2 holds, consider two graphs $G = (X, D)$ and $G' = (X', D')$, and a dissimilarity-reducing map $\varphi : X \to X'$. Let $(X, d_q) = P^\star_q(G)$ and $(X', d'_q) = P^\star_q(G')$ be the projected spaces.

For a pair $x, x' \in X$, let $c^\star_{xx'} = [x = x_0, x_1, \ldots, x_\ell = x']$ be a path achieving the minimum for $d_q(x, x') = \|c^\star_{xx'}\|_q$. Consider the image path in X':

$$P^\star_{\varphi(x)\varphi(x')} = [\varphi(x_0), \varphi(x_1), \ldots, \varphi(x_\ell)]. \tag{47}$$

Since φ is dissimilarity-reducing, we have:

$$D'(\varphi(x_i), \varphi(x_{i+1})) \leq D(x_i, x_{i+1}) \quad \text{for all } i. \tag{48}$$

Hence,

$$\|P^\star_{\varphi(x)\varphi(x')}\|_q \leq \|c^\star_{xx'}\|_q = d_q(x, x'). \tag{49}$$

Now, since $d'_q(\varphi(x), \varphi(x'))$ is defined as the minimum over all such paths in X', we get:

$$d'_q(\varphi(x), \varphi(x')) \leq \|P^\star_{\varphi(x)\varphi(x')}\|_q \leq d_q(x, x'), \tag{50}$$

which completes the proof of Axiom A2. □

E.2 Canonical Projection preserves the Nearest Neighbor

We canonical projection maps any graph to a q-metric space, as showed in Lemma 1. Through Section C, we exposed the advantages of searching nearest neighbors in such metric spaces. It remains to be proved, that this search is not only efficient but also accurate, i.e., that the projection preserves the nearest neighbor.

Proposition 5. *Given a graph $G = (X, D)$ and a query x_o, define $H = (Y, E)$ with $Y = X \cup \{x_o\}$ and let $(Y, E_q) = P^\star_q(H)$ be the projected graph. Then, the set of nearest neighbors satisfies*

$$\hat{x}_o \;\equiv\; \operatorname*{argmin}_{x \in X} E(x_o, x) \;\subseteq\; \operatorname*{argmin}_{x \in X} E_q(x_o, x). \tag{51}$$

Proof. We prove the theorem by showing that any *admissible* projection i.e, any map satisfying axioms (A1) and (A2), *cannot* result in larger dissimilarities for any pair of points. We further show that nearest neighbor dissimilarities are preserved by the canonical projection. The combination of these two facts yields (51). A diagram to better follow the proof is provided in Figure 10.

To prove that dissimilarities are not increased, construct an arbitrary graph $\tilde{G} = (\tilde{X}, \tilde{E})$ made up of two nodes $\tilde{X} = \{\tilde{x}, \tilde{y}\}$, along with a unique edge connecting them $\tilde{E}(\tilde{x}, \tilde{y})$. We also enforce this graph to have its unique edge weight, equal to the distance between the nearest neighbor $\hat{x}_o$ and the query point x_o, i.e $\tilde{E}(\tilde{x}, \tilde{y}) = E(x_o, \hat{x}_o)$. In addition, we trivially set any reflexive edge as null, namely $\tilde{E}(\tilde{x}, \tilde{x}) = \tilde{E}(\tilde{y}, \tilde{y}) = 0$.

Notice that $(\tilde{X}, \tilde{E})$ has only two vertices. As such, it is a trivial q-metric space: no triplet can be formed, and the q-triangle inequality holds by default. As a direct consequence of axiom (A1), any

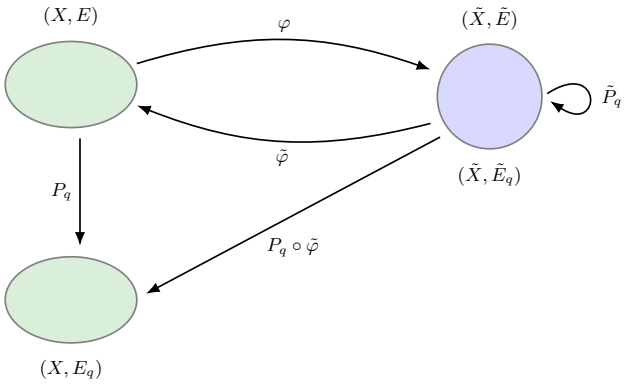


Figure 10: Canonical projection preserves the nearest neighbor. Starting from the original graph (X, E), we construct the graph $(\tilde{X}, \tilde{E})$ together with two dissimilarity-reducing maps φ and $\tilde{\varphi}$ between them. Applying any *admissible* map P_q and $\tilde{P}_q$ yields the respective q-metric spaces (X, E_q) and $(\tilde{X}, \tilde{E}_q)$. Note that, since $(\tilde{X}, \tilde{E})$ is already a trivial q-metric space, the projection leaves it unchanged (A1).

admissible projection acting on $(\tilde{X}, \tilde{E})$ leaves the space unchanged. Therefore, for the remainder of the proof, $(\tilde{X}, \tilde{E})$ will be used indistinguishably as $(\tilde{X}, \tilde{E}_q)$.

Let us now map the original graph G to the new one as in Figure 10,

$$\varphi(x) = \begin{cases} \tilde{x} & \text{if } x = x_o, \\ \tilde{y} & \text{otherwise.} \end{cases} \tag{52}$$

Since the only existing edge in $\tilde{G}$ attains the minimum possible value, namely $E(\hat{x}_o, x_o)$, every edge of the original graph G that is incident to x_o will be mapped to a smaller dissimilarity by (52). On the other hand, edges that are not incident to x_o, are mapped to the self-loop $\tilde{E}(\tilde{y}, \tilde{y}) = 0$. Because we can assume E to be a *positive*[3] dissimilarity, this also yields a dissimilarity reduction. Therefore, we can consider φ a dissimilarity-reducing map. Consequently, for all $x \in X$,

$$E(x_o, x) \geq \tilde{E}(\varphi(x_o), \varphi(\hat{x}_o)) = E(x_o, \hat{x}_o) \tag{53}$$

Now we can invoke axiom (A2), which ensures the satisfaction of the previous inequality also in the projected spaces,

$$E_q(x_o, x) \geq \tilde{E}_q(\varphi(x_o), \varphi(\hat{x}_o)) = E(x_o, \hat{x}_o). \tag{54}$$

While this may appear to conclude the proof, it does not. Up to this point, we've shown that any admissible map cannot decrease edge weights below the original nearest-neighbor distance. It remains to prove that this nearest-neighbor distance is in fact preserved by any admissible map. With this purpose, we define the following map from $\tilde{G}$ to the original graph G, illustrated in Figure 10,

$$\begin{aligned} \tilde{\varphi}(\tilde{x}) &= x_o, \\ \tilde{\varphi}(\tilde{y}) &= \hat{x}_o. \end{aligned} \tag{55}$$

This map is a bijection sending $\tilde{E}(\tilde{x}, \tilde{y})$ to $E(x_o, \hat{x}_o)$. Consequently, we can conclude that $\tilde{\varphi}$ is a trivial dissimilatity-reducing map,

$$E(x_o, \hat{x}_o) = \tilde{E}(\tilde{x}, \tilde{y}) \geq E(\tilde{\varphi}(\tilde{x}), \tilde{\varphi}(\tilde{y})) = E(x_o, \hat{x}_o). \tag{56}$$

We then proceed as with the previous case, invoking Axiom (A2), which preserves the inequality in the projected space,

$$E(x_o, \hat{x}_o) = \tilde{E}_q(\tilde{x}, \tilde{y}) \geq E_q(\tilde{\varphi}(\tilde{x}), \tilde{\varphi}(\tilde{y})) = E_q(x_o, \hat{x}_o). \tag{57}$$

[3] Notice that by definition, dissimilarity measures are lower bounded Muñoz (2012), therefore, it is always possible to obtain a positive version of them.

Therefore, the projected distance between x_o and its nearest neighbor $\hat{x}_o$ coincides with the original distance $E(x_o, \hat{x}_o)$, and thus attains the lower bound,

$$E_q(x, x_o) \geq E(x_o, \hat{x}_o) = E_q(x_o, \hat{x}_o) \tag{58}$$

The proof was made with an arbitrary *admissible* projection, taking into account that the canonical projection is the only admissible map (Theorem 2), we conclude the proof. □

Having a preservation of the nearest neighbors makes the projection a valid candidate for search. The previous result can be further extended as follows,

Corollary 1. *Under the setting of Proposition 5, if $q < \infty$, the nearest neighbor is preserved with the following equality:*

$$\hat{x}_o \equiv \operatorname*{argmin}_{x \in X} E(x, x_o) = \operatorname*{argmin}_{x \in X} E_q(x, x_o). \tag{59}$$

Proof. The proof will follow by contradiction.

Assume there exists $z \in X$ a nearest neighbor in the projected space E_q, such but not in the original space E, i.e,

$$z \notin \operatorname*{argmin}_{x \in X} E(x, x_o) \quad \text{and} \quad z \in \operatorname*{argmin}_{x \in X} E_q(x, x_o). \tag{60}$$

Theorem 2, demands E_q to be obtained with the canonical projection operator P_q^*, which is the q-shortest path between points. Thus, there exist a shortest chain $c^* = [x_o, \ldots, z]$ linking x_o to z such that

$$E_q(x_o, z) = ||c^*||_q = \min_{c \in \mathcal{C}_{x_o z}} \ell_q(c). \tag{61}$$

However, since $z \notin \operatorname{argmin}_{x \in X} E(x, x_o)$, the one-hop chain $[x_o, z]$, is not a valid minimum path,

$$E(x_o, z) = \sqrt[q]{d(x_o, z)^q} = \ell_q([x_o, z]) \neq \min_{x \in X} d(x_o, x). \tag{62}$$

Without loss of generality, assume there exists at least one intermediate vertex in the optimal path, i.e $\exists\, x_1 \neq z$ such that $c^* = [x_o, x_1, \ldots, z]$. Since E is a positive dissimilarity (see footnote 3), ℓ_q is strictly increasing,

$$\ell_q([x_o, x_1, \ldots, z]) > \ell_q([x_o, x_1]) \geq \min_{c \in \mathcal{C}_{x_o z}} \ell_q(c). \tag{63}$$

This means that an intermediate node must exist; yet in that case it would induce a projected distance smaller than that of z, contradicting the assumption that z is a nearest neighbor. Therefore,

$$z \notin \operatorname*{argmin}_{x \in X} E_q(x, x_0). \tag{64}$$

□

Notice that Proposition 5 formulates nearest neighbor preservation as an inclusion. In contrast, Proposition 1 characterizes preservation via a strict equality between nearest-neighbor sets. This distinction matters: when projecting with $q = \infty$, points that were not nearest neighbors in the original space may become nearest neighbors in the projected space, as illustrated in Example 1. We refer to such points as *spurious neighbors*. From the perspective of vector search, this behavior is undesirable, it enlarges the candidate set, detracting from the advantages discussed in Section C.

Example 1. Let $\{x_1, x_2, x_3\} = X$ be three points equipped with the original distances

$$E(x_1, x_2) = 3, \quad E(x_2, x_3) = 2,$$
$$E(x_1, x_3) = 5. \tag{65}$$

Clearly $E(x_1, x_2) < E(x_1, x_2)$, so the unique nearest neighbor of x_1 in $G = (\{x_1, x_2, x_3\}, E)$ is x_2. We now enforce the strong triangle inequality by projecting:

$$E_\infty(x_2, x_3) \;=\; \max\Big[\{E_\infty(x_1, x_2),\, E_\infty(x_2, x_3)\Big] \;=\; 3. \tag{66}$$

while leaving $E_\infty(x_1, x_2) = 3$ and $E_\infty(x_2, x_3) = 2$. Under the new distance E_∞, one finds

$$E_\infty(x_1, x_2) = 3 \quad \text{and} \quad E_\infty(x_1, x_3) = 3. \tag{67}$$

so x_1 is equidistant from x_2 and x_3. As showed in Figure 11, the original nearest neighbor x_2 is no longer uniquely closest, i.e, any nearest-neighbor search on $G' = (\{x_1, x_2, x_3\}, E_\infty)$ may return x_3 instead of x_2, demonstrating that ultra-projecting a metric can extend original nearest neighbor set.

Proposition 1 guarantees the nearest neighbor to be preserved under the Canonical Projection. Therefore, the Nearest Neighbor Search problem in the transformed space is a relaxation of the original problem. Although this relaxation provides speedup, it can also make the problem more difficult if many solutions are added to the optimal set. This problem is addressed in Section F.5, by enriching the VPTree with a two-stage search.

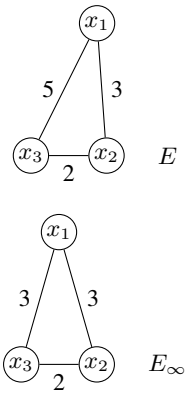


Figure 11: The Canonical Projection with $q = \infty$, added nearest neighbors to the original set.

E.3 Efficient Approximation of P_q^*

Calculating the canonical projection $P_q^\star$ over the training points can incur prohibitive computing times. Algorithms 4 and 5 describe the computation of the canonical projection, inspired by a matrix-form variant of the classical Floyd-Warshall algorithm Floyd (1962); Warshall (1962). These procedures are highly paralellizable and can be efficiently executed on GPUs, which significantly reduces the projection time. Nevertheless, computing the canonical projection for n points still requires $O(n^3)$ operations in the worst case, which remains prohibitive for large datasets.

Instead of computing the shortest q-norm path in the complete graph $D \in \mathbb{R}^{n\times n}$, we restrict the search to only consider the k-nearest neighbor graph. Doing this reduces the complexity of computing all pairwise distances from $O(n^3)$ to a factor $O(n^2k^2)$. Moreover, Groisman et al. (2022) showed that when the original space is euclidean and $k \approx log(n)$, the projected distances remain unchanged with high probability, with worst-case complexity $O(n^2log^2(n))$.

Algorithm 4: Canonical Projection P_q^*

Input: Distance matrix $D \in \mathbb{R}_+^{n\times n}$ and parameter q.

Output: Projected matrix $P_q^*(D)$.

1: **Function** $P_q^*(D, q)M \leftarrow D^q$
2: $\mathbf{1} \leftarrow (1, \dots, 1) \in \mathbb{R}^n$
3: **for** $i = 1 : n$ **do**
4: $C_i \leftarrow M[:, i]\mathbf{1} + \mathbf{1}^\top M[i, :]$
5: $M \leftarrow \min(M, C_i)$
6: **end for**
7: **return** $M^{1/q}$

Algorithm 5: Canonical Projection P_∞^*

Input: Distance matrix $D \in \mathbb{R}_+^{n\times n}$.

Output: Projected matrix $P_\infty^*(D)$.

1: **Function** $P_\infty^*(D)M \leftarrow D$
2: $\mathbf{1} \leftarrow (1, \dots, 1) \in \mathbb{R}^n$
3: **for** $i = 1 : n$ **do**
4: $C_i \leftarrow \max\big(M[:, i]\mathbf{1},\ \mathbf{1}^\top M[i, :]\big)$
5: $M \leftarrow \min(M, C_i)$
6: **end for**
7: **return** M

Figure 12: Algorithms for Canonical Projection computation. Complexity: $O(n^3)$. The $\max$ and $\min$ operators are applied element-wise to each element of the matrix.

These modifications are incorporated in Algorithms 6 and 7. Algorithm 6 begins by raising each entry of the distance matrix D to the power q (Step 2), thereby producing edge weights consistent with the q-path metric. However, this operation does is not performed in Algorithm 7, since in ultrametric spaces, the Canonical Projection only accounts for the maximum value along a path.

Next, for each node x_i, we restrict attention to its k nearest neighbors in the original space via the adjacency matrix A. Concretely, we introduce a masking matrix S_A (Step 3), which is later added to the working distance matrix M (Step 4), in order to render non-neighbor edges effectively unreachable, i.e., assigned a prohibitively large cost.

Algorithm 6: Sparse Canonical Projection P_q^*

Input: Distance matrix D, adjacency matrix A, number of pivots l, parameter q.

Output: Projected approximate matrix $P_q^*(D; A, l)$.

1: **Function** $P_q^*(D, A, l, q) M \leftarrow D^q$
2: $(S_A)_{ij} \leftarrow \begin{cases} 0, & \text{if } A_{ij} = 1, \\ \infty, & \text{otherwise} \end{cases}$
3: $M \leftarrow M + S_A$
4: $\mathbf{1} \leftarrow (1, \ldots, 1)^\top \in \mathbb{R}^n$
5: **for** $t = 1 : l$ **do**
6: **for** $i = 1 : n$ **do**
7: $C_i \leftarrow M[:, i]\mathbf{1}^\top + \mathbf{1}M[i, :]$
8: $M \leftarrow \min\big(M,\ C_i + S_A\big)$
9: **end for**
10: **end for**
11: **return** $M^{1/q}$

Algorithm 7: Sparse Canonical Projection P_∞^*

Input: Distance matrix D, adjacency matrix A, number of pivots l.

Output: Projected approximate matrix $P_\infty^*(D; A, l)$.

1: **Function** $P_\infty^*(D, A, l) M \leftarrow D$
2: $(S_A)_{ij} \leftarrow \begin{cases} 0, & \text{if } A_{ij} = 1, \\ \infty, & \text{otherwise} \end{cases}$
3: $M \leftarrow M + S_A$
4: $\mathbf{1} \leftarrow (1, \ldots, 1)^\top \in \mathbb{R}^n$
5: **for** $t = 1 : l$ **do**
6: **for** $i = 1 : n$ **do**
7: $C_i \leftarrow \max\big(M[:, i]\mathbf{1}^\top,\ \mathbf{1}M[i, :]\big)$
8: $M \leftarrow \min\big(M,\ C_i + S_A\big)$
9: **end for**
10: **end for**
11: **return** M

Figure 13: Sparse Canonical Projection using adjacency A (restricting updates to k local neighbors) and early stopping after l pivots (paths of length $\leq l$). Complexity: $O(lnk^2)$.

At each iteration, for every node x_i and each of its neighbors x_j, we update the path cost $D(x_i, x')$ to a third node x', to remain the minimum possible cost. In the q-metric case, Algorithm 6 combines previously q-powered distances via addition (Step 5), as in

$$M(x_i, x') \leftarrow \min\left\{M(x_i, x'),\ M(x_i, x_j)^q + M(x_j, x')^q\right\}. \tag{68}$$

In contrast, Algorithm 7 applies an entrywise $\max$ update (Step 5), which corresponds to the canonical projection in ultrametric spaces.

Notice that, without loss of generality (see footnote (3)), we may assume a *positive* dissimilarity matrix $D \in \mathbb{R}^{n \times n}$. Consequently, raising D to any power q yields a positive matrix as well. This observation is relevant when applying $\min$ or $\max$ operators, since it preserves the interpretation of a minimum path.

Finally, the result is root-transformed to return to the original distance scale (Step 12):

$$d_q^*(x_i, x_j) \approx \left(M(x_i, x_j)\right)^{1/q}. \tag{69}$$

In addition, paths may be truncated to length l, so that at most l intermediate components can participate in any path (Step 6). This additional masking step reduces the runtime to $O(ln\log^2(n))$. At present, however, we do not have theoretical guarantees on the approximation error it induces on the projected distances. Still, since the $q = \infty$ case depends only on the maximum entry along a path, it is plausible that limiting the path length yields a small approximation error in practice.

After the aforementioned complexity reduction, the approximated algorithm remains highly paralellizable in GPU and now benefits from sparse matrix multiplications due to the sparsity of A.

F EXPERIMENTS & RESULTS

F.1 EXPERIMENTAL SETTINGS

Vector Datasets

The experiments were conducted on two commonly used datasets and one with distinct characteristics. We split the data randomly, using 80% for indexing and the remaining 20% is used as queries. We provide a summary of the datasets below, additional details can be found in the references provided.

- **(Xiao et al., 2017)Fashion-MNIST-784 Euclidean**: Image samples from the Fashion-MNIST collection, each flattened into a 784-dimensional vector.
- **(Pennington et al., 2014)GloVe-200 Cosine**: Text embeddings extracted from the GloVe (Global Vectors for Word Representation) model, each of dimension 200.
- **(Oliva and Torralba, 2001)Gist-960 Euclidean**: A set of real-valued GIST descriptors of natural scene images, each represented as a 960-dimensional vector. A GIST descriptor is a global, low-dimensional representation of an image's spatial envelope.
- **(Newman, 2008)NYTimes-256 Cosine**: Document embeddings from the UCI NYTimes Bag-of-Words corpus. We convert term-count vectors to TF–IDF and apply truncated SVD to 256 dimensions; evaluation uses cosine (angular) distance.
- **(Rácz et al., 2005)Kosarak-41,000 Jaccard**: A real-world sparse binary transaction dataset derived from click-stream data of a Hungarian news portal. Each transaction is represented as a 41,000-dimensional vector.
- **(Babenko and Lempitsky, 2016)Deep1B-96 Euclidean**: A billion-scale collection of CNN-based image descriptors which are later PCA-reduced to 96 dimensions.

Dissimilarities

Apart from searching using the euclidean distance, as customary for Glove and FashionMNIST, to showcase how our approach can accommodate arbitrary dissimilarity functions, we also evaluate our method and results searching with other dissimilarities on the same datasets. The dissimilarities used are specified in Table 2.

Table 2: Distance and Dissimilarity Metrics used.

Metric	**Formula**
Euclidean Distance	$d(x,y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$
Manhattan Distance	$d(x,y) = \sum_{i=1}^{d} \lvert x_i - y_i \rvert$
Cosine Dissimilarity	$d(x,y) = 1 - \frac{x \cdot y}{\lVert x \rVert \, \lVert y \rVert}$
Correlation	$d(x,y) = 1 - \frac{(x-\bar{x})\cdot(y-\bar{y})}{\lVert x-\bar{x} \rVert \, \lVert y-\bar{y} \rVert}$

Metrics

Approximate search algorithms are evaluated along two key dimensions: retrieval quality and search efficiency.

Search efficiency is assessed primarily by the number of comparisons needed to retrieve a result for a given query. Unlike throughput, this metric is agnostic to implementation and hardware, offering a fair basis for comparing algorithmic efficiency.

- **Number of Comparisons**: Every time the q-Metric VP-Tree visits a node. Reflects the computational cost related to search speed.
- **Queries Per Second (QPS)**: Measures the throughput of the algorithm, indicating how many queries can be processed per second under the current configuration.

Retrieval quality is evaluated using metrics that capture not just whether relevant results are retrieved, but how well their ordering is preserved. In particular, we rely on Recall@k and Rank Order, which – unlike recall – penalizes deviations in the relative ranking of retrieved results. This is crucial in downstream tasks such as recommendation, where the order of results matters. The metrics used in our evaluation are summarized below:

- **(Rácz et al., 2005)Kosarak-41,000 Jaccard**: A real-world sparse binary transaction dataset derived from click-stream data of a Hungarian news portal. Each transaction is represented as a 41,000-dimensional vector.
- **(Babenko and Lempitsky, 2016)Deep1B-96 Euclidean**: A billion-scale collection of CNN-based image descriptors which are later PCA-reduced to 96 dimensions.
- **RankOrder@k**: Measures how well the approximate method preserves the original ordering of the true nearest neighbors. Let $\mathcal{N}_k^{\text{true}}(y)$ be the true k-nearest neighbors of a query y, and $\mathcal{N}_k^{\text{approx}}(y) = \{x_1, \ldots, x_k\}$ the corresponding approximate result. Let $\pi(x_i, \mathcal{N}_k^{\text{true}}(y))$ denote the position of x_i in the true result $\mathcal{N}_k^{\text{true}}(y)$ (or $k+1$ if not found). The metric is defined as:

  Absolute RankOrder@k

  $$(y) = \sum_{i=1}^{k} \left| i - \pi(x_i, \mathcal{N}_k^{\text{true}}(y)) \right| \cdot \frac{1}{k} \tag{70}$$

  Lower values indicate better rank preservation, with 0 being optimal.

  In addition, a variation of the rank order that takes into account the total number of points in the dataset is also used:

  Relative RankOrder@k

  $$(y) = \sum_{i=1}^{k} \left| i - \pi(x_i, \mathcal{N}_k^{\text{true}}(y)) \right| \cdot \frac{100}{nk} \tag{71}$$

  where n is the size of the indexed points, i.e $|X|$. This measure expresses rank order but now as a percentage of the points available for retrieval.

- **Recall@k**: Measures the proportion of true k-nearest neighbors that are successfully retrieved by the approximate method. Let $\mathcal{N}_k^{\text{true}}(y)$ be the true k-nearest neighbors of a query y, and $\mathcal{N}_k^{\text{approx}}(y)$ the corresponding approximate result. The recall is defined as:

  $$\texttt{Recall@k}(y) = \frac{|\mathcal{N}_k^{\text{true}}(y) \cap \mathcal{N}_k^{\text{approx}}(y)|}{k} \tag{72}$$

  This metric ranges from 0 to 1, where 1 indicates that all true neighbors were retrieved. It reflects the *coverage* of the ground-truth neighbors in the approximate result.

  Setup

  Experiments were run on a workstation equipped with a 64-core (128-thread) AMD CPU, 256GB of system RAM, and two NVIDIA RTX A5000 GPUs with 24GB of VRAM each. This configuration provides sufficient CPU parallelism and GPU acceleration to support the projection step in Algorithms 6 and 7, as well as the embedding-model training performed in the experiments.

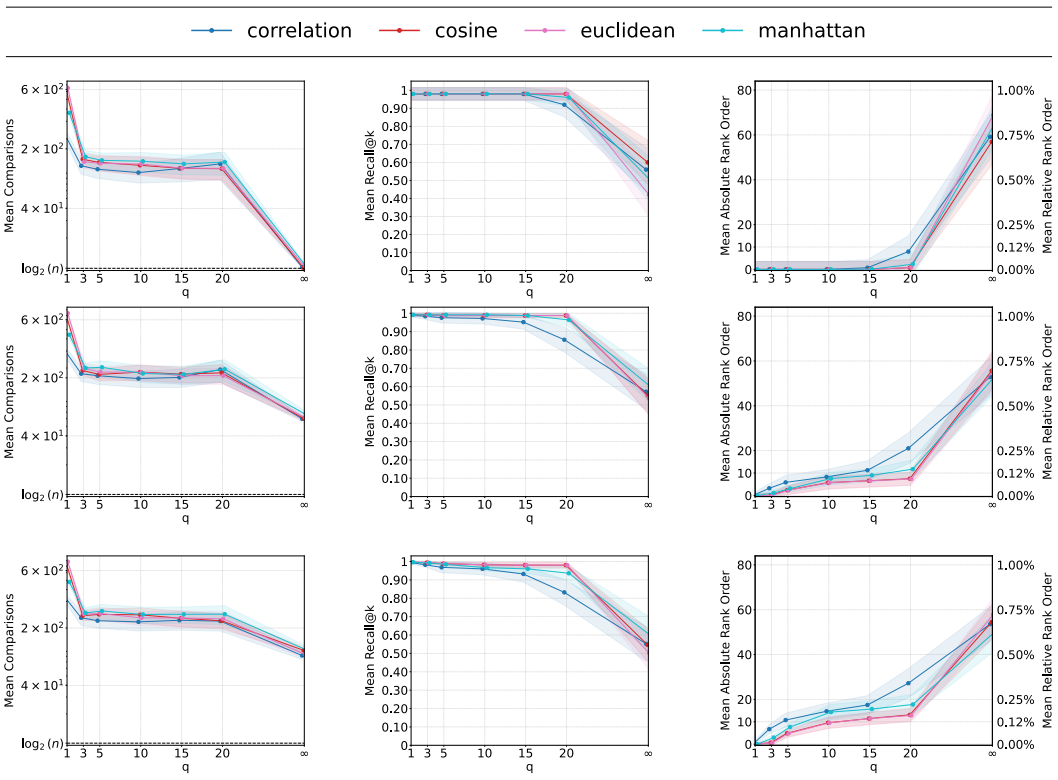
F.2 Canonical Projection $P_q^\star$

We validate the theoretical properties of searching in q-metric and ultrametric spaces presented in Section 2. In particular, we confirm our complexity claims (C2), the preservation of nearest neighbors, and the stability of rank order under projection (C3).

In order to do so, we use the Canonical Projection $P_q^\star$ presented in Section 3 to project distances imposing q-metric structure on FashionMNIST and GloVe. For these two datasets we use four

different dissimilarities. After projecting dissimilarities, we search using the q-metric VP tree as described in Appendix C.

Figures 14a and 14b (first row), show that the number of nodes visited during search decreases monotonically with increasing q, reaching the theoretical minimum of $\log_2(n)$ at $q = \infty$. This directly confirms Theorem 1 and supports claim (C2). Additionally, a rank order of zero across all queries–for a wide range of q values (excluding $q = \infty$) and across all dissimilarities– confirms claim and (C3), as established in Lemma 1 and Proposition 1. At the same time, this nearest neighbor preservation is also observed at recall, showing perfect matches at $k = 1$ for moderate q values.



(a) $n = 1{,}000$ points of Fashion-MNIST

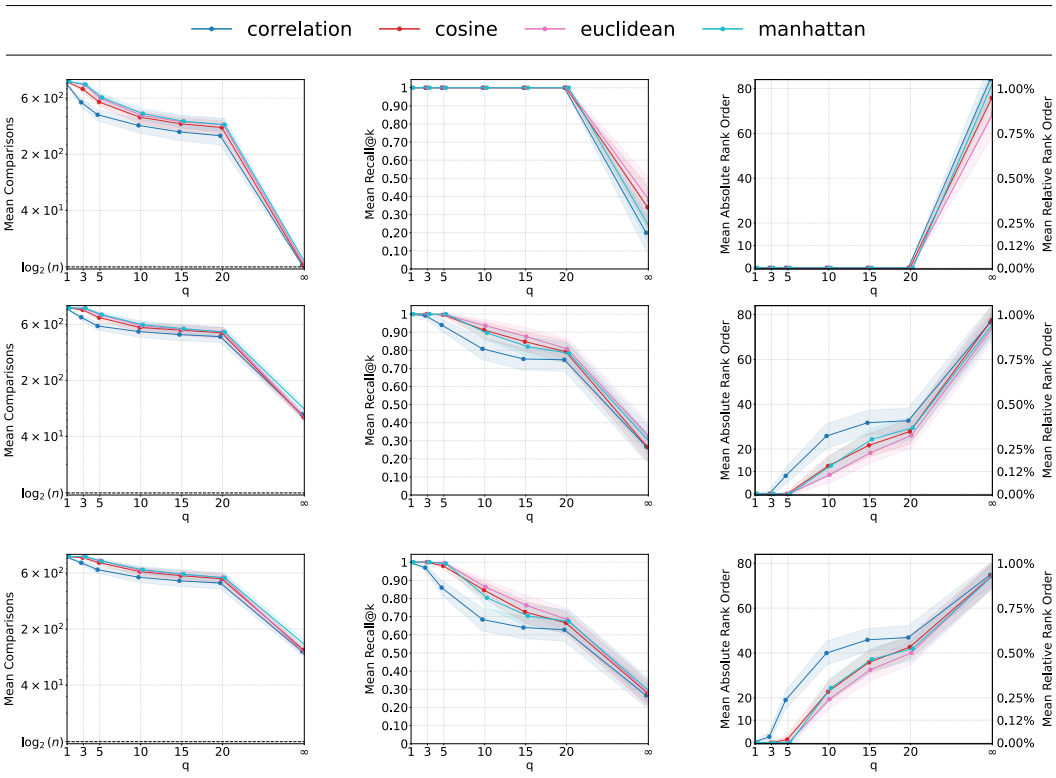
Figure 14: Number of comparisons and rank order across different dissimilarities when searching after applying Canonical Projection when a query point is added (E_q). The search was performed with a q-VPTree. Solid lines denote the mean and shading the standard deviation computed across queries. Each row shows results for k-nearest neighbors, with $k = 1, 5, 10$ from top to bottom.

At $q = \infty$, the projection may introduce spurious optima not included in the original nearest neighbor set. Although the original nearest neighbours are still a solution in the transformed space, in practice we observe that the spurious optima at $q = \infty$ indeed affect accuracy. A similar effect occurs for large q, as distances between points can become artificially close, imitating this behavior.

Although our theoretical guarantees focus on the $k = 1$ nearest neighbor case, an empirical preservation of locality observed in Figures 14a and 14b. We evaluate this by searching for $k = 5$ and $k = 10$ neighbors using the projected distances, as shown in the second and third rows of Figures 14a and 14b. While the number of comparisons does not decrease as rapidly as in the $k = 1$ case, the method consistently yields improvements across all values of q.

F.3 Approximating the canonical projection with $\Phi(x; \theta^\star)$

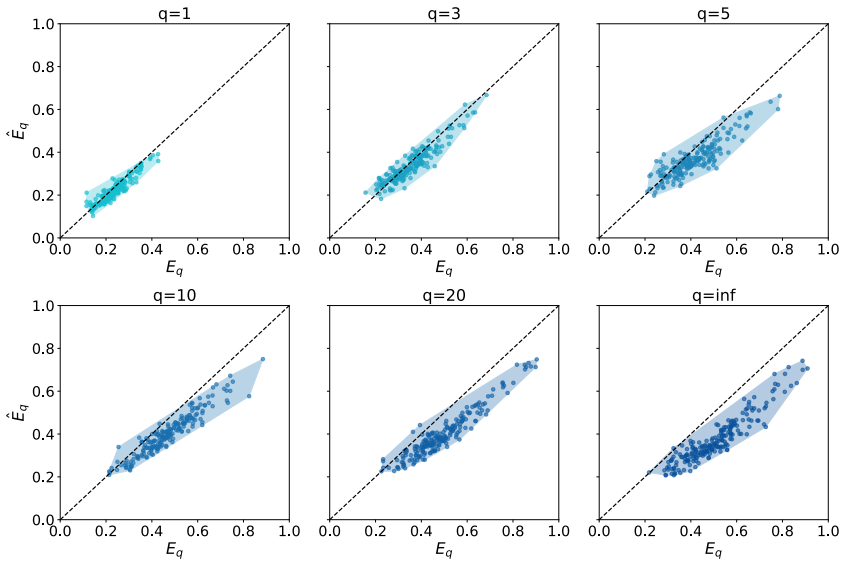
We analyze how well the learned distances $\hat{E}_q(x, x') = \|\Phi(x; \theta^\star) - \Phi(x'; \theta^\star)\|$, described in Section 4, reproduce the properties of the true q-metric distances $E_q(x, x')$.



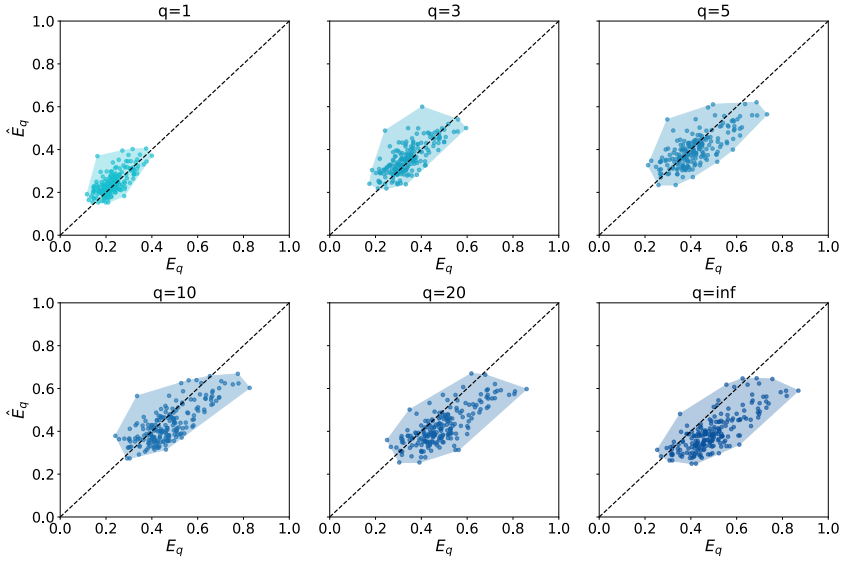
(b) $n = 1{,}000$ points of GloVe

Figure 14: Number of comparisons and rank order across different dissimilarities when searching after applying Canonical Projection when a query point is added (E_q). The search was performed with a q-VPTree. Solid lines denote the mean and shading the standard deviation computed across queries. The k-nearest neighbors are listed from top to bottom for $k = 1, 5, 10$

Predicted and ground-truth distances in Figure 15 exhibit similar distributions for training and testing, albeit with a clear generalization gap. As q increases, the approximation error increases accordingly. The model architecture is the multilayer perceptron shown in Table 3; an Optuna study Akiba et al. (2019) was used to select the hyperparameters of each experiment, including the number of layers, intermediate widths, and the final embedding dimension.



(a) Train



(b) Test

Figure 15: Distribution of the distance to Nearest Neighbor. In the X-axis E_q depicts distance values obtained after projecting when a query point is added. The Y-axis shows the learned approximation of the canonical projection $\hat{E}_q$. The dashed line represents the perfect match between projected and approximated $x = y$.

F.4 Searching with $\phi_q(\cdot, \theta^\star)$

Table 3: Embedding network (MLP) architecture. Optuna is used to tune the number of blocks L and their hidden width.

Model	**EmbNet**
Block	Linear $\rightarrow$ GELU $\rightarrow$ Dropout
Output layer	Linear

In this section we demonstrate that the speedup observed in the exact projection experiments of Section F.2 is also attained when replacing the Canonical projection with the learned map $\phi_q(\cdot, \theta^\star)$, at the cost of small errors in search results. The final ∞-Search framework is summarized in Figure 16.

Figures 17a and 17b show a consistent reduction in the number of comparisons as q increases, mirroring the trend observed in the exact projection experiments. This reduction is accompanied by a moderate increase in rank error and a decrease in recall, suggesting that the retrieved neighbors remain close but are not always exact. Nevertheless, cases with recall above 0.9 still yield substantial speedups. Since recall captures only exact matches and the method guarantees order preservation only for the 1-nearest neighbor, rank order can provide a complementary view of performance.

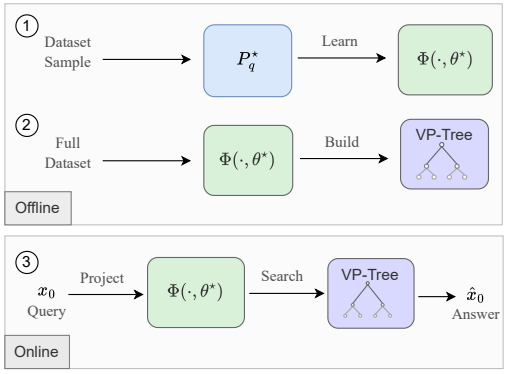


Figure 16: Infinity Search pipeline. Offline (up): Dataset samples are projected into a q-metric space using the canonical projection, and used to fit an embedding projection (1). Then, the dataset is transformed with the learned projection to build the VP-tree (2). Online (down): A query x_o is transformed with the learned projection and then search is conducted using the VP-tree index (3).

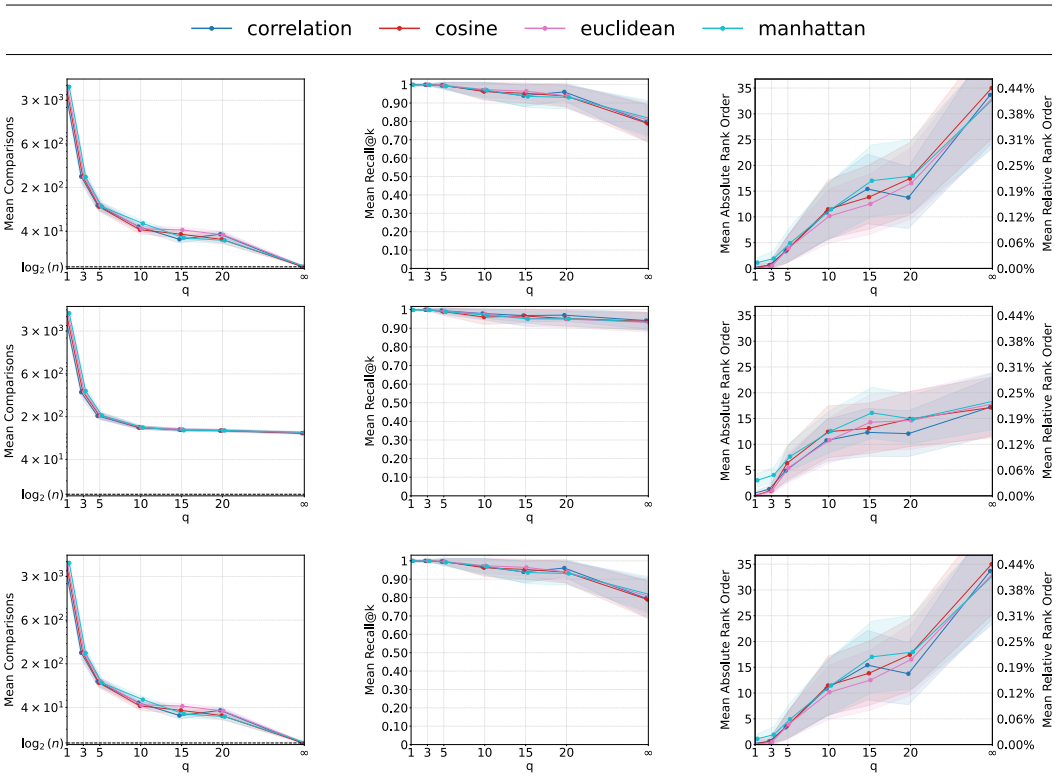
Despite the degradation of the approximation quality for high q values, the retrieved neighbors remain within the top 80 nearest, which corresponds to less than 1% of the indexed dataset. When higher precision is required, setting $q = 10$ can yield a two-orders-of-magnitude speedup (Figure 17a) while returning neighbors ranked around 10th, corresponding to a relative error close to 0.12%. These results demonstrate that the learned embeddings preserve local structure to a satisfactory extent.

The results also extend to k-nearest neighbor search with $k > 1$. While the reduction in comparisons is less pronounced than for $k = 1$, the method maintains a consistent speedup across values of q. Rank Order remains low, and in some cases improves as k increases, suggesting that the learned map preserves small-scale neighborhood structure reasonably well even on these datasets.

Examples of retrieval results were also generated. For each dataset, queries were selected from varied categories. In the Fashion-MNIST case (Figure 18), each panel shows the original query image, its true nearest neighbor, and the result returned by Infinity Search. While the exact nearest neighbor was not retrieved in some cases—such as those involving sandals or sneakers—the returned items consistently belonged to the same category as the query. For GloVe-200 text embeddings, Figure 18b includes several exact matches, as well as examples where the retrieved word preserved the semantic meaning of the query.

F.5 Two-Stage Infinity Search

Although Proposition 12 ensures preservation of the nearest neighbor structure, it also notes that the projected nearest neighbor may not remain unique. This ambiguity can lead to mismatches during retrieval. The effect appears in theoretical settings, as seen in Figures 14a and 14b, where rank order



(a) $n = 10{,}000$ points of Fashion-MNIST

Figure 17: Number of comparisons and rank order when searching after approximating the Canonical Projection $\hat{E}_q$, with the learned map $\Phi(x;\theta)$. Solid lines denote the mean and shading the standard deviation computed across queries. The k-nearest neighbors are listed from top to bottom for $k = 1, 5, 10$

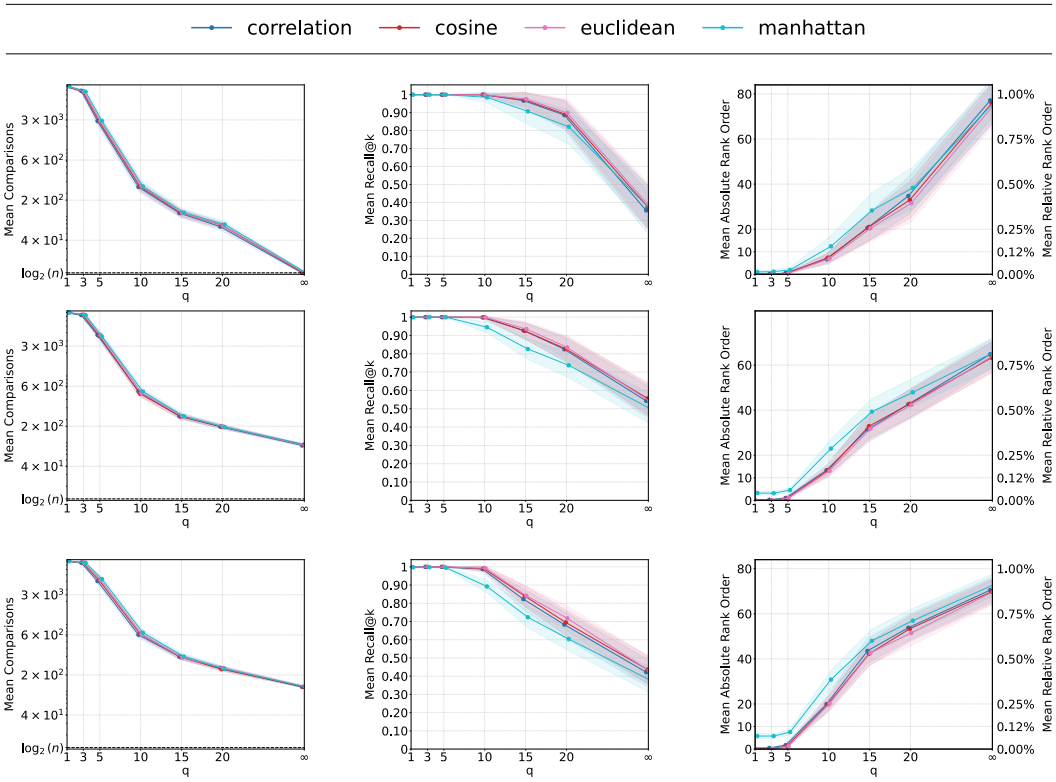
the rank order exhibits a sharp increase at $q = \infty$. A similar trend can be observed in the approximate setting, shown in Figures 17a and 17b. In Figure 19, the distribution of projected q-metrics shows a clustering effect as q increases. Distances become more concentrated around mean, while the frequency of close values also increases. This suggests that distances between points become closer, making true nearest neighbors more difficult to discern.

To address this, one can extend the nearest neighbor set using the Canonical Projection and then prune it to avoid loss in accuracy. This motivates a two-stage modification of the Infinity Search algorithm:

- **Broad Search**: The Infinity Search algorithm is used to retrieve an initial candidate set of K nearest neighbors. This will retrieve close neighbors in the q-metric space.
- **Specific Search**: Once the list of K candidate neighbors is available, the original distance D is used to retrieve the k real nearest neighbors.

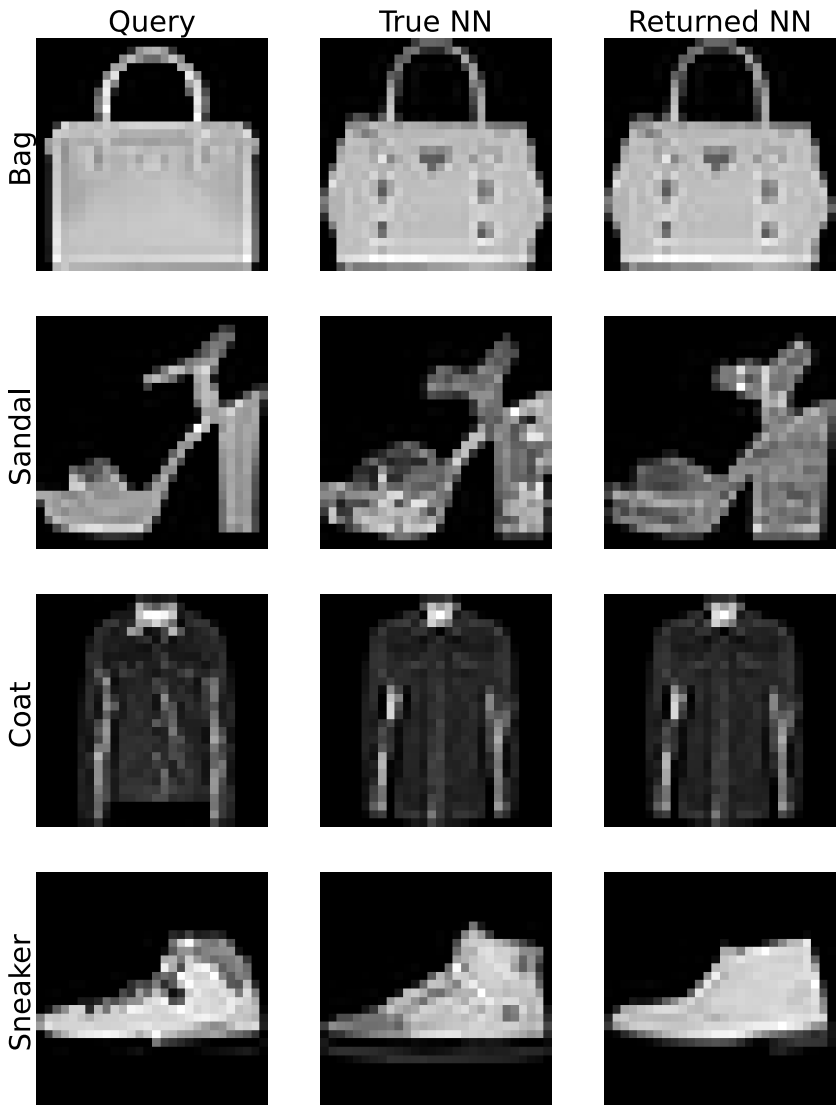
This two-stage retrieval strategy is used in some ANN methods, including HNSW (Malkov and Yashunin, 2020). As shown in the theoretical and approximate experiments of Sections F.2 and F.4, the Canonical Projection preserves locality but not the exact order of nearest neighbors. This makes the two-stage approach suitable for improving accuracy.

Figures 20a and 20b confirm the improvement, showing higher recall and more accurate rank alignment compared to earlier approximations. The gains are particularly notable in rank order, with a 3 to 4 times reduction in error. As expected, this comes with a decrease in speedup, since



(b) $n = 10{,}000$ points of Glove

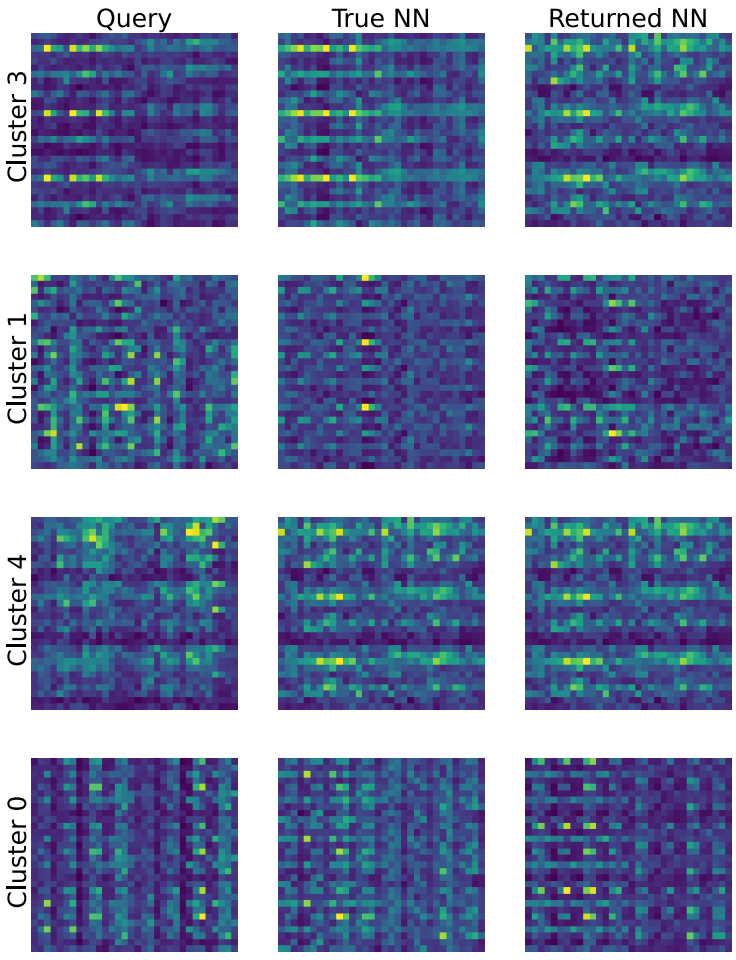
Figure 17: Number of comparisons and rank order when searching after approximating the Canonical Projection $\hat{E}_q$, with the learned map $\Phi(x;\theta)$. Solid lines denote the mean and shading the standard deviation computed across queries. The k-nearest neighbors are listed from top to bottom for $k = 1, 5, 10$



(a) Fashion-MNIST

Category	Query	True NN	Returned NN
Animals	dog	dogs	dogs
	lion	wolf	wolf
Colors	blue	pink	purple
	red	pink	purple
Clothing	pants	jeans	jeans
	shirt	shirts	worn
Tools	drill	drilling	drilling
	hammer	throw	flame

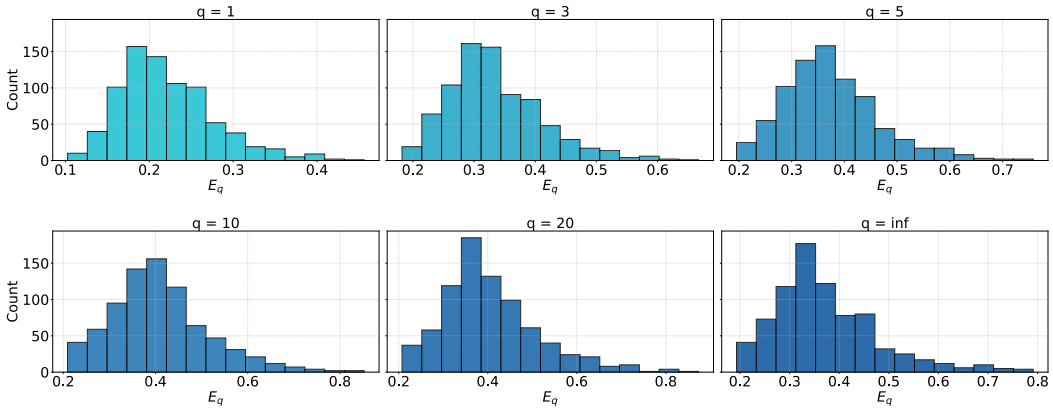
(b) GloVe



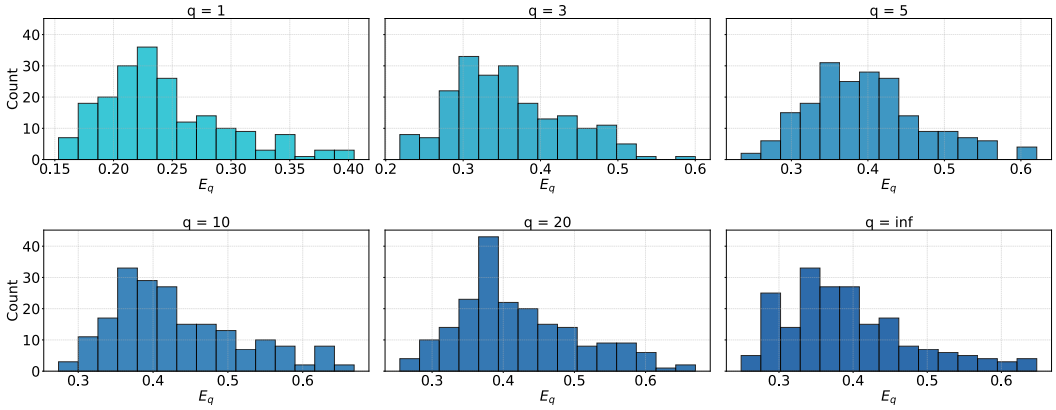
(c) GIST

Figure 18: Retrieval of dataset items with Infinity Search ($q = 5$).

the method processes a larger candidate set and computes original distances during Specific Search. Although the logarithmic comparison bound ($\log_2(n)$) no longer holds, the resulting speed remains competitive. The original Infinity Search can be recovered by setting $K = k$, while choosing $K > k$ offers additional flexibility to trade off speed and accuracy depending on the requirements of the searching problem.

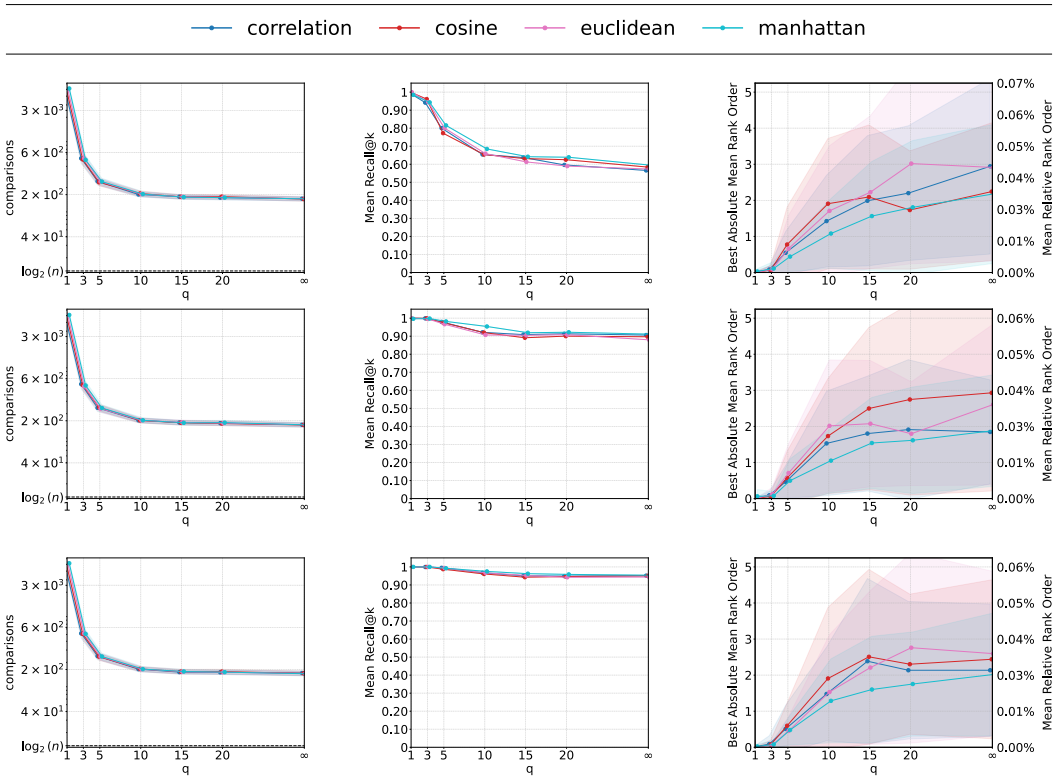


(a) Train



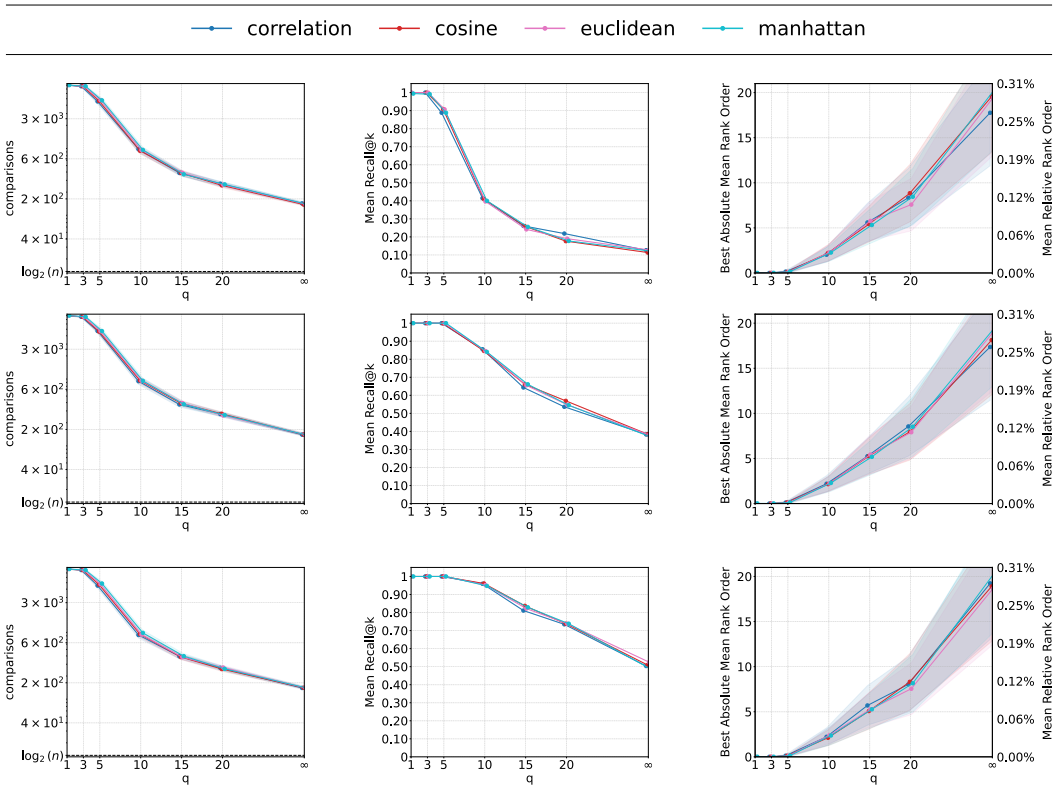
(b) Test

Figure 19: Histogram of the distance to Nearest Neighbor. In the X-axis E_q depicts distance values obtained after projecting when a query point is added. The Y-axis shows number of counts for that distance bin.



(a) $n = 10{,}000$ points of Fashion-MNIST

Figure 20: Number of comparisons, Recall@k and Rank Order when searching with a two-stage retrieval Infinity Search. Solid lines denote the mean and shading the standard deviation computed across queries. The k-nearest neighbors are listed from top to bottom for $k = 1, 5, 10$



(b) $n = 10{,}000$ points of Glove200

Figure 20: Number of comparisons, Recall@k and Rank Order when searching with a two-stage retrieval Infinity Search. Solid lines denote the mean and shading the standard deviation computed across queries. The k-nearest neighbors are listed from top to bottom for $k = 1, 5, 10$

F.6 VP-FORESTS

The VP-Tree structure can be naturally extended to a forest of smaller VP-Trees, allowing the index to scale without substantially degrading search performance. We split each dataset into T independent subsets and train a corresponding embedding $\Phi_1, \ldots, \Phi_T$ for each of them before constructing their respective VP-Trees. At query time, the query is embedded using every Φ_t, and the search is performed across all trees in the forest, collecting the candidates returned by each tree. Because the trees are built using independent embeddings, their errors are distributed across the forest, and taking the union of their candidate sets can recover neighbors that may be missed by any individual tree. Moreover, each tree can be constructed independently, providing opportunities for parallelization.

Empirically, we find that using a small number of trees becomes beneficial as the number of points grows. In particular, we observe that the approximation error remains smaller than the median pruning threshold between vantage points and their children (Figure 4). This separation between the learned distances and the pruning thresholds enables the VP-Trees to maintain effective pruning as the dataset scales.

F.7 EXTENDED EXPERIMENTAL REULTS

F.7.1 INFINITY SEARCH COMPETES IN ANN-BENCHMARKS

Infinity Search offers a configurable trade-off between query throughput and recall. To assess its competitiveness against modern ANN methods, we evaluated it within the ANN-Benchmarks framework (Aumüller et al., 2020) with batch mode and allowing parallelization. The analysis was lmited to 10k points due to computational limitations . We ran experiments on five datasets provided by the library and compared against a wide set of algorithms chosen for their balance of speed, accuracy, and open-source availability.

In both theoretical analysis and empirical benchmarks, Infinity Search consistently accelerates nearest-neighbor queries across all tested dissimilarities. On moderate-dimensional datasets such as Fashion-MNIST and GIST (Figure 21), it delivers a clear speedup by, in some cases, sacrificing perfect accuracy. Remarkably, on the high-dimensional Kosarak dataset—with Jaccard dissimilarity—Infinity Search outperforms competing methods by an even wider margin. This supports the flexibility of the method when less popular dissimilarities are required.

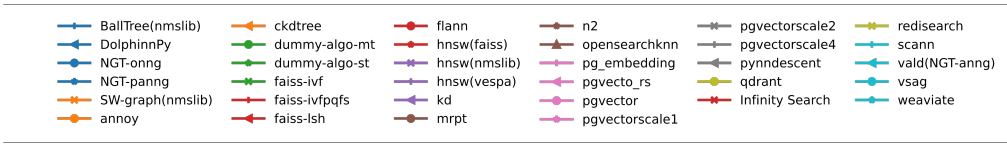
Across all datasets, it offers a favorable speed–accuracy trade-off for the $k = 1$ nearest-neighbor task. For larger neighborhood sizes ($k \in \{5, 10\}$), the increased comparison overhead prevents it from always leading in Recall@k; nevertheless, its performance remains competitive. Note that, since $n = 10{,}000$ in these experiments, modest rank-order errors at extreme speeds still correspond to few misplaced neighbors.

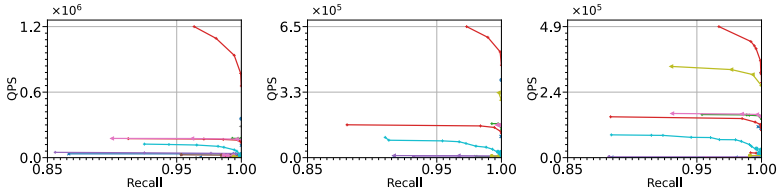
Overall, Infinity Search is a viable alternative when fast retrieval is required or non-Euclidean or less structured similarity are used.

F.7.2 INFINITY SEARCH COMPETES IN LARGER SCALE BENCHMARKS

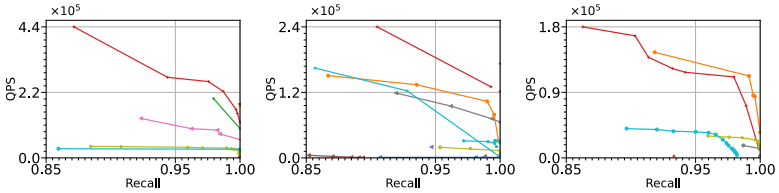
We provide an extension of the results in Section 5. In particular, we show the Pareto curves for retrieving $k \in 5, 10$ nearest neighbors. As expected, the performance of Infinity Search embeddings degrades slightly when retrieving a larger number of neighbors; however, the overall performance remains comparable to that observed for $k = 1$.

Additionally, we provide the training loss curves. Figure 23 depicts a sustained decrease in both training and test contrastive and distance losses across datasets.

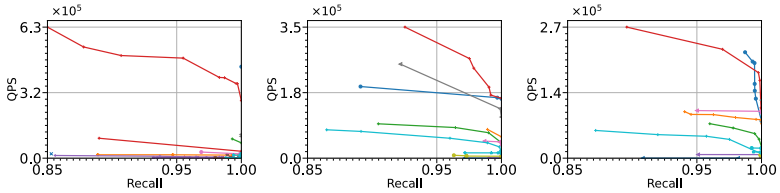




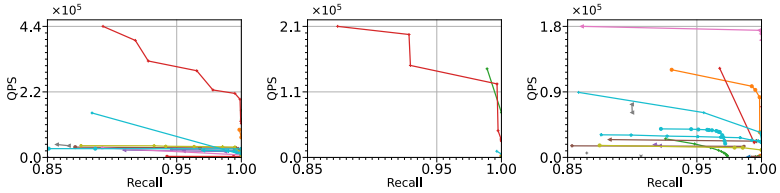
(a) Fashion-MNIST-784 (Euclidean)



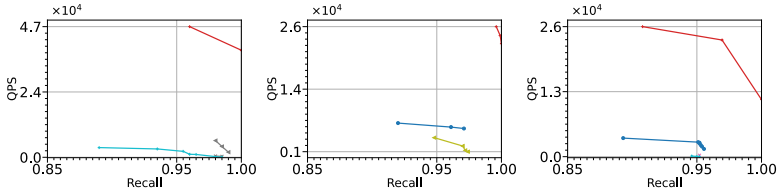
(b) GloVe-200 (Cosine)



(c) GIST-960 (Euclidean)



(d) NYTimes-256 (Cosine)



(e) Kosarak-41,000 (Jaccard)

Figure 21: ANN-Benchmarks comparison for $n = 10K$. Recall@k vs queries-per-second (QPS) across datasets. Columns within each row are $k = 1, 5, 10$

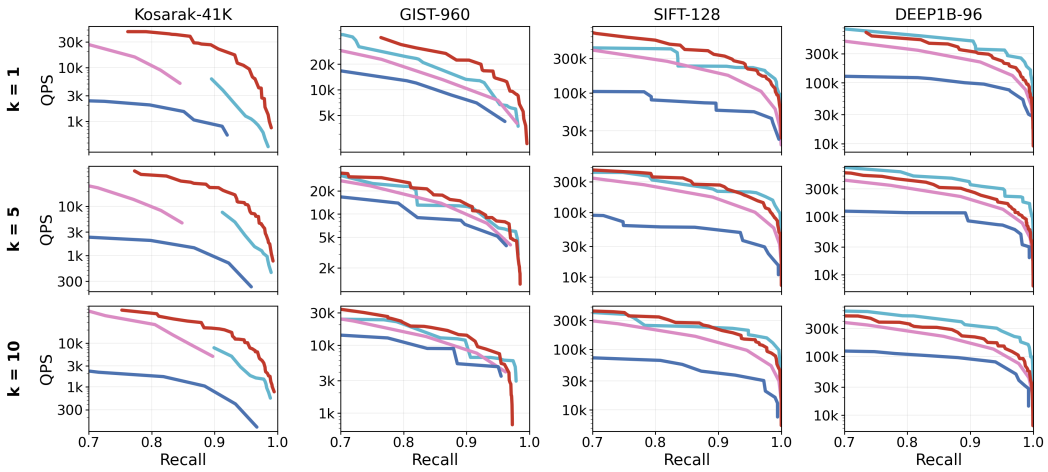


Figure 22: **Infinity Search supports retrieving multiple neighbors.** Pareto curves for retrieving $k \in \{5, 10\}$ nearest neighbors. Performance degrades only slightly compared with retrieving a single nearest neighbor.

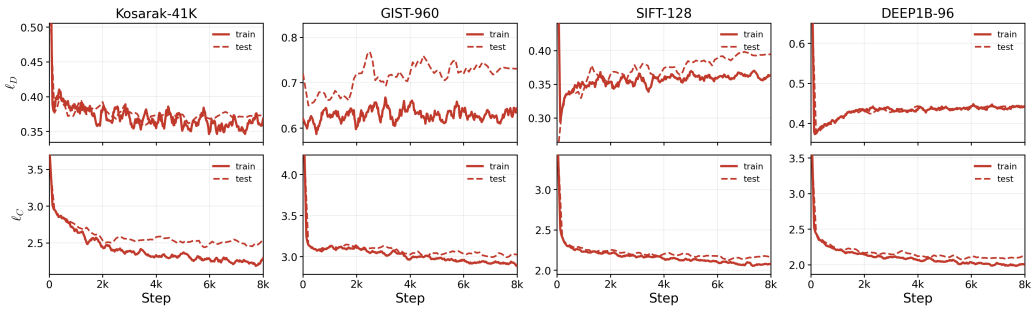


Figure 23: **Training loss curves.** We observe a sustained decrease in both ℓ_D and ℓ_C training loss curves.