Pruning Algorithms for Low-Dimensional Non-metric k-NN Search: A Case Study

Leonid Boytsov(✉) and Eric Nyberg

Carnegie Mellon University, Pittsburgh, PA, USA
{srchvrs,enh}@cs.cmu.edu

Abstract. We focus on low-dimensional non-metric search, where tree-based approaches permit efficient and accurate retrieval while having short indexing time. These methods rely on space partitioning and require a pruning rule to avoid visiting unpromising parts. We consider two known data-driven approaches to extend these rules to non-metric spaces: TriGen and a piece-wise linear approximation of the pruning rule. We propose and evaluate two adaptations of TriGen to non-symmetric similarities (TriGen does not support non-symmetric distances). We also evaluate a hybrid of TriGen and the piece-wise linear approximation pruning. We find that this hybrid approach is often more effective than either of the pruning rules. We make our software publicly available.

Keywords: k-NN search · Non-metric distance · VP-tree · TriGen

1 Introduction and Problem Definition

We consider a k nearest neighbor (k-NN) search, which is a popular technology used in many domains including, machine learning (ML), data mining, information retrieval, and natural language processing. Informally, k-NN search is a task of retrieving k data set entries closest to a query point with respect to some distance or similarity function. This problem originated from the real-world spatial search. In particular, Knuth famously formulated k-NN search as the (nearest) post-office problem [14]. With subsequent developments of the vector-space abstraction, the problem was generalized to searching in a multi-dimensional vector and/or generic metric space, where the latter may lack the structure of the vector space [10,21]. Motivated by emergence of useful non-metric distances—such as Bregman divergences [7]—the problem was recently generalized to more challenging domains [5,8,23,27].

Formally, we assume to have a possibly infinite domain containing objects x, y, z, ..., which are commonly called data points or simply points. The domain—sometimes called a *space*—is equipped with a *distance function* $d(x, y)$, which is used to measure dissimilarity of objects x and y. The value of $d(x, y)$ is interpreted as a degree of dissimilarity. The larger is $d(x, y)$, the more dissimilar points x and y are. Some distances

Authors gratefully acknowledge the support by the NSF grant #1618159: "Matching and Ranking via Proximity Graphs: Applications to Question Answering and Beyond".

Table 1. Distance functions

Denotation/Name	d(x, y)
Euclidean distance (L_2)	$\|\|x-y\|\|_2 = \left[\sum_i (x_i - y_i)^2\right]^{1/2}$
L_p ($p > 0$)	$\left[\sum_{i=1}^m (x_i - y_i)^p\right]^{1/p}$
Squared euclidean (L_2^2)	$\|\|x-y\|\|_2^2 = \sum_i (x_i - y^i)^2$
Cosine distance	$1 - \frac{\sum_i x_i y_i}{\|\|x\|\|_2 \|\|y\|\|_2}$
Kullback-Leibler diverg. (KL-div.) [15]	$\sum_{i=1}^m x_i \log \frac{x_i}{y_i}$
Itakura-Saito distance [13]	$\sum_{i=1}^m \left[\frac{x_i}{y_i} - \log \frac{x_i}{y_i} - 1\right]$
Rényi diverg. [20]	$\frac{1}{\alpha-1} \log \left[\sum_{i=1}^m x_i^\alpha y_i^{1-\alpha}\right]$, $\alpha > 0$ and $\alpha \neq 0$

are non-negative and become zero only when x and y have the highest possible degree of similarity. The *metric* distances are additionally symmetric and satisfy the triangle inequality. However, in general, we do not impose any restrictions on the value of the distance function (except that smaller values represent more similar objects).

We further assume that there is a data set D containing a *finite* number of domain points and a set of queries that belong to the domain but not to D. We then consider a standard top-k retrieval problem. Given a query q, a retrieval task consists in finding k data set points $\{x_i\}$ with smallest values of distances to the query among all data set points (ties are broken arbitrarily). Data points $\{x_i\}$ are called *nearest neighbors*. A search should return $\{x_i\}$ in the order of increasing distance to the query. If the distance is not symmetric, two types of queries can be considered: *left* and *right* queries. In a *left* query, a data point compared to the query is always the first (i.e., the left) argument of $d(x, y)$. For simplicity of exposition we consider only the case of left queries.

We employ a space-partitioning method VP-tree [19,24,26], but many other space-partitioning approaches can be used. Importantly, applying space-partitioning methods to non-metric data of even moderate dimensionality entails two problems. First, exact space-partitioning methods can degenerate to a brute-force search for just a dozen of dimensions [1,25]. Second, many generic space-partitioning methods incorporate pruning rules that crucially rely on the triangle inequality, which does not generally hold in non-metric spaces. Most existing non-metric space-partitioning methods employ specialized extensions specific to a concrete class of distances, e.g., to Bregman divergences [8,27] or Ptolemaic distances [12]. However, in a more general case we clearly need to resort to empirically derived analogs of the triangle inequality, which are inferred from data with a certain degree of approximation.

For these reasons, we focus only on *approximate* search methods. We also restrict our attention to low- and moderate-dimensional methods, because even approximate pruning methods are not effective in truly high dimensions. There has been a tremendous effort put into design of metric space-partitioning algorithms [10,21], but many

Table 2. Data sets

Name	Max. # of rec.	Dimensionality	Source
RandHist-d	0.5×10^6	$d = 8$	Histograms sampled uniformly from a simplex
RCV-d	0.5×10^6	$d \in \{8, 32, 128\}$	d-topic LDA [2] RCV1 [16] histograms
Wiki-d	2×10^6	$d \in \{8, 32, 128\}$	d-topic LDA [2] Wikipedia histograms

fewer methods are designed for non-metric domains. We aim to fill this gap by making the following contribution, which we detail in the rest of the paper:

- We carry out the first experimental comparison of two existing generic pruning algorithms, which include the piecewise linear approximation of the pruning rule [5] and TriGen [22].
- Unlike most prior work, many of our distances are non-symmetric. To deal with non-symmetry, we propose two adaptation of TriGen to non-symmetric distances and demonstrate that the choice of the symmetrization algorithm can be quite important.
- In our comprehensive evaluation, which includes 40 combinations of data sets and distances, we demonstrate the feasibility of accurate non-metric k-NN search for data of moderate dimensionality.
- We demonstrate that often best results can be achieved by combining these pruning methods.
- We find that on data of moderate dimensionality, the pruning algorithm needs to be quite efficient.

2 Methods and Materials

2.1 Data Sets and Distances

In our experiments, we use the following non-metric distances: L_2^2 (squared Euclidean) L_p distance, cosine distance, KL-divergence, the Itakura-Saito distance, and the family of Rényi divergence distances. The first three distances are symmetric. The remaining distances are statistical distances defined over probability distributions. For expository purposes, we also use the Euclidean metric distance L_2. Distances are listed in Table 1.

Statistical distances in general and, KL divergence in particular, play an important role in ML [8, 17]. They are typically non-symmetric. Both the KL-divergence and the Itakura-Saito distances were used in prior work [8]. The Rényi divergence is a single-parameter family of distances, which are not symmetric when the parameter $\alpha \neq 0.5$. By changing the parameter we can vary the degree of symmetry. In particular, large values of α and close-to-zero values result in highly non-symmetric distances. This flexibility allows us to "stress-test" retrieval methods on challenging non-symmetric distances.

The data sets are listed in Table 2. Wiki-d and RCV-d data sets consist of dense vectors of topic histograms with d topics. RCV-d set are created by Cayton [8] by applying the latent Dirichlet allocation (LDA) method [2] to the RCV1 collection [16]. These data sets have only 500K entries. Thus, we created larger sets from Wikipedia following a similar methodology. RandHist-d is a synthetic set of topics sampled uniformly from a d-dimensional simplex.

2.2 Pruning Algorithms for Space-Partitioning Methods

We employ a simple approach called a *vantage-point* tree (VP-tree) [19,24,26]. There are two reasons for this choice: for low- and moderate-dimensional data, it is often a hard-to-beat method. For example, in a preliminary experiment with L_2 on Wiki-8 data set for exact 10-NN search using NMSLIB [4], SA-tree [18], GH-tree [24], MVP-tree (binary version) [6], and VP-tree are respectively $70\times$, $210\times$, $1200\times$, $1600\times$ faster than the brute-force search. This comparison was done using the leaf bucket of size 50 for all methods (except SA-tree, which does not easily support bucketing) and without using any specific optimizations for any of the methods. We can see that VP-tree can outperform fancier alternatives including MVP-tree, which carries out $3\times$ fewer distance computations in this experiment.

VP-tree is a hierarchical space-partitioning method, which divides the space using hyperspheres. The output of an indexing algorithm is a hierarchical partitioning of the data set represented by a binary tree. This algorithm is a *recursive* procedure that operates on a subset of data—which we call an *active subset*—and on a partially built tree. At each step of recursion, the indexing algorithm checks if the number of active data points is below a certain threshold called the *bucket* size. If this is the case, the active data points are simply stored as a single bucket. Otherwise, the algorithm divides the active subset into two nearly equal parts, each of which is further processed recursively.

Division of the active subset starts with selecting a pivot π (e.g., randomly) and computing the distance from π to every other data point in the active subset. Assume that R is the median distance. Then, the active subset is divided into two subsets by the hypersphere with radius R and center π. Two subtrees are created. Points inside the pivot-centered hypersphere are placed into the left subtree. Points outside the pivot-centered hypersphere are placed into the right subtree. Points on the separating hypersphere may be placed arbitrarily. Because R is the median distance, each of the subtrees contains approximately half of active points.

In VP-tree k-NN search can be seen as a range search with a shrinking radius. The search algorithm is a *best-first* traversal procedure that starts from the root of the tree and proceeds recursively. It updates the search radius r as it encounters new close data points. Let us consider one step of recursion. If the search algorithm reaches a leaf of the tree, i.e., a bucket, all bucket elements are compared against the query. In other words, elements in the buckets are searched via brute-force search.

If the algorithm reaches an internal node X, there are exactly two subtrees representing two spaces partitions. The query belongs to exactly one partition. This is the "best" partition and the search algorithm always explores this partition recursively before deciding whether to explore the other partition. While exploring the best partition, we may encounter new close data points (pivots or bucket points) and further shrink the search radius. On completing the sub-recursion and returning to node X, we make a decision about pruning or exploring the other partition.

Piecewise-Linear Approximation of the Decision Rule. An essential part of this process is a decision function, which identifies situations when pruning is possible without sacrificing accuracy. Let us review the decision process. Recall that each internal node keeps pivot π and radius R, which define the division of the space into two subspaces. Although there are many ways to place a query ball, all locations can be divided into three categories, which are illustrated by Fig. 1. The red query ball "sits" inside the inner partition. Note that it does not intersect with the outer partition. For this reason, the outer partition cannot have sufficiently close data points, i.e., points with radius r from the query. Hence, this partition can be safely pruned. The blue query ball is located in the outer partition. Likewise, it does not intersect the other, inner, partition. Thus, this inner partition can be safely pruned. Finally, the gray query ball intersects both partitions. In this situation, sufficiently close points may be located in both partitions and no safe pruning is possible.

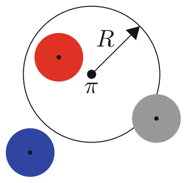


Fig. 1. Three types of query balls in VP-tree.

The pruning algorithm can be seen as the *binary classification problem*, which tells us whether we should visit both partitions or only the partition that contains the query. As we show previously [5], the problem can be solved by collecting training data and building a non-parametric model, but a simple two-parameter approach—described below—delivers better results. Let us first consider the case of a metric distance. From the triangle inequality it follows that the VP-tree search algorithm visits:

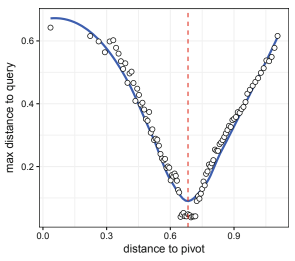
- *only* the left subtree if $d(\pi, q) < R - r$;
- *only* the right subtree if $d(\pi, q) > R + r$;
- both subtrees if $R - r \le d(\pi, q) \le R + r$.

Let us rewrite these rules using notation $D_{\pi,R}(x) = |R - x|$. It is easy to see that the search algorithm has to visit both partitions if and only if $r \ge D_{\pi,R}(d(\pi, q))$. If $r < D_{\pi,R}(d(\pi, q))$, we need to visit only one partition that contains the query point whereas the other partition can be safely pruned.

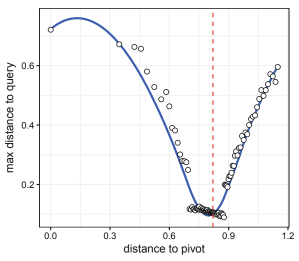
In other words, the pruning decision is made by comparing the query radius r with the value of the function $D_{\pi,R}(x)$, whose only argument is the distance from the query to the pivot $d(\pi, q)$.[1] This basic rule can also be learned from data for non-metric distances. Our initial approach to learn $D_{\pi,R}(x)$ employed a stratified sampling procedure (see § 2 of the supplemental materials of our publication [5]). However, it was expensive and not very accurate. For this reason, we also implemented a simple *parametric* approximation whose parameters are selected to optimize efficiency at a given value of recall.

To choose the right parametric representation, we examine the (*approximate*) functions $D_{\pi,R}(x)$ learned by the sampling algorithm. Plots of functions $D_{\pi,R}(x)$ learned from data are shown in Fig. 2. Small dots in these plots represent function values obtained by sampling. Blue curves are fit to these dots. In these plots, we use only topic histogram data RCV-d, where $d \in \{8, 32\}$ and random 8-dimensional histograms (RandHist-8).

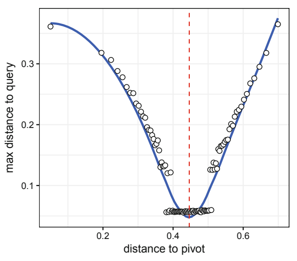
[1] Recall that k-NN search is executed as a best-first range search with a shrinking radius.



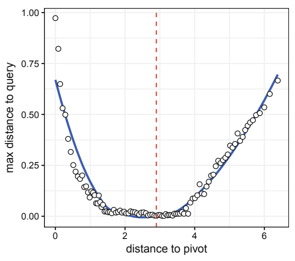
(a) RCV-8 (L_2)



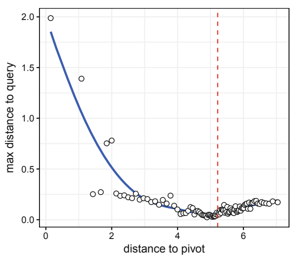
(b) RCV-32 (L_2)



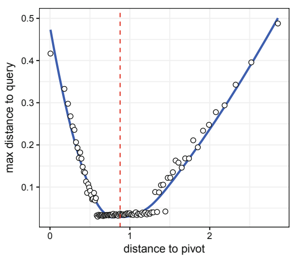
(c) RandHist-8 (L_2)



(d) RCV-8 (KL-div.)



(e) RCV-32 (KL-div.)



(f) RandHist-8 (KL-div.)

Fig. 2. The empirically obtained (*approximate*) pruning decision function $D_{\pi,R}(x)$

For the Euclidean data (Panels 2a–2c in Fig. 2), $D_{\pi,R}(x)$ resembles a piecewise linear function close to the exact metric pruning function $|R - x|$. For the KL-divergence data (Panels 2d–2f in Fig. 2), $D_{\pi,R}(x)$ looks like either a U-shape or a hockey-stick curve. These observations originally motivated the use of a piecewise *polynomial* decision function, which is formally defined as:

$$D_{\pi,R}(x) = \begin{cases} \alpha_{left}|x - R|^{\beta_{left}}, & \text{if } x \le R \\ \alpha_{right}|x - R|^{\beta_{right}}, & \text{if } x \ge R \end{cases}, \tag{1}$$

where β_i are positive integers. However, preliminary experiments convinced us to switch to a simple piece-wise linear variant. First, we learned that using different β_i did not make our pruning function sufficiently more accurate. However, it made the optimization problem harder due to additional parameters (so we set $\beta = \beta_1 = \beta_2$). Second, we found that in many cases a polynomial approximation was not better than a piecewise linear one, especially when dimensionality was high.

This is not very surprising: Due to the concentration of measure, for most data points the distance to the pivot π is close to the median distance R (which corresponds to the boundary between two VP-tree partitions). If we explore the shape of $D_{\pi,R}(x)$ in Panels 2a and 2e around the median, we can see that a piecewise linear shape approximation is quite reasonable. To sum up, we ended up using the piecewise linear parametric decision rule defined as:

$$D_{\pi,R}(x) = \begin{cases} \alpha_{left}|x - R|, & \text{if } x \le R \\ \alpha_{right}|x - R|, & \text{if } x \ge R \end{cases} \tag{2}$$

This is similar to stretching of the triangle inequality proposed by Chávez and Navarro [9]. There are two crucial differences, however. First, we utilize different values of α_i, i.e., $\alpha_{left} \neq \alpha_{right}$, while Chávez and Navarro used $\alpha_{left} = \alpha_{right}$. Second, we devise a simple training procedure to obtain values of α_i that maximize efficiency at a given recall value. For details, we address the reader to relevant publications [3,5].

TriGen. TriGen consists in "stretching" the distance function using a monotonic concave transformation [22] that reduces non-metricity of the distance. TriGen is designed only for *bounded*, *semimetric* distances, which are crucially *symmetric*, non-negative, and become zero only for identical data points. We are not aware of any prior extensions to non-symmetric distances except a straightforward filter-and-refine approach.

Let x, y, z be an arbitrary ordered triple of points such that $d(x, y)$ is the largest among three pairwise distances, i.e., $d(x, y) \geq \max(d(x, z), d(z, y))$. If $d(x, y)$ is a metric distance, the following conditions should all be true:

$$\begin{aligned} d(x,y) &\leq d(x,z) + d(z,y) \\ d(y,z) &\leq d(y,x) + d(x,z) \\ d(x,z) &\leq d(x,y) + d(y,z) \end{aligned} \tag{3}$$

Because $d(x, y) \geq \max(d(x, z), d(z, y))$, the second and the third inequalities in (3) are trivially satisfied for (not necessarily metric) *symmetric* and *non-negative* distances. However, the first condition can be violated if the distance is non-metric. The closer is the distance to the metric distance, the less frequently we encounter such violations. Thus, it is quite reasonable to assess the degree of deviation from metricity by estimating a probability that the triangle inequality is violated (for a randomly selected triple), which is exactly what is suggested by Skopal [22].

Skopal proposes a clever way to decrease non-metricity by constructing a new distance $f(d(x, y))$, where $f()$ is a monotonically increasing concave function. The concave function "stretches" the distance and makes it more similar to a true metric compared to the original distance $d(x, y)$. At the same time, due to the monotonicity of such a transformation, the k-NN search using the modified distance produces the same result as the k-NN search using the original distance. Thus, the TriGen strategy to dealing with non-metric data consists in (1) employing a monotonic transformation that makes a distance approximately metric while preserving the original set of nearest neighbors, and (2) indexing data using an exact metric-space access method.

A TriGen mapping $f(x)$—defined for $0 \leq x \leq 1$—is selected from the union of two parametric families of concave functions, which are termed as bases:

- A fractional power base $FP(x, w) = x^{\frac{1}{1+w}}$;
- A Rational Bézier Quadratic (RBQ) base $RBQ_{(a,b)}(x, w)$, $0 \leq a < b \leq 1$. The exact functional form of RBQ is not relevant to this discussion (see [22] for details).

Note that parameters w, a, and b are treated as constants, which define a specific functional form. By varying these parameters we can design a necessary stretching function. The larger is the value of w the more concave is the transformation and the more "metric" is the transformed distance. In particular, as $w \to \infty$, both RBQ and FP converge to one minus the Dirac delta function. This limit function of all bases is equal to zero for

$x = 0$ and to one for $0 < x \le 1$. As noted by Skopal [22], applying such a degenerate transformation produces a *trivial* metric space where $d(x, x) = 0$ and $d(x, y) = C$ for some constant $C > 0$ and all $x \ne y$.

A learning objective of TriGen, however, is to select a *single* concave function that satisfies the accuracy requirements while allowing for efficient retrieval. The fraction of violations is computed for a set of `trigenSampleTripletQty` ordered data point triples sampled from a set of `trigenSampleQty` data points, which are, in turn, selected randomly from the data set (uniformly and without replacement). The fraction of violations is required to be above the threshold `trigenAcc`. Values `trigenSampleTripletQty`, `trigenSampleQty`, and `trigenAcc` are all parameters in our implementation of TriGen. To assess efficiency Skopal uses the value of an intrinsic dimensionality as a proxy metric (see [22] for details). The idea is that the modification of the distance with the lowest intrinsic dimensionality should result in the fast retrieval method.

Because it is not feasible to optimize over the infinite set of transformation functions, TriGen employs a finite pool of bases, which includes FB and multiple RBQ bases for all possible combinations of parameters a and b such that $0 \le a < b \le 1$. For each base, TriGen uses a binary search to find the minimum parameter w such that the transformed distance deviates from a metric distance within specified limits. Then the base with minimum intrinsic dimensionality is selected.

TriGen has two major limitations: In addition to be non-negative, the distance should be symmetric and bounded. Bounding can be provided by using $\min(d(x, y)/D_{\max}, 1)$ instead of the original distance.[2] Note that $D_{\max}$ is an empirically estimated maximum distance (obtained by computing $d(x, y)$ for a sample of data set point pairs).

As noted by Skopal [22], searching with a non-symmetric distance can be partially provided by a filter-and-refine approach where a fully *min*-symmetrized distance $\min(d(x, y), d(y, x))$ is used during the filtering step. However, as we learn from our prior work §§ 2.3.2.3–2.3.2.4 [3], the filtering step has to carry out a k_c-NN search with k_c (sometimes substantially) larger than k. This is required to compensate for the lack of accuracy due to replacing the original distance with the symmetrized one. In that, using $k_c > k$ leads to reduced efficiency. Thus, instead of the complete filter-and-refine symmetrization, we consider two simple alternatives. In both cases we first apply the TriGen algorithm to the min-symmetrized distance. As a result, we obtain a mapping that makes this min-symmetrized distance to be closer to a metric distance. However, this mapping is used differently in the two modifications of TriGen.

Recall that in a typical space-partitioning method, we divide the data into reasonably large buckets (50 in our experiments). The k-NN search is simulated as a range search with a shrinking radius. In the case the first modification of TriGen, while we traverse the tree, we compute the original and the min-symmetrized distance for two purposes:

– shrinking the dynamic radius of the query using the *symmetrized* distance;
– checking if the *original* distance is small enough to update the current set of k nearest neighbors.

[2] For efficiency reasons this is simulated via multiplication by inverse maximum distance.

Table 3. Efficiency-effectiveness results for metric VP-tree on non-metric data for 10-NN search (using complete data sets).

	RCV-8		Wiki-8		RandHist-8		Wiki-128	
	Recall	Impr. in eff.	Recall	Impr. in eff.	Recall	Impr. in eff.	Recall	Impr. in eff.
$L_p(p = 0.125)$	0.41	1065	0.66	15799	0.45	136	0.07	14845
$L_p(p = 0.25)$	0.61	517	0.78	14364	0.66	115	0.09	396
$L_p(p = 0.5)$	0.91	926	0.94	14296	0.92	174	0.50	33
L_2^2	0.69	1607	0.78	5605	0.56	1261	0.55	114
Cosine dist	0.67	1825	0.62	3503	0.58	758	0.73	55
Rényi div. ($\alpha = 0.25$)	0.66	5096	0.70	24246	0.50	3048	0.48	1277
Rényi div. ($\alpha = 0.75$)	0.61	9587	0.66	35940	0.50	4673	0.50	468
Rényi div. ($\alpha = 2$)	0.40	22777	0.66	46122	0.38	11762	0.71	55
KL-div.	0.52	1639	0.67	5271	0.46	610	0.56	41
Itakura-Saito	0.46	706	0.69	4434	0.41	1172	0.14	384

When we reach a bucket, for every data point in the bucket, we can compute both the original and the symmetrized distance. The symmetrized distance is used to update the query radius, while the original distance is used to update the set of k nearest neighbors. This is our first modification of TriGen which we refer to as *TriGen 0*.

In the second variant of TriGen, which we refer to as *TriGen 1*, we use *only* the original distance to compute the distance from the query to bucket data points. When we compute the distance to the pivots, we compute the min-symmetrized distance and apply a metrizing transformation. However, when we process bucket data points, we compute only the original distance. Consequently, we shrink the dynamic query radius using values of $f(d(x, y))$ instead of $\min(f(d(x, y)), f(d(y, x)))$, In TriGen 1, we expect the query radius to shrink somewhat slower compared to TriGen 0, which, in turn, can reduce the effectiveness of pruning. However, we hope that nearly halving the number of distance computations would have a larger effect on overall retrieval time.

3 Experiments

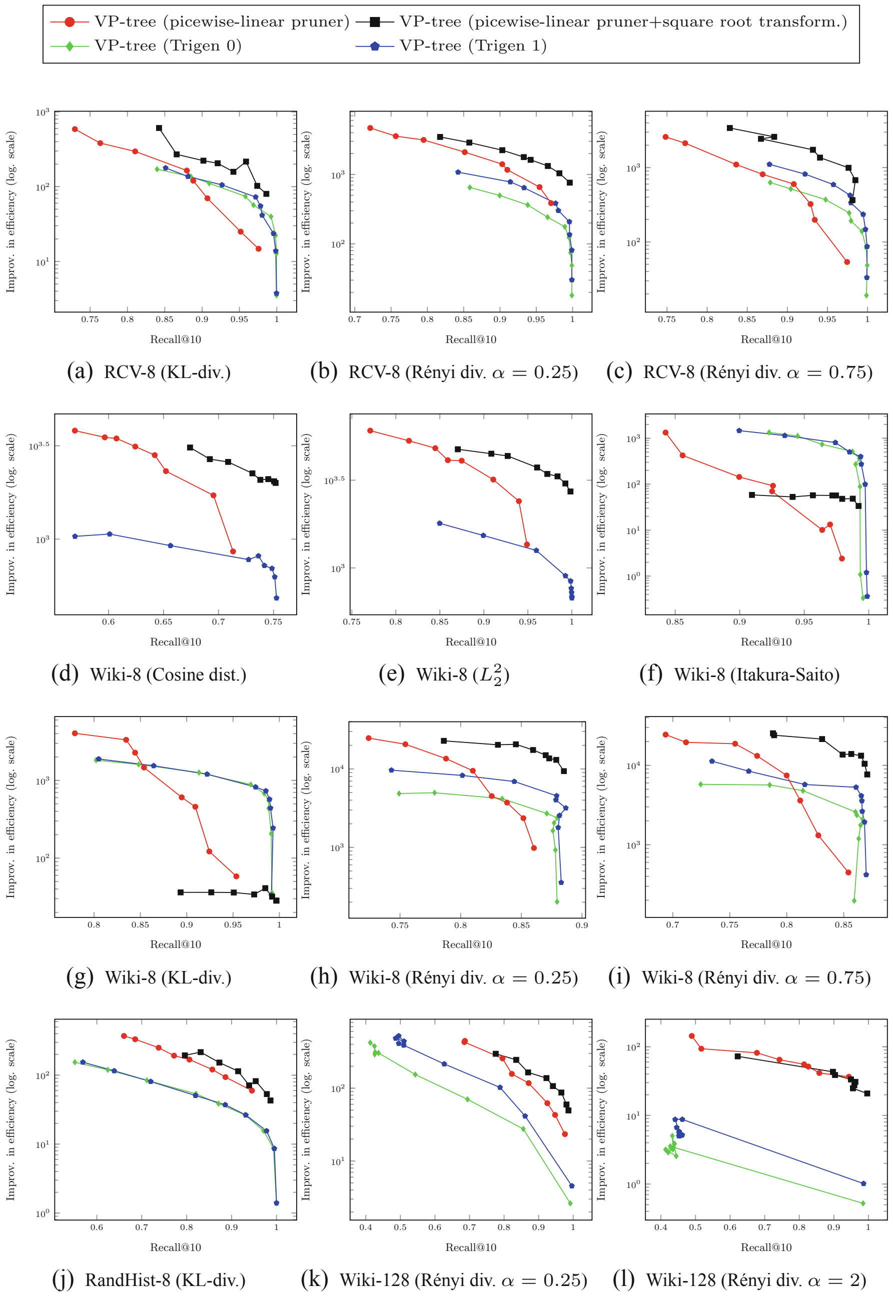
3.1 Experimental Setup and Preliminary Experiments

We compare TriGen and the piecewise-linear pruning approach using the NMSLIB [4] implementation of the VP-tree (method `vptree_trigen`)[3]. Experiments are run on a laptop (i7-4700MQ @ 2.40 GHz with 16 GB of memory). The accuracy of retrieval is measured via recall (equal to the average fraction of neighbors found).

We use two variants of TriGen (TriGen 0 and TriGen 1), but for symmetric distances, we use only TriGen 1. The TriGen algorithm that finds an optimal mapping function was downloaded from the author's website[4] and incorporated into NMSLIB.

[3] https://github.com/nmslib/nmslib/tree/nmslib4a_bigger_reruns.

[4] http://siret.ms.mff.cuni.cz/skopal/download.htm.



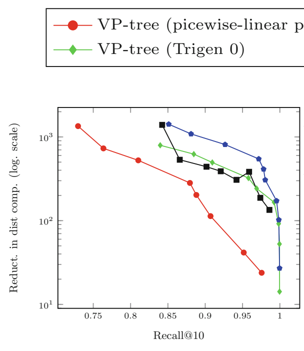
(a) RCV-8 (KL-div.) (b) RCV-8 (Rényi div. $\alpha = 0.25$) (c) RCV-8 (Rényi div. $\alpha = 0.75$)

(d) Wiki-8 (Cosine dist.) (e) Wiki-8 (L_2^2) (f) Wiki-8 (Itakura-Saito)

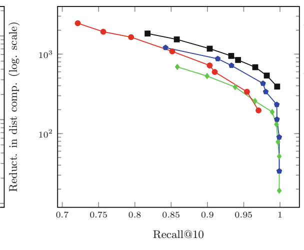
(g) Wiki-8 (KL-div.) (h) Wiki-8 (Rényi div. $\alpha = 0.25$) (i) Wiki-8 (Rényi div. $\alpha = 0.75$)

(j) RandHist-8 (KL-div.) (k) Wiki-128 (Rényi div. $\alpha = 0.25$) (l) Wiki-128 (Rényi div. $\alpha = 2$)

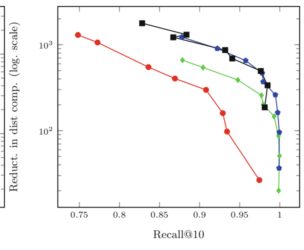
Fig. 3. Improvement in efficiency vs recall for VP-tree based methods in 10-NN search. Best viewed in color.



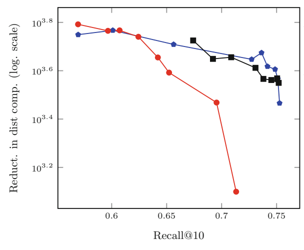
(a) RCV-8 (KL-div.)



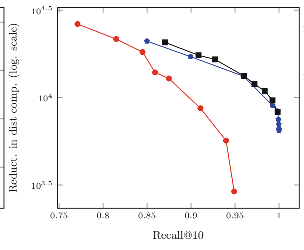
(b) RCV-8 (Rényi div. $\alpha = 0.25$)



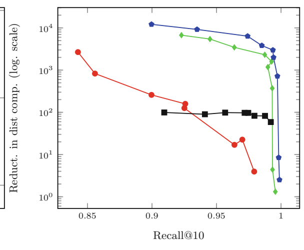
(c) RCV-8 (Rényi div. $\alpha = 0.75$)



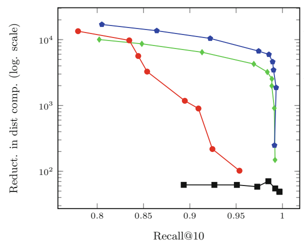
(d) Wiki-8 (Cosine dist.)



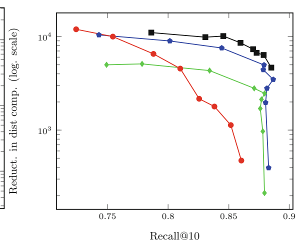
(e) Wiki-8 (L_2^2)



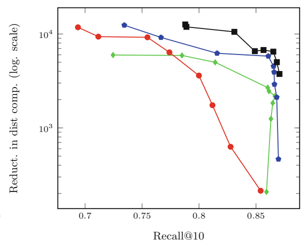
(f) Wiki-8 (Itakura-Saito)



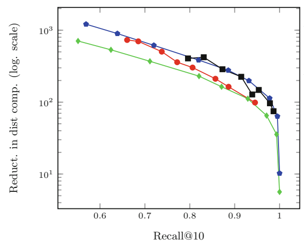
(g) Wiki-8 (KL-div.)



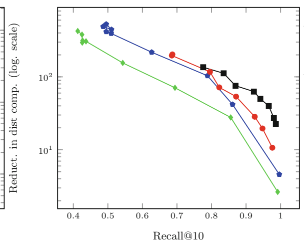
(h) Wiki-8 (Rényi div. $\alpha = 0.25$)



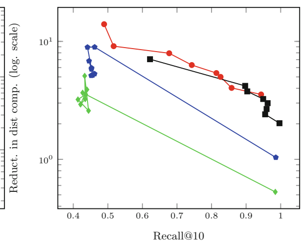
(i) Wiki-8 (Rényi div. $\alpha = 0.75$)



(j) RandHist-8 (KL-div.)



(k) Wiki-128 (Rényi div. $\alpha = 0.25$)



(l) Wiki-128 (Rényi div. $\alpha = 2$)

Fig. 4. Reduction in the number of distance computations vs recall for VP-tree based methods in 10-NN search. Best viewed in color.

The optimization procedure employs a combination of parameters a and b, where a are multiples of 0.01, b are multiples of 0.05, and $0 \leq a < b \leq 1$. The sampling parameters are set as follows: `trigenSampleTripletQty` = 10000, `trigenSampleQty` = 5000.

TriGen is compared against two variants of NMSLIB VP-tree, which rely on the piecewise-linear pruner. The second variant uses a clever TriGen idea of applying a concave mapping to make the distance more similar to a metric one. However, unlike TriGen [22], we do not carry an extensive search for an optimal transformation but rather apply, perhaps, the simplest and fastest *monotonic concave* transformation possible, which consists in taking a square root. On Intel the square root is computed the instruction `sqrtss`, which typically takes less than 10 cycles [11].

In our main experiments, we employ 40 combinations of data sets and distances. All distances are non-metric: We experiment with both symmetric and non-symmetric ones. Due to space limitations, we do not present all the results here and refer the reader to our unpublished technical report for the complete set of results (§ 2.3.3 [3]).

Before we proceed, we must answer the following question: "How difficult are these data sets and distances"? To ensure we do not deal with mildly non-metric data, we attempted to index this data using a metric variant VP-tree without adapting the pruning rule to non-metric distances. Results for randomly selected 1K queries are presented in Table 3 (for a subset of distances and data sets), where we show improvement in efficiency and respective recall.

It can be seen that nearly all the combinations of data and distance functions are substantially non-metric: Searching using a metric VP-tree is usually fast, but the accuracy is *unacceptably* low. In particular, this is true for Wiki-8 and Wiki-128 data sets with KL-divergence (which are also used in our main experiments). One exception, is the L_p distance for $p = 0.5$, where recall of about 90% is achieved for three low-dimensional data sets. However, as p decreases, the recall decreases sharply, i.e., the distance function becomes "less" metric. To summarize, we clearly deal with challenging non-metric data sets, where both accurate and efficient retrieval is not possible to achieve by a straightforward application of metric-space search methods.

3.2 Main Experiments

Experimental results for 16 out of 40 cases are presented in Figs. 3 and 4. The remaining results can be found in the technical report (§ 2.3.3 [3]). In Fig. 3, we measure efficiency directly in terms of wall-clock time improvement over the brute-force search. In Fig. 4, we show improvement in the number of distance computations (again compared to the brute-force search).

First and foremost, we can see that VP-tree with a data-adapted pruning rule can enable accurate non-metric k-NN search for data of moderate dimensionality. When comparing TriGen against the piecewise linear pruner in terms of pure efficiency, the results are a bit of the mixed bag. Yet, the piecewise linear pruner is typically better (in 23 cases out of 40 on the full set, see § 2.3.3 [3]).

However, the piecewise linear pruner combined with the square-root distance transform is nearly always better than the basic piecewise linear pruner. In Panels 3d, 3e, 3a, 3b, 3c the improvement is up to one order of magnitude. The combination of the piecewise linear pruner with the square root transform outperforms TriGen in all but two cases, sometimes by an order of magnitude. In Panels 3g and 3f, however, TriGen can also be an order of magnitude faster than the piecewise linear pruner.

It is important to note, however, that there is often little to no difference between the hybrid pruning approach and TriGen in terms of the reduction in the number of distance computations (see Fig. 4). The most likely explanation for this discrepancy is that the transformation functions used in the adopted TriGen implementation are quite expensive to compute.

Finally, we can see that TriGen 1 is never less efficient than TriGen 0. Furthermore, TriGen 1 is up two times more efficient in four cases (see Panels 3h, 3i, 3k, 3l). This is somewhat unsurprising, because TriGen 0 computes both $d(x, q)$ and $d(q, x)$ for every data point visited by the search. Although this may permit a more effective pruning, the cost of extra distance computations outweigh the benefits (at least on our data).

4 Conclusion

We carry out the first comparison of two generic pruning approaches for non-metric data. Our approach is comprehensive and involves 40 combinations of data sets and distances, which cannot be handled by a classic metric-space access method. We extend TriGen to the case of non-symmetric distances and demonstrate that VP-tree with a data-adapted pruning rule can enable accurate non-metric k-NN search for data of moderate dimensionality by using the modified TriGen, the piecewise linear approximation of the metric pruning rule, or by the hybrid approach. In that, we find that this hybrid approach is often more effective than either of the pruning rules. Our software is publicly available: NMSLIB branch `nmslib4a_bigger_reruns`, search method `vptree_trigen`.[5]

Acknowledgments. This work was done while Leonid Boytsov was a PhD student at CMU. Authors gratefully acknowledge the support by the NSF grant #1618159.

References

1. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C., Buneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-49257-7_15
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Boytsov, L.: Efficient and accurate non-metric k-NN search with applications to text matching. Ph.D. thesis, Carnegie Mellon University (2017)
4. Boytsov, L., Naidan, B.: Engineering efficient and effective non-metric space library. In: Brisaboa, N., Pedreira, O., Zezula, P. (eds.) SISAP 2013. LNCS, vol. 8199, pp. 280–293. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41062-8_28
5. Boytsov, L., Naidan, B.: Learning to prune in metric and non-metric spaces. In: Proceedings of NIPS 2013, pp. 1574–1582 (2013)

[5] https://github.com/nmslib/nmslib/tree/nmslib4a_bigger_reruns.

6. Bozkaya, T., Özsoyoglu, Z.M.: Indexing large metric spaces for similarity search queries. ACM Trans. Database Syst. **24**(3), 361–404 (1999). https://doi.org/10.1145/328939.328959
7. Bregman, L.: The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. USSR Comput. Math. Math. Phys. **7**(3), 200–217 (1967)
8. Cayton, L.: Fast nearest neighbor retrieval for bregman divergences. In: Proceedings of the 25th International Conference on Machine Learning, pp. 112–119. ACM (2008)
9. Chávez, E., Navarro, G.: Probabilistic proximity search: fighting the curse of dimensionality in metric spaces. Inf. Process. Lett. **85**(1), 39–46 (2003)
10. Chávez, E., Navarro, G., Baeza-Yates, R.A., Marroquín, J.L.: Searching in metric spaces. ACM Comput. Surv. **33**(3), 273–321 (2001)
11. Fog, A.: Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for intel, AMD and VIA CPUs (2011)
12. Hetland, M.L., Skopal, T., Lokoč, J., Beecks, C.: Ptolemaic access methods: challenging the reign of the metric space model. Inf. Syst. **38**(7), 989–1006 (2013)
13. Itakura, F., Saito, S.: Analysis synthesis telephony based on the maximum likelihood method. In: Proceedings of the 6th International Congress on Acoustics, pp. C17–C20 (1968)
14. Knuth, D.E.: The Art of Computer Programming: Volume 3: Sorting and Searching. Atmospheric Chemistry & Physics (1973)
15. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
16. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: a new benchmark collection for text categorization research. J. Mach. Learn. Res. **5**, 361–397 (2004)
17. Markatou, M., Chen, Y., Afendras, G., Lindsay, B.G.: Statistical distances and their role in robustness. In: Chen, D.-G., Jin, Z., Li, G., Li, Y., Liu, A., Zhao, Y. (eds.) New Advances in Statistics and Data Science. IBSS, pp. 3–26. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69416-0_1
18. Navarro, G.: Searching in metric spaces by spatial approximation. VLDB J. **11**(1), 28–46 (2002)
19. Omohundro, S.M.: Five balltree construction algorithms (1989). iCSI Technical Report TR-89-063. http://www.icsi.berkeley.edu/icsi/publication_details?ID=000562
20. Rényi, A.: On measures of entropy and information. In: Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 547–561 (1961)
21. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc., San Francisco (2005)
22. Skopal, T.: Unified framework for fast exact and approximate search in dissimilarity spaces. ACM Trans. Database Syst. **32**(4), 29 (2007)
23. Skopal, T., Bustos, B.: On nonmetric similarity search problems in complex domains. ACM Comput. Surv. **43**(4), 34 (2011)
24. Uhlmann, J.K.: Satisfying general proximity/similarity queries with metric trees. Inf. Process. Lett. **40**(4), 175–179 (1991)
25. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: VLDB, vol. 98, pp. 194–205 (1998)
26. Yianilos, P.N.: Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proceedings of ACM/SIGACT-SIAM 1993, pp. 311–321 (1993)
27. Zhang, Z., Ooi, B.C., Parthasarathy, S., Tung, A.K.H.: Similarity search on Bregman divergence: towards non-metric indexing. PVLDB **2**(1), 13–24 (2009)