Similarity Search and Applications. SISAP 2017. The final authenticated version is available online at:
https://doi.org/10.1007/978-3-319-68474-1_7

High-Dimensional Simplexes for Supermetric Search

Richard Connor[1], Lucia Vadicamo[2], and Fausto Rabitti[2]

[1] Department of Computer and Information Sciences,
University of Strathclyde, Glasgow, G1 1XH, United Kingdom
[2] ISTI - CNR, Via Moruzzi 1, 56124 Pisa, Italy
richard.connor@strath.ac.uk
{lucia.vadicamo,fausto.rabitti}@isti.cnr.it

Abstract. [3]
In a metric space, triangle inequality implies that, for any three objects, a triangle with edge lengths corresponding to their pairwise distances can be formed. The *n-point property* is a generalisation of this where, for any $(n+1)$ objects in the space, there exists an n-dimensional simplex whose edge lengths correspond to the distances among the objects. In general, metric spaces do not have this property; however in 1953, Blumenthal showed that any semi-metric space which is isometrically embeddable in a Hilbert space also has the n-point property.
We have previously called such spaces *supermetric* spaces, and have shown that many metric spaces are also supermetric, including Euclidean, Cosine, Jensen-Shannon and Triangular spaces of any dimension.
Here we show how such simplexes can be constructed from only their edge lengths, and we show how the geometry of the simplexes can be used to determine lower *and upper* bounds on unknown distances within the original space. By increasing the number of dimensions, these bounds converge to the true distance.
Finally we show that for any Hilbert-embeddable space, it is possible to construct Euclidean spaces of arbitrary dimensions, from which these lower and upper bounds of the original space can be determined. These spaces may be much cheaper to query than the original. For similarity search, the engineering tradeoffs are good: we show significant reductions in data size and metric cost with little loss of accuracy, leading to a significant overall improvement in exact search performance.

Keywords: Supermetric Space · Metric Search · Metric Embedding · Dimensionality Reduction

1 Introduction

To set the context, we are interested in searching a (large) finite set of objects S which is a subset of an infinite set U, where (U, d) is a metric space. The

[3] The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-68474-1

general requirement is to efficiently find members of S which are similar to an arbitrary member of U, where the distance function d gives the only way by which any two objects may be compared. There are many important practical examples captured by this mathematical framework, see for example [**?**,**?**]. Such spaces are typically searched with reference to a query object $q \in U$. A threshold search for some threshold t, based on a query $q \in U$, has the solution set $\{s \in S \text{ such that } d(q,s) \leq t\}$.

This becomes an interesting problem when exhaustive search is intractable, in which case the research problem is to find ways of pre-processing the collection ahead of query time in order to minimise the cost of query. There are three main problems with achieving efficiency. Most obviously, if the search space is very large, scalability is required. Less obviously, when the search space is large, semantic accuracy is important to avoid large numbers of false positive results – in the terminology of information retrieval, *precision* becomes relatively more important that *recall*. To achieve higher semantic accuracy will usually require more expensive metrics, and larger data representations.

Here, we present a new technique which can be used to address all three of these issues in supermetric spaces. Using properties of finite isometric embedding, we show a mechanism which allows spaces with certain properties to be translated into a second, smaller, space. For a metric space (U,d), we describe a family of functions ϕ_n which can be created by measuring the distances among n objects sampled from the original space, and which can then be used to create a *surrogate* space:

$$\phi_n : (U,d) \rightarrow (\mathbb{R}^n, \ell_2)$$

with the property

$$\ell_2(\phi_n(u_1), \phi_n(u_2)) \leq d(u_1,u_2) \leq g(\phi_n(u_1), \phi_n(u_2))$$

for an associated function g.

The advantages of the proposed technique are that (a) the ℓ_2 metric is very much cheaper than some Hilbert-embeddable metrics; (b) the size of elements of $\mathbb{R}^n$ may be much smaller than elements of U, and (c) in many cases we can achieve both of these along with an *increase* in the scalability of the resulting search space.

2 Related Work

Finite Isometric Embeddings are excellently summarised by Blumenthal [**?**]. He uses the phrase *four-point property* to mean a space that is 4-embeddable in 3-dimensional Euclidean space: that is, that for any four objects in the original space it is possible to construct a distance-preserving tetrahedon. Wilson [**?**] shows various properties of such spaces, and Blumenthal points out that results given by Wilson, when combined with work by Menger [**?**], generalise to show that some spaces with the four-point property also have the n-point property: any n points can be isometrically embedded in an $(n-1)$-dimensional Euclidean

space (ℓ_2^{n-1}). In a later work, Blumenthal [**?**] shows that any space which is isometrically embeddable in a Hilbert space has the n-point property. This single result applies to many metrics, including Euclidean, Cosine, Jensen-Shannon and Triangular [**?**], and is sufficient for our purposes here.

Dimensionality Reduction aims to produce low-dimensional encodings of high-dimensional data, preserving the local structure of some input data. See [**?**,**?**] for comprehensive surveys on this topic.

The *Principal Component Analysis* (PCA) [**?**] is the most popular of the techniques for unsupervised dimensionality reduction. The idea is to find a linear transformation of n-dimensional to k-dimensional vectors ($k \leq n$) that best preserves the *variance* of the input data. Specifically, PCA projects the data along the direction of its first k principal components, which are the eigenvectors of the covariance matrix of the (centered) input data.

According to the *Johnson-Lindenstrauss Flattening Lemma* (JL) (see e.g. [**?**, pag. 358]), a projection can also be used to embed a finite set of n euclidean vectors into a k-dimensional euclidean space space ($k < n$) with a "small" distortion. Specifically the Lemma asserts that for any n-points of ℓ_2 and every $0 < \epsilon < 1$ there is a mapping into ℓ_2^k that preserves all the interpoint distances within factor $1 + \epsilon$, where $k = O(\epsilon^{-2} \log n)$. The low dimensional embedding given by the Johnson Lindenstrauss lemma is particularly simple to implement.

General metric spaces do not allow either PCA or JL as these require access to a coordinate space. Mao et al. [**?**] pointed out that multidimesional-methods can be indirectly applied to metric space by using the *pivot space* model. In that case each metric object is represented by its distance to a finite set of pivots.

In the general metric space context, perhaps the best known technique is *metric Multidimensional Scaling* (MDS) [**?**]. MDS aims to preserve *inter-point distances* using spectral analysis. However, when the number m of data points is large the classical MDS is too expensive in practice due to a requirement for $O(m^2)$ distance computations and spectral decomposition of a $m \times m$ matrix.

The *Landmark MDS* (LMDS) [**?**] is a fast approximation of MDS. LMDS uses a set of k *landmark* points to compute $k \times m$ distances of the data points from the pivots. It applies classical MSD to these points and uses a distance-based triangulation procedure to project the remaining data points.

LAESA [**?**] is a more tractable mechanism which has been used for metric filtering, rather than approximate search. n reference objects are somehow selected. For each element of the data, the distances to these points are recorded in a table. At query time, the distances between the query and each reference point are calculated. The table can then be scanned row at a time, and each distance compared; if, for any reference object p_i and data object s_j the absolute difference $|d(q, p_i) - d(s_j, p_i)| > t$, then from triangle inequality it is impossible for s_j to be within distance t of the query, and the distance calculation can be avoided.

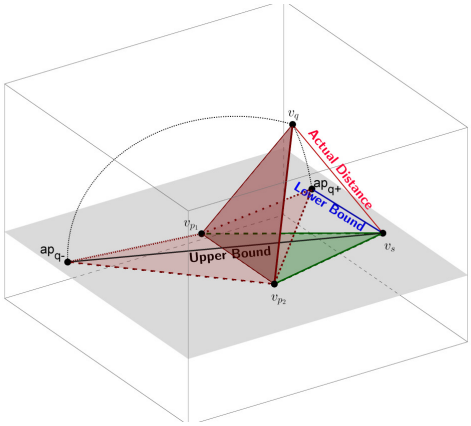

Fig. 1: Tetrahedral embedding of four points into 3D Euclidean space.

LAESA can be used as an efficient pre-filter for exact search when memory size is limited, and we make an experimental comparison with the new lower-bound mechanism we describe in this paper.

3 Upper and Lower Bounds from Simplexes

For any $(n+1)$ objects u_i in a supermetric space (U, d), there exists a simplex in ℓ_2^n where each vertex v_i corresponds to one object u_i and whose edge lengths correspond to distances in the original space, i.e. $\ell_2(v_i, v_j) = d(u_i, u_j)$.

We show now how this property can be used to give bounds on distances between two elements of U whose distance cannot be directly measured. This is useful in many different search paradigms where the bounds are required between an arbitrary elements $s_i \in S \subset U$ which has been pre-processed before a search, and an element $q \in U$ which is not known when the pre-processing occurs.

Our strategy is to choose a set of n reference points $P \subset U$, from which an isometric $(n-1)$-dimensional simplex σ is created. Now, given a further point $u \in U$, and all the distances $d(p_i, u)$, an n-dimensional simplex σ_u can be created by the addition of a single vertex to σ.

For simplicity, Figure 1 shows an ℓ_2^3 space into which four objects have been projected. Here we have only two reference points, p_1 and p_2. For each element a the notation v_a is used to denote a corresponding point in the ℓ_2^3 space. The distance $d(s, q)$ is not known; however the 4-point property means that the corresponding distance $\ell_2(v_s, v_q)$ must be able to form the final edge of a tetrahedron. From this Figure, the intuition of the upper and lower bounds on $d(s, q)$ is clear, through rotation of the triangle $v_{p_1} v_{p_2} v_q$ around the line $v_{p_1} v_{p_2}$ until it is coincident with the plane in which $v_{p_1} v_{p_2} v_s$ lies. The two possible orientations give the upper and lower bounds, corresponding to the distances between v_s and the two apexes ap_{q_-} and ap_{q_+} of the two possible planar tetrahedra.

The same intuition generalises into many dimensions. The inter-object distances within a set $\{p_i\}$ of n reference objects are used to form a *base* simplex σ_0, with vertices $v_{p_1}, \ldots, v_{p_n}$, in $(n-1)$ dimensions. This corresponds to the line segment $v_{p_1}v_{p_2}$ in the figure, which gives a two-vertex simplex in ℓ_2^1. The simplex σ_0 is contained within a hyperplane of the ℓ_2^n space, and the distances from object s to each p_i are used to calculate a new simplex σ_s, in ℓ_2^n, consisting of a new apex point v_s set above the base simplex σ_0. There are two possible positions in ℓ_2^n for v_s, one on either side of the hyperplane containing σ_0; we denote these as v_s^+, and v_s^- respectively. Now, given the distances between object q and all p_i, there also exist two possible simplexes for σ_q, with two possible positions for v_q denoted by v_q^+ and v_q^-.

The process of rotating a triangle around its base generalises to that of rotating the apex point of any simplex around the hyperplane containing its base simplex. Furthermore, the n-point property guarantees the existence of a simplex σ_1 in ℓ_2^{n+1} which preserves the distance $d(s,q)$ as $\ell_2(v_s, v_q)$. From these observations we immediately have the following inequalities:

$$\ell_2^n(v_s^+, v_q^+) \quad \leq \quad d(s,q) \quad \leq \quad \ell_2^n(v_s^+, v_q^-)$$

Proofs of the correctness of these inequalities are available in [**?**].

4 Constructing Simplexes from Edge Lengths

In this section, we show a novel algorithm for determining Cartesian coordinates for the vertices of a simplex, given only the distances between points. The algorithm is inductive, at each stage allowing the apex of an n-dimensional simplex to be determined given the coordinates of an $(n-1)$-dimensional simplex, and the distances from the new apex to each vertex in the existing simplex. This is important because, given a fixed base simplex over which many new apexes are to be constructed, the time required to compute each one is $\mathcal{O}(n)$ for n dimensions, whereas construction of the whole simplex is $\mathcal{O}(n^2)$

A simplex is a generalisation of a triangle or a tetrahedron in arbitrary dimensions. In one dimension, the simplex is a line segment; in two it is the convex hull of a triangle, while in three it is the convex hull of a tetrahedron. In general, the n-simplex of vertices $p_1, \ldots, p_{n+1}$ equals the union of all the line segments joining p_{n+1} to points of the $(n-1)$-simplex of vertices $p_1, \ldots, p_n$.

The structure of a simplex in n-dimensional space is given as an $n+1$ by n matrix representing the cartesian coordinates of each vertex. For example, the following matrix represents four coordinates which are the vertices of a tetrahedron in 3D space:

$$\begin{bmatrix} 0 & 0 & 0 \\ v_{2,1} & 0 & 0 \\ v_{3,1} & v_{3,2} & 0 \\ v_{4,1} & v_{4,2} & v_{4,3} \end{bmatrix}$$

For all such matrices Σ, the invariant that $v_{i,j} = 0$ whenever $j \geq i$ can be maintained without loss of generality; for any simplex, this can be achieved

Algorithm 1: nSimplexBuild

Input: $n+1$ points $p_1, \ldots, p_{n+1} \in (U, d)$
Output: n-dimensional simplex in ℓ_2^n represented by the matrix $\Sigma \in \mathbb{R}^{(n+1)\times n}$

$\Sigma = 0 \in \mathbb{R}^{(n+1)\times n}$;
if $n = 1$ **then**
  $\delta = d(p_1, p_2)$;
  $\Sigma = \begin{bmatrix} 0 \\ \delta \end{bmatrix}$;
  return Σ;
end
$\Sigma_{Base} = \text{nSimplexBuild}(p_1, \ldots, p_n)$;
$Distances = 0 \in \mathbb{R}^n$;
for $1 \leq i \leq n$ set $Distances[i] = d(p_i, p_{n+1})$;
$newApex = \text{ApexAddition}(\Sigma_{Base}, Distances)$;
for $1 \leq i \leq n$ and $1 \leq j \leq i-1$ set $\Sigma[i][j]$ to $\Sigma_{Base}[i][j]$;
for $1 \leq j \leq n$ set $\Sigma[n+1][j]$ to $newApex[j]$;
return Σ;

by rotation and translation within the Euclidean space while maintaining the distances among all the vertices. Furthermore, if we restrict $v_{i,j} \geq 0$ whenever $j = i - 1$ then in each row this component represents the *altitude* of the i^{th} point with respect to a base face represented by the matrix cut down from Σ by selecting elements above and to the left of that entry.

4.1 Simplex Construction

This section gives an inductive algorithm (Algorithm 1) to construct a simplex in n dimensions based only on the distances measured among $n+1$ points.

For the base case of a one-dimensional simplex (i.e. two points with a single distance δ) the construction is simply $\Sigma = \begin{bmatrix} 0 \\ \delta \end{bmatrix}$. For an n-dimensional simplex, where $n > 1$, an $(n-1)$-dimensional simplex is first constructed using the distances among the first n points. This simplex is used as a simplex base to which a new apex, the $(n+1)^{th}$ point, is added by the *ApexAddition* algorithm (Algorithm 2).

For an arbitrary set of objects $s_i \in S$, the apex $\phi_n(s_i)$ can be pre-calculated. When a query is performed, only n distances in the metric space require to be calculated to discover the new apex $\phi_n(q)$ in ℓ_2^n.

In essence, the *ApexAddition* algorithm is derived from exactly the same intuition as the lower-bound property explained earlier, at each stage lifting the final dimension out of the same hyperplane into a new dimension to capture the measured distances. Proofs of correctness for both the construction and the lower-bound property are available in [**?**].

Algorithm 2: ApexAddition

Input: A $(n-1)$-dimensional base simplex and the distances between a new (unknown) apex point and the vertices of the base simplex:

$$\Sigma_{\text{Base}} = \begin{bmatrix} 0 & & & \\ v_{2,1} & 0 & & \\ v_{3,1} & v_{3,2} & \ddots & \\ \vdots & & \ddots & 0 \\ v_{n,1} & & \cdots & v_{n,n-1} \end{bmatrix} \in \mathbb{R}^{n \times n-1}$$

$$Distances = \begin{bmatrix} \delta_1 & \cdots & \delta_n \end{bmatrix} \in \mathbb{R}^n$$

Output: The cartesian coordinates of the new apex point

$Output = \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^n$;
for $i = 2$ **to** n **do**
  $l = \ell_2(\Sigma_{Base}[i], Output)$;
  $\delta = Distances[i]$;
  $x = \Sigma_{Base}[i][i-1]$;
  $y = Output[i-1]$;
  $Output[i-1] = y - (\delta^2 - l^2)/2x$;
  $Output[i] = +\sqrt{y^2 - (Output[i-1])^2}$;
end
return $Output$

4.2 Bounds

Because of the method we use to build simplexes, the final coordinate always represents the altitude of the apex above the hyperplane containing the base simplex. Given this, two apexes exist, according to whether a positive or negative real number is inserted at the final step of the algorithm.

As a direct result of this observation, and those given in Section 3, we have the following bounds for any two objects s_1 and s_2 in the original space:

Let

$$\begin{aligned} \phi_n(s_1) &= (x_1, x_2, \ldots, x_{n-1}, x_n) \\ \phi_n(s_2) &= (y_1, y_2, \ldots, y_{n-1}, y_n) \end{aligned}$$

then

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad \leq \quad d(s_1, s_2) \quad \leq \quad \sqrt{\sum_{i=1}^{n-1}(x_i - y_i)^2 + (x_n + y_n)^2}$$

From the structure of these calculations, it is apparent that they are likely to converge rapidly around the true distance as the number of dimensions used becomes higher, as we will show in Section 5. It can also be seen that the cost of calculating both of these values together, especially in higher dimensions, is essentially the same as a simple ℓ_2 calculation.

Finally, we note that the lower-bound function is a proper metric, but the upper-bound function is not even a semi-metric: even although it is a Euclidean distance in the apex space, one of the domain points is constructed by reflection across a hyperplane and thus the distance between a pair of identical points is in general non-zero.

5 Measuring Distortion

We define distortion for an approximation (U', d') of a space (U, d) mapped by a function $f : U \to U'$ as as the smallest D such that, for some scaling factor r

$$r \cdot d'(f(u_i), f(u_j)) \quad \leq \quad d(u_i, u_j) \quad \leq \quad D \cdot r \cdot d'(f(u_i), f(u_j))$$

We have measured this for a number of different spaces, and present results over the SISAP *colors* benchmark set which are typical and easily reproducible. Summary results are shown in Figure 2.

In each case, the X-axis represents the number of dimensions used for the representation, with the distortion plotted against this. For Euclidean distance, there are two entries for n-simplex: one for randomly-selected reference points, and the other where the choice of reference points is guided by the use of PCA. In the latter case we select the first n principal components (eigenvectors of the covariance matrix) as pivots.

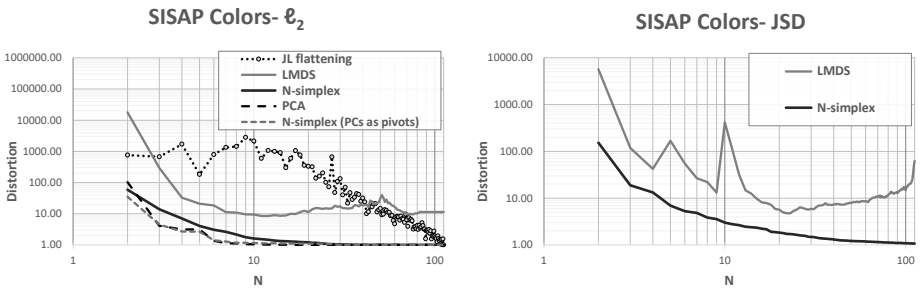

Fig. 2: Distortion measurements for various dimensionality reduction strategies for the *colors* data set. The left figure gives measurements for Euclidean distance, the right for Jensen-Shannon distance where only LMDS and n-simplex are applicable. The *colors* data set has 112 physical dimensions.

It can be seen that n-simplex outperforms all other strategies except for PCA, which is not applicable to non-Euclidean spaces. LMDS is the only other mechanism applicable to general metric spaces[4]; this is a little more expensive than n-simplex to evaluate, and performs relatively badly. The comparison with JL is a slightly unfair, as the JL lemma applies only for very high dimensions in an evenly distributed space; we have also tested such spaces, and JL is still out-performed by n-simplex, especially at lower dimensions.

The distortion we show here is only for the lower-bound function of n-simplex. We have measured the upper-bound function also, which gives similar results. Unlike the lower-bound, the upper-bound is not a proper metric; however for non-metric approximate search it should be noted that the mean of the lower- and upper-bound functions give around half the distortion plotted here.

The implications of these results for exact search should be noted. For Euclidean search, the distortion has dropped to almost zero at between 20 and 30 dimensions, implying the possibility of accurate search using data which is less than one-quarter of the original size. For Jensen-Shannon, more dimensions will be required, but the cost of the ℓ_2 metric required to search the compressed space is around one-hundredth the cost of the original metric. In the next section we present experimental results consistent with these observations.

6 Exact Search: Indexing with n-Simplex

The simplex-building mechanism, along with the observations of upper and lower bounds, might be used in many different metric search contexts. Here, we examine only one of these to demonstrate the potential.

To this end we examine the use of n-simplex in the context of exact search, using the lower and upper-bound properties. Any such mechanism can be viewed

[4] In [**?**] the authors note it works better for some metrics than for others; in our understanding, it will work well only for spaces with the n-point property.

as similar to LAESA [**?**], in that there exists an underlying data structure which is a table of numbers, n per original object, with the intention of using this table to exclude candidates which cannot be within a given search threshold.

In both cases, n reference objects are chosen from the space. For LAESA, each row of the table is filled, for one element of the data, with the distances from the candidate to each reference object. For n-simplex, each row is filled for one element of the data with the Cartesian coordinates of the new apex formed in n dimensions by applying these distances to an $(n-1)$-dimensional simplex formed from the reference objects.

The table having been established, a query notionally proceeds by measuring the distances from the query object to each reference point object. In the case of LAESA, the metric for comparison is Chebyshev: that is, if any pairwise difference is greater than the query threshold, the object from which that row was derived cannot be a solution to the query. For n-simplex, the metric used is ℓ_2: that is, if the apex represented in a row is further than the query threshold from the apex generated from the query, again the object from which that apex was derived cannot be a solution to the query.

In both cases, there are two ways of approaching the table search. It can be performed sequentially over the whole table, in which case either metric can be terminated within a row if the threshold is exceeded, without continuing to the end of the row. Alternatively the table can itself be re-indexed using a metric index. Although this compromises the amount of space available for the table itself, it may avoid many of the individual row comparisons.

In the context of re-indexing we also note that, in the case of n-simplex, the Euclidean metric used over the table rows itself has the four-point property, and so the Hilbert Exclusion property as described in [**?**] may be used.

In all cases the result is a filtered set of candidate objects which is guaranteed to contain the correct solution set. In general, this set must be re-checked against the original metric, in the original space. For n-simplex however the upper-bound condition is checked first; if this is less than the query threshold, then the object is guaranteed to be an element of the result set with no further check required.

6.1 Experiment - SISAP *colors*

Any such mechanism will perform differently over data sets with different characteristics and we cannot yet provide a full survey. To give useful comparisons with other studies in the literature, we apply the techniques to the SISAP *colors* [**?**] data set, using three different supermetrics: Euclidean, Cosine, and Jensen-Shannon[5]. We chose this data set because (a) it has only positive values and is therefore indexable by all of the metrics, and (b) it shows an interesting non-uniformity, in that its intrinsic dimensionality [**?**] for all metrics is much less than its physical dimensionality (112). It should thus give an interesting "real world" context to assess the relative value of the different mechanisms. Although it is a relatively small set, further experiments performed on much larger sets with

[5] For precise definitions of the non-Euclidean metrics used, see [**?**].

Table 1: Elapsed Times - SISAP *colors*, Euclidean distance.
All times are in seconds, for executing 11268 queries over 101414 data. The $Tree$ times are independent of the row as reference points are not used.

	$t_0 = 0.051768$					$t_1 = 0.082514$					$t_2 = 0.131163$				
Dims	L_{seq}	L_{rei}	N_{seq}	N_{rei}	$Tree$	L_{seq}	L_{rei}	N_{seq}	N_{rei}	$Tree$	L_{seq}	L_{rei}	N_{seq}	N_{rei}	$Tree$
5	18.6	28.0	13.8	5.8	5.5	33.4	80.9	22.4	29.0	18.1	56.2	201.6	34.9	70.4	54.4
10	17.7	22.1	15.0	3.3		30.3	67.9	20.3	14.7		58.1	220.3	25.5	50.6	
15	16.3	15.2	14.6	**3.0**		26.7	59.7	20.2	12.1		45.8	159.5	**24.4**	44.7	
20	19.0	16.3	18.9	3.3		28.2	56.6	19.4	**11.5**		46.8	189.3	27.8	48.3	
25	22.5	16.9	20.4	3.4		27.4	56.8	22.3	13.4		45.5	167.5	26.2	40.1	
30	20.9	16.8	20.4	3.5		28.6	57.3	24.5	13.6		45.9	181.2	28.5	45.1	
35	22.0	16.4	21.3	3.9		28.7	65.0	22.5	13.9		43.9	163.0	31.2	44.9	
40	23.1	17.3	22.1	4.0		28.8	55.9	22.8	14.3		49.4	180.5	34.2	46.1	
45	22.5	18.7	22.2	4.4		32.0	61.5	27.7	15.0		48.5	169.8	37.1	44.9	
50	21.3	17.1	18.9	4.5		32.0	59.0	24.0	15.5		55.2	207.6	34.5	45.3	

different properties give quite consistent results, which we do not have space to report here.

For Euclidean distance, we used the three benchmark thresholds; for the other metrics, we chose thresholds that return around 0.01% of the data. In all cases the first 10% of the file is used to query the remaining 90%. Pivots are randomly-selected both for LAESA and n-simplex approach.

For each metric, we tested different mechanisms with different allocations of space: 5 to 50 numbers per data element, thus the space used per object is between 4.5% and 45% of the original. All results reported are for exact search, that is the initial filtering is followed by re-testing within the original space where required. Five different mechanism were tested, as follows:

sequential LAESA (L_{seq}) each row of the table is scanned sequentially, each element of each row is tested against the query and that row is abandoned if the absolute difference is greater than the threshold.

reindexed LAESA (L_{rei}) the data in the table is indexed using a monotone hyperplane tree, searched using the Chebyshev metric.

sequential n-simplex (N_{seq}) each row of the table is scanned sequentially, for each element of each row the square of the absolute difference is added to an accumulator, the row is abandoned if the accumulator exceeds the square of the threshold, and the upper-bound is applied if the end of the row is reached before re-checking in the original space.

reindexed n-simplex (N_{rei}) the data in the table is indexed using a monotone hyperplane tree using the Hilbert Exclusion property, and searched using the Euclidean metric; the upper-bound is applied for all results, before re-checking in the original space.

normal indexing ($Tree$) the space is indexed using a monotone hyperplane tree with the Hilbert Exclusion property, without the use of reference points.

Table 2: Elapsed Times - SISAP *colors* with Cosine and Jensen-Shannon distances, and a 30-dimensional generated Euclidean space.
All times are in seconds. The generated Euclidean space is evenly distributed in $[0, 1]^{30}$, and gives the elapsed time for executing 1,000 queries against 9,000 data, with a threshold calculated to return one result per million data (t=0.7269)

	SISAP *colors*										30-dim ℓ_2^{30}					
	Cosine (t=0.042)					Jensen-Shannon (t=0.135)										
Dims	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*Tree*	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*Tree*	Dims	L_{seq}	L_{rei}	N_{seq}	N_{rei}	*Tree*
5	10.3	4.5	8.8	1.0	3.1	248.4	335.5	61.9	65.5	124.8	3	0.5	2.5	0.5	1.6	1.4
10	9.8	3.4	10.4	0.8		155.3	233.2	29.0	29.3		6	0.5	2.3	0.5	1.8	
15	12.7	2.4	11.7	**0.7**		103.5	163.2	22.3	17.2		9	0.5	2.4	0.4	1.3	
20	16.5	2.8	16.7	**0.7**		95.7	162.8	23.8	**14.7**		12	0.5	2.6	0.3	1.2	
25	17.9	2.8	17.7	0.8		87.2	155.6	25.9	16.1		15	0.5	2.8	0.3	1.0	
30	18.1	2.6	17.4	0.9		67.7	130.4	27.0	16.5		18	0.6	3.4	0.3	1.0	
35	17.7	3.1	17.1	1.1		69.6	136.3	27.9	17.2		21	0.6	3.3	**0.2**	1.1	
40	18.1	3.0	18.1	1.0		62.4	131.2	27.8	17.1		24	0.7	2.9	**0.2**	1.1	
45	17.4	2.7	18.2	1.1		61.1	133.4	29.7	18.4		27	0.7	3.5	0.3	1.2	
50	17.6	3.5	17.3	1.4		58.3	130.4	30.6	18.6		30	0.7	3.5	0.3	1.4	

The monotone hyperplane tree is used as, in previous work, this has been found to be the best-performing simple indexing mechanism for use with Hilbert Exclusion.

Measurements different figures are measured for each mechanism: the elapsed time, the number of original-space distance calculations performed and, in the case of the re-indexing mechanisms, the number of re-indexed space calculations. All code is available online for independent testing[6].

The tests were run on a 2.8 GHz Intel Core i7, running on an otherwise bare machine without network interference. The code is written in Java, and all data sets used fit easily into the Java heap without paging or garbage collection occurring.

Results As can be seen in Table 1, N_{rei} consistently and significantly outperforms the normal index structure at between 15 and 25 dimensions, depending on the query threshold. It is also interesting to see that, as the query threshold increases, and therefore scalability decreases, N_{seq} takes over as the most efficient mechanism, again with a "sweet spot" at 15 dimensions.

Table 2 shows the same experiment performed with Cosine and Jensen-Shannon distances. In these cases, the extra relative cost saving from the more expensive metrics is very clear, with relative speedups of 4.5 and 8.5 times respectively. In the Jensen-Shannon tests, the relatively very high cost of the metric evaluation to some extent masks the difference between N_{seq} and N_{rei}, but we note that the latter maintains scalability while the former does not. Finally, in the essentially intractable Euclidean space, with a relatively much smaller search threshold, N_{seq} takes over as the fastest mechanism.

[6] https://richardconnor@bitbucket.org/richardconnor/metric-space-framework.git

Table 3: Distance Calculations Performed in Original and Re-indexed Space (figures given are thousands of calculations per query)

	Euclidean (t=0.051768)					Jensen-Shannon (t=0.135)				
	Original Space			Re-indexed		Original Space			Re-indexed	
Dims	L	N	*Tree*	L_{rei}	N_{rei}	L	N	*Tree*	L_{rei}	N_{rei}
5	2.75	0.38	1.48	5.28	1.76	12.77	2.29	5.97	18.40	6.91
10	1.33	0.05	1.48	4.40	1.23	7.81	0.58	5.97	19.66	6.32
15	0.57	0.04	1.48	3.24	1.13	4.62	0.16	5.97	15.46	4.99
20	0.51	0.03	1.48	3.42	1.15	3.89	0.11	5.97	15.85	4.80
25	0.43	0.04	1.48	3.15	1.18	3.65	0.09	5.97	14.88	4.87
30	0.37	0.04	1.48	3.02	1.21	2.53	0.08	5.97	13.83	4.70
35	0.34	0.04	1.48	2.85	1.31	2.59	0.08	5.97	13.56	4.86
40	0.33	0.04	1.48	2.95	1.29	2.14	0.08	5.97	13.48	4.64
45	0.31	0.05	1.48	2.82	1.32	1.95	0.08	5.97	13.74	4.89
50	0.27	0.05	1.48	2.57	1.33	1.83	0.08	5.97	12.63	4.87

Scalability Table 3 shows the actual number of distance measurements made, for Euclidean and Jensen-Shannon searches of the *colors* data. The number of calls required in both the original and re-indexed spaces are given. Note that original-space calls are the same for both table-checked and re-indexed mechanisms; the number of original-space calls include those to the reference points, from which the accuracy of the n-simplex mechanism even in small dimensions can be appreciated. By 50 dimensions almost perfect accuracy is achieved for Euclidean search 50 original-space calculations are made, but in fact even at 10 dimensions almost every apex value can be deterministically determined as either a member or otherwise of the solution set based on its upper and lower bounds. At 20 dimensions, only 10 elements of the 101414-element data set have bounds which straddle the query threshold. This indeed reflects the results presented in Figure 2 where it is shown that for $n \geq 20$ the n-simplex lower bound is practically equivalent to the Euclidean distance to search *colors* data.

Equally interesting is the number of re-indexed distance measurements. This requires further investigation: for n-simplex, these are generally less than for the original space. This seems to hold for all data other than perfectly evenly-distributed (generated sets), for which the scalability is the same. The implication is that the re-indexed metric has better scalability properties than the original, although we would have expected indexing over the lower-bound function to be less, rather than more, scalable.

7 Conclusions and Further Work

Based on observations made over half a century ago, we have observed that a class of useful metric spaces have the n-point property. We have discovered a practical application for this previously abstract knowledge, by showing that irregular simplexes of any dimension can be constructed from only their edge

lengths. This then allows upper and lower bounds to be calculated for any two objects, when the only knowledge available is their respective distances to a fixed set of reference objects.

There are a number of ways in which this knowledge can be used towards efficient search for suitable spaces. We have so far examined only one in detail, where a Euclidean space is extracted and used to pre-filter exact search. Over the benchmark SISAP *colors* data set, for some different metrics, this technique gives the best-recorded performance for exact search. However we believe the real power of this technique will emerge with huge data sets and more expensive metrics, and is yet to be experienced.

Acknowledgements The work was partially funded by Smart News, "Social sensing for breaking news", co-funded by the Tuscany region under the FAR-FAS 2014 program, CUP CIPE D58C15000270008.