An Empirical Evaluation of Intrinsic Dimension Estimators *

Cristian Bustos[1], Gonzalo Navarro[2], Nora Reyes[1], and Rodrigo Paredes[3]

[1] Dpto. de Informática, Universidad Nacional de San Luis
[2] Center of Biotechnology and Bioengineering, Department of Computer Science, University of Chile
[3] Departamento de Ciencias de la Computación, Universidad de Talca, Chile
{cjbustos,nreyes}@unsl.edu.ar, gnavarro@dcc.uchile.cl, raparede@utalca.cl

Abstract. We study the practical behavior of different algorithms that aim to estimate the intrinsic dimension (ID) in metric spaces. Some of these algorithms were specifically developed to evaluate the complexity of searching in metric spaces, based on different theories related to the distribution of distances between objects on such spaces. Others were originally designed for vector spaces only, and have been extended to general metric spaces. To empirically evaluate the fitness of various ID estimations with the actual difficulty of searching in metric spaces, we compare one representative of each of the broadest families of metric indices: those based on pivots and those based on compact partitions. Our preliminary conclusions are that Fastmap and the measure called Intrinsic Dimensionality fit best their purpose.

1 Introduction

Similarity search in metric spaces has received much attention due to its applications in many fields, ranging from multimedia information retrieval to machine learning, classification, and searching the Web. While a wealth of practical algorithms exist to handle this problem, it has been often noted that some datasets are intrinsically harder to search than others, no matter which search algorithms are used. An intuitive concept of "curse of dimensionality" has been coined to denote this intrinsic difficulty, but a clear method to measure it, and thus to predict the performance of similarity searching in a space, has been elusive.

The similarity between a set of objects $\mathbb{U}$ is modeled using a *distance function* (or *metric*) $d : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}^+ \cup \{0\}$ that satisfies the properties of triangle inequality, strict positivity, reflexivity, and symmetry. In this case, the pair $(\mathbb{U}, d)$ is called a *metric space* [6, 24, 21].

In some applications, the metric spaces are of a particular kind called "vector spaces", where the elements consist of D coordinates of real numbers and the distance is some Minkowski metric. Many works exploit the geometric properties of vector spaces, but they usually cannot be extended to general metric spaces, where the only available information is the distance between objects. Since in

* Partially funded by basal funds FB0001, Conicyt, Chile and Fondecyt grant 1131044, Chile.

most cases the distance is very expensive to compute, the main goal when searching in metric spaces is to reduce the number of distance evaluations. In contrast, vector space operations tend to be cheaper and the primary goal when searching them is to reduce the CPU cost or the number of I/O operations carried out.

Similarity queries are usually of two types. For a given database $S \subseteq \mathbb{U}$ with size $|S| = n$, $q \in \mathbb{U}$ and $r \in \mathbb{R}^+$, the *range query* $(q, r)_d$ returns all the objects of S at distance at most r from q, whereas the *nearest neighbor query* $kNN_d(q)$ retrieves the k elements of S that are closest to q.

A naïve way to answer similarity queries is to compare all the database elements with the query q and return those that are close enough to it. This *brute force* approach is too expensive for real applications. Research has then focused on ways to reduce the number of distance computations performed to answer similarity queries. There has been significant progress around the idea of building an *index*, that is, a data structure that allows discarding some database elements without explicitly comparing them to q.

In uniform vector spaces, the *curse of dimensionality* describes the well-known exponential increase of the cost of all existing search algorithms as the dimension grows. Non-uniform vector spaces may be easier to search than uniform ones, despite having the same *explicit* dimensionality. The phenomenon also extends to general metric spaces despite their absence of coordinates: some spaces are intrinsically harder to search than others. This has lead to the concept of *intrinsic dimensionality (ID)* of a metric space, as a measure of the difficulty of searching it. A reliable measure of ID has been elusive, despite the existence of several formulas.

Computing the ID of a metric space is useful, for example, to determine whether it is amenable to indexing at all. If the ID is too high, then we must just resort to brute-force solutions or to approximate search algorithms (which do not guarantee to find the exact answers). Even when exact indexing is possible, the ID helps decide which kind of index to use. For example, pivot-based methods work better on lower dimensions, whereas compact partiioning methods outperform them in higher dimensions [6].

In this work we aim to empirically study the fitness of various ID measures to predict the search difficulty of metric space searching. Some measures were specifically developed for metric spaces, based on different theories related to the distribution of distances between objects. Others were originally designed for vector spaces and have then been adapted to general metric spaces. We chose various synthetic and real-life metric spaces and two indexing methods that are representatives of the major families of indices: one based on pivots one and another based on compact partitions. Our comparison between real and estimated search difficulty yield, as preliminary conclusions, that *Fastmap* [10] and the formula by Chávez et al. [6] are currently the best predictors in practice.

2 Intrinsic Dimension Estimators for Vector Spaces

There are several interesting applications where the data are represented as D-dimensional vectors in $\mathbb{R}^D$. For instance, in pattern recognition applications,

objects are usually represented as vectors [14]. So, data are embedded in $\mathbb{R}^D$, even though this does not imply that its *intrinsic* dimension is D.

There are many definitions of ID. For instance, the ID of a given dataset is the minimum number of free variables needed to represent the data without loss of information [2]. In general terms, a dataset $\mathbb{X} \subseteq \mathbb{R}^D$ has ID $M \leq D$, if its elements fall completely within an M-dimensional subspace of $\mathbb{R}^D$ [12]. Another intuitive notion is the logarithm of the search cost, as in many cases this cost grows exponentially with the dimension.

Even in vector spaces, there are many reasons to estimate the ID of a dataset. Using more dimensions (more coordinates in the vectors) than necessary can bring several problems. For example, the space to store the data may be an issue. A dataset $\mathbb{X} \subseteq \mathbb{R}^D$ with $|\mathbb{X}| = n$ requires to store $n \times D$ real coordinates. Instead, if we know that the ID of $\mathbb{X}$ is $M \leq D$, we can map the points to $\mathbb{R}^M$ and just store $n \times M$ real coordinates. The CPU cost to compute a distance is also reduced. This can in addition help identify the important dimensions in the original data.

There are two approximations to estimate the ID of a vector space [14, 2], namely, *local* and *global* methods. The local ones make the estimation by using the information contained in sample neighborhoods, avoiding the data projection over spaces of lower dimensionality. The global ones deploy the dataset over an M-dimensional space using all the dataset information.

In this work we focus on global ID estimators. That is, we consider all the dataset information to estimate the ID as accurately as possible. Global methods can be split into three families: projection techniques, multidimensional scaling methods, and fractal based methods. The last two are more suitable to extend to metric spaces, so we have selected and adapted some representatives of these groups.

3 Intrinsic Dimension Estimators for Metric Spaces

In this section we analyze various methods to estimate the ID of vector spaces and others to general metric spaces. We discuss how to adapt the former to the case of general metric spaces. Note that, since multidimensional spaces are a particular case of metric spaces, our estimators can also be applied to obtain the ID of D-dimensional vector spaces.

3.1 Fractal Based Methods

Unlike other families, fractal based methods can estimate non-integer ID values. The most popular techniques of this family are *Box Counting* [17], which is a simplified version of the *Haussdorff dimension* [9, 18], and *Correlation* [3].

The dimension estimation by Box Counting D_B of a set $\Omega \subseteq \mathbb{R}^D$ is defined as follows: if $v(r)$ is the number of boxes of size r needed to cover Ω, then $D_B = \lim_{r \to 0} \frac{\ln(v(r))}{\ln\left(\frac{1}{r}\right)}$.

In this method, the boxes are multidimensional regions of side r on each dimension (that is, they are hypercubes of side r). Regrettably, even though

efficient algorithms have been proposed, the Box Counting dimension can be computed only for low dimension datasets, because its algorithmic complexity grows exponentially with the dimension.

Estimating the dimension by Correlation is an alternative to Box Counting. It is defined as follows. Let $\Omega = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^D$ and the correlation integral $C_m(r) = \lim_{n\to\infty} \frac{2}{n(n-1)} \sum_{1\le i<j\le n} I(||x_j - x_i|| < r)$, where $I(\cdot)$ is the indicator function. Intuitively, $C_m(r)$ is the fraction of object pairs whose distance is lower than r. So, the dimension estimation by Correlation D_C is $D_C = \lim_{r\to 0} \frac{\ln(C_m(r))}{\ln r}$.

The most popular method to estimate the dimension by Correlation and Box Counting is the log-log plot. It consists in plotting $\ln(C_m(r))$ versus $\ln(r)$. The dimension by Correlation is the slope of the linear section of the curve. To estimate the dimension by Box Counting we replace $\ln(C_m(r))$ by $\ln(v(r))$.

In the general case of metric spaces, we do not have coordinates in general. Thus, to adapt the Box Counting method, we consider *balls* of radius r, that is, the set of objects within a distance r from a reference object o. We randomly pick the reference objects from the dataset, and count the number $B(r)$ of balls of radius r needed to cover the dataset. To do so, we use the *List of Clusters (LC)* index [5], whose code is available from SISAP [11], with the variant of fixed radius and centers chosen at random. Then the ID is just the length of the LC.

To estimate the dimension by Box Counting, which in this case is Ball Counting, we replace $\ln(v(r))$ by $\ln(B(r))$, plot $\ln(B(r))$ versus $\ln\left(\frac{1}{r}\right)$ in log-log and obtain the slope of the linear section of the curve by using linear regression with least squares over the experimental data $\left(\ln(B(r)), \ln\left(\frac{1}{r}\right)\right)$.

3.2 Distance Exponent

Traina et al. [22] discuss the problem of the selectivity estimation for range queries in real-world metric spaces, including spatial or multidimensional datasets as special cases. Their main finding is that several datasets follow the so-called *Power Law*. They call *Distance Exponent* the exponent of the power law, and show how to use it to derive formulas for estimating the selectivity of range queries. For instance, the number of objects relevant to the query, the number of I/Os to answer the query when the data is stored on disk, the amount of time needed to answer the query, and so on.

To find a formula that estimates the number of neighbors of objects within a distance r in a n-objects dataset, they introduce the following notions: (i) the *Distance Plot* of a metric set is the number of object pairs at distance at most r versus the distance r, and both axes are drawn in logarithmic scale; and (ii) the *Distance Exponent* is the slope of the line that better fits the distance plot in case it is linear for a range of scales. Using these two notions, they define the *Distance Law*.

Distance Law - *Given a dataset of n objects from a metric space with distance function $d(x, y)$, the average number of distances lower than a radius r follows a power law; that is, the average number of neighbors $\overline{nb}(r)$ within a distance r is proportional to $r^{\mathcal{D}}$. Formally, $N \cdot \Phi(r) = \overline{nb}(r) \propto r^{\mathcal{D}}$.*

If a dataset has a metric to evaluate the distance between every object pair, then this plot can always be drawn. They show that the distance plot has an almost linear behavior for many databases, both real and synthetic. Building the distance plot requires $O(n^2)$ distance computations. To reduce this cost, $\overline{nb}(r)$ is estimated using an index [22], in particular the *M-tree* [7]. Since in this work we are only interested in comparing the different ID measures, indexing the space is not necessary and we compute $\overline{nb}(r)$ directly, considering a reference object chosen at random from the dataset. We only determine the number of elements at distance r from that object. The result is averaged over various choices for the object.

3.3 Fastmap

This method arises from the proposal [13] of a fast algorithm to map objects of any metric space onto points of a k-dimensional space (k being defined by the user), so that the dissimilarities are preserved. Its goal is to speed up searches in traditional or multimedia databases.

To do so, the objects are mapped onto the k-dimensional space using k feature extraction functions, provided by domain experts [13]. The main issue is how to define such feature extraction functions. For example, in the metric case of strings with the edit distance [16], it is not clear which features can be considered.

For a domain expert, it is generally easier to provide a distance function to compare objects than to provide feature extraction functions. *Fastmap* [10] is a generalization of the original method [13], where the objects are mapped using only a distance function.

Fastmap finds, given a dataset of n objects from a metric space $(\mathbb{U}, d)$, n image points in a k-dimensional target space, such that the distances between the objects in the original space are preserved as much as possible in the target space.

For evaluating the dissimilarity preservation in the target space, a *stress* function is defined as follows, $stress^2 = \frac{\left(\sum_{i,j}(\hat{d}_{ij}-d_{ij})^2\right)}{\left(\sum_{i,j} d_{ij}^2\right)}$, where d_{ij} is the dissimilarity measure (the distance of the original space) between objects o_i and o_j, and $\hat{d}_{ij}$ is the Euclidean distance between their respective images p_i and p_j. The stress function gives the relative error that the distances in the target space suffer on average after the transformation. Fastmap begins with an estimation that is iteratively improved, until no additional improvement is possible.

In the metric case, we can assume that we have the $n \times n$ matrix of distances between all the dataset objects, and Fastmap must find n points in the k-dimensional space whose Euclidean distances are close to the original matrix of $n \times n$ distances. The crux is to assume that objects are points in some m-dimensional space, with unknown m, and to project these points over k mutually orthogonal directions. The challenge is to compute all these projections using only the distance matrix. Fastmap projects the objects over carefully selected lines. It chooses two objects o_a and o_b, and considers the "line" passing through them in the original space. The projections of the objects over this line are obtained using the *cosine law*:

Theorem 1 (*Cosine Law*) Any triangle $o_a \overset{\triangle}{o_i} o_b$ satisfies:

$$d(o_b, o_i)^2 = d(o_a, o_i)^2 + d(o_a, o_b)^2 - 2x'_i d(o_a, o_b). \quad (1)$$

Eq. 1 can be solved for x'_i to compute the projection of object o_i with the formula $x'_i = \frac{d(o_a,o_i)^2+d(o_a,o_b)^2-d(o_b,o_i)^2}{2d(o_a,o_b)}$.

Thus, the input of Fastmap is a set S of size n and, in each iteration, it computes the coordinates of all the n objects over the new axis. So, after k iterations, it produces a k-dimensional target space S' where each object $o_i \in S$ is mapped to a k-coordinate vector $p_i = (x'_{i,1}, x'_{i,2}, \ldots, x'_{i,k}) \in S'$, where $x'_{i,j}$ is the j-th projection of the image p_i of the object o_i.

In our case, we want to estimate the number of projections needed so that the target space reaches a mapping with a small enough *stress*, that is, preserving the distances sufficiently well. Thus, we modify the Fastmap algorithm so that it computes the *stress* of the target space after each new dimension is added. If the difference between the current and the previous *stress* values is significative, we compute another projection (thus increasing the dimensionality of the target space). Otherwise, the current dimension of the target space is reported as the estimation of the ID of the original metric space.

3.4 Intrinsic Search Difficulty

Chávez et al. [6] introduced a measure of the intrinsic complexity of searching in general metric spaces. It is easy to estimate and independent of the search algorithm.

Several authors [1, 4, 8] have proposed to use the *distance histogram* to characterize the hardness of searching in arbitrary metric spaces, yet the cost was tailored to a specific index. Instead, this measure [6] depends only on the histogram and not on any assumption on the indexing method.

The intuition behind this measure is that, in random vectors in D dimensions, the histogram has a larger mean μ and a smaller variance σ^2 as D increases. In fact, it holds $D = c \cdot \mu^2/\sigma^2$ for some constant c [23]. Thus, the same formula could be used to estimate a dimension D from the mean and variance of the histogram of distances in a general metric space. We do not have the histogram of the whole universe $\mathbb{U}$, but we can approximate it using the histogram of the dataset $S \subset \mathbb{U}$.

Definition: Let μ be the mean and σ^2 be the variance of the histogram of distances of a metric space. Then, its *intrinsic search difficulty* is $\rho = \frac{\mu^2}{2\sigma^2}$.

An obvious advantage of this measure, which has contributed to its popularity, is that it is easy to compute from a reasonable sampling of pairs in S. Other techniques require more complex and expensive computations.

Pestov [19] presents a system of three axioms an intrinsic dimension function must satisfy. He proves that the intrinsic dimension measure ρ satisfies a weak version of these axioms. Later [20], he introduces a set of goals an intrinsic dimension function should fulfill, and ρ achieves many of them.

As the measure ρ has been designed for general metric spaces, we use it as is. We consider the dataset S and we compute all the distances $d(x, y), \forall\ x, y \in S$. Then we compute the average $\mu = \frac{1}{n^2} \sum_{x,y \in S} d(x, y)$ and the variance $\sigma^2 = \frac{1}{n^2} \sum_{x,y \in S} (d(x, y) - \mu)^2$. Finally, we obtain the value of $\rho = \frac{\mu^2}{2\sigma^2}$ and report it as the ID of the metric space.

4 Experimental Results

We evaluate experimentally the four ID estimators described, on general metric spaces. We consider two kinds of metric spaces, depending on the data source:

Synthetic: these are spaces generated artificially so that they present some interesting characteristic to be evaluated. For instance, uniformly distributed vectors in $\mathbb{R}^D$ with known dimension.

Real world: these are metric spaces obtained from real-world applications. For instance, a feature vectors space of images obtained from a NASA image set.

4.1 Synthetic Metric Spaces

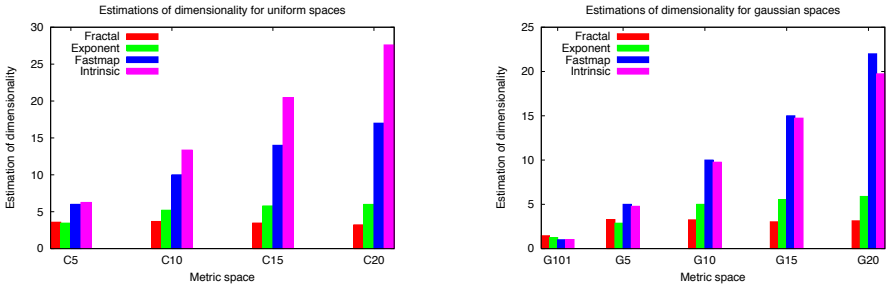
These are vector spaces with Euclidean distance. They are treated as metric spaces, as we do not consider the coordinate information. A first set is formed by vectors with uniform distribution, so that the representational dimension matches the ID. Here we can test the estimators in a case where the ID is known. A second set is formed by vectors with Gaussian distribution, so that the representational dimension is greater than the ID (the more clustered is the space, the lower is the ID). The distance is also Euclidean. Here we aim to check whether the estimators give lower values as the ID decreases.

Uniformly Distributed Vectors with Euclidean Distance We generate four datasets of 100,000 vectors uniformly distributed in the unitary cube $[0, 1)^D$, with $D = 5$, 10, 15 and 20. The spaces are called C5, C10, C15 and C20, respectively.

Fig. 1(a) depicts the estimations for these four metric spaces. As it can be seen, the Fractal estimator (Ball counting) is insensitive to the correct dimension. The Distance Exponent increases with D, but not proportionally. The other two estimations grow at the same rate of D, with Fastmap matching it almost perfectly and Intrinsic Search Difficulty showing a consistent factor multiplying D.

Search degradation as ID grows. To verify that the dataset ID is responsible of the search degradation, we pick C5 and extend its vectors with zeroes to produce spaces with 10, 15 and 20 representational dimensions, and study the search performance over it.

We perform 10 executions of the algorithms, building the index with 90% of the database elements, and reserving the remaining 10% (chosen at random) for the queries. So, the query objects do not belong to the index. We average



(a) Uniform spaces. (b) Gaussian spaces.

Fig. 1. Comparison of dimensionality estimations for synthetic metric spaces.

the results over the 10 executions. In each execution, the objects in the metric space are permuted at random. Therefore, each of the 10 indices uses a different dataset S, and the query objects are also different.

We use a pivot index and a compact partition index. For the pivot index family, we use the generic pivot algorithm. We choose at random a set of pivots $\mathcal{P} = \{P_1, P_2, \ldots, P_k\} \subset S$ of size $|\mathcal{P}| = k$. We store the kn distances between pivots and objects, and use them to filter out candidates using the triangle inequality. For each space, we experimentally determine the number of pivots that obtains the best search performance. Thus, the results shown for each case correspond to the best possible ones for this method, in the corresponding metric space.

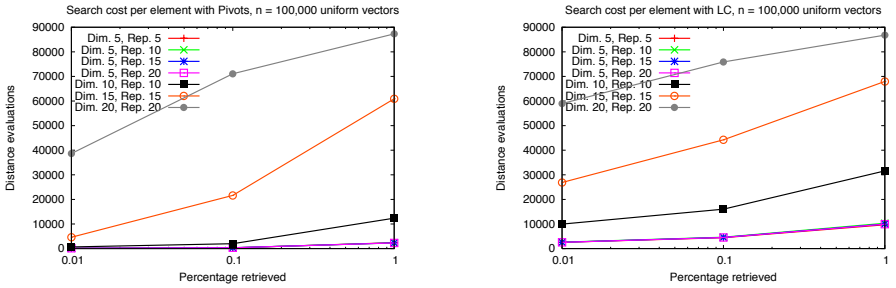
For the case of compact partition based algorithms, we consider the LC, which is one of the best indexes for medium and high dimensions [5]. We use the LC variant that has a maximum size for each cluster. For each metric space considered, we experimentally determine the cluster size whose perfomance is the best, and this is the result shown in the plots.

In Fig. 2, we show the cost of range queries retrieving 0.01%, 0.1% and 1% of the vector dataset per query, using the generic pivot index (Fig. 2(a)) and the LC (Fig. 2(b)). These results are compared with the ones for searching C10, C15 and C20. Both plots show that the four spaces of ID 5 overlay each other (independently of the representational dimension of the space), while the curves for C10, C15 and C20 show the usual degradation.

Gaussian Distributed Vectors with Euclidean Distance We generate 100,000 vectors in $\mathbb{R}^D$ with mean $\mu = 1$ and variance $\sigma^2 = 0.1$, for $D = 5$, 10, 15 and 20. In these spaces, there are no, a priori, clusters of elements. These spaces are called G5, G10, G15 and G20.

We also generate 100,000 vectors in $\mathbb{R}^{101}$ with mean $\mu = 1$ and variance $\sigma^2 = 0.1$ with 200 clusters (the cluster centers are uniformly distributed in the space). This space is called G101.

Fig. 1(b) shows the estimations obtained with Fractal, Distance Exponent, Fastmap, and Intrinsic Search Difficulty, for these metric spaces. Again, the



(a) Generic pivot index. (b) List of clusters.

Fig. 2. The searching effort does not vary when the ID of the space does not change.

Fractal estimation fails in these spaces, being insensitive to the dimension, and the Distance Exponent grows very slowly. The other two measures grow proportionally to D as they should, although Fractal is less sensitive to the fact that the distribution is not uniform. Instead, the Intrinsic Search Difficulty gives markedly lower values than in the uniform case.

4.2 Real Metric Spaces

We pick four spaces from the Metric Library [11] [4] in order to estimate their IDs with the four ID estimators. The selected spaces are varied:

Dictionary: it is a dictionary of 69,069 English words. In this space, we use a discrete function, the *Edit Distance* or *Levenshtein Distance* [16].

NASA: this is a set of 40,700 images from NASA, represented as feature vectors of 20 real coordinates per vector, under the Euclidean distance. They were generated from images downloaded from the NASA site.

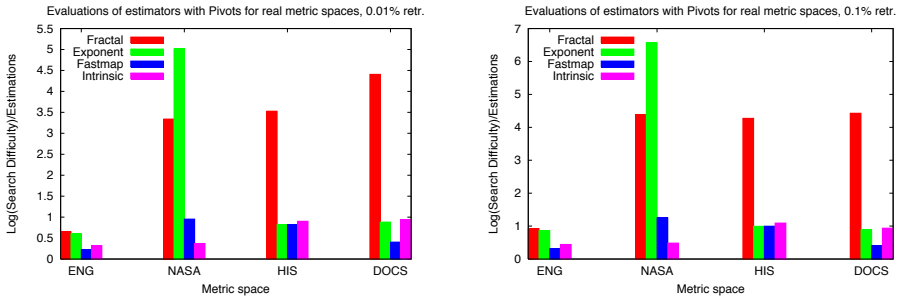
Histograms: this is a dataset of 112,682 histograms of medical images, each one composed by 8-D color histograms of 112 real components. As any quadratic form function can be used as the distance in this case, we also have chosen the Euclidean distance, as it is the simplest alternative.

Documents: this space has 1,265 documents, represented as vectors according to the vectorial model of documents used in the Information Retrieval field. To compare documents we use the *cosine distance*. Each vector has a coordinate for each vocabulary term in the colection, and documents can be seen as points in a vector space of high representational dimension. The documents are files obtained form the TREC-3 collection.

To measure the intrinsic hardness of the searching, we consider the same two indices as before, using range queries:

Dictionary: As the metric is discrete, we use radii 1, 2, 3, and 4, retrieving on average about 0.003%, 0.037%, 0.326%, and 1.757% of the database.

[4] Available at `http://www.sisap.org/library/dbs/` .



(a) Retrieving 0.01%. (b) Retrieving 0.1%.

Fig. 3. Comparison of ID estimators for real metric spaces, using Pivots.

NASA: In this continuous metric we use radii 0.012, 0.285, and 0.53, retrieving on average approximately 0.01%, 0.1%, and 1% of the dataset.

Histograms: This metric is also continuous. To retrieve on average approximately 0.01%, 0.1%, and 1% of the dataset, we use query radii 0.051768, 0.082514, and 0.131163.

Documents: The distance is also continuous. We use query radii 0.14, 0.15, and 0.195, which retrieve on average 0.01%, 0.1%, and 1% of the database.

Figs. 3 and 4 show the correlation between the search cost with the Pivot index and the List of Clusters, respectively, and the estimation reported for each considered ID estimator. For lack of space, we only show the results of the search that retrieve 0.01% and 0.1% of the database.

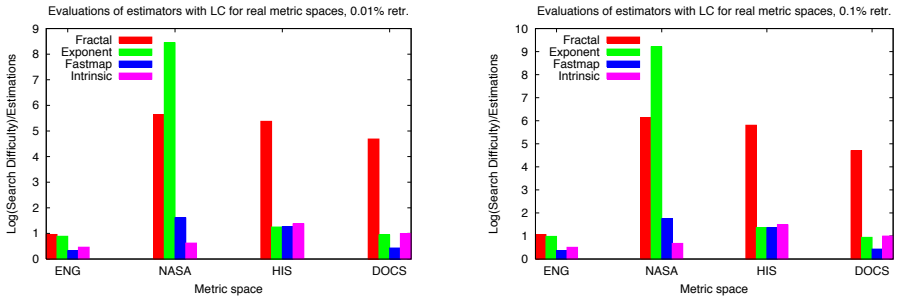
We plot the ratio between the logarithm of the search cost and the estimations of the ID. This measures how close is the logarithm of the predicted ID to the actual search costs: if the search cost is consistently $s = c^d$, where d is the predicted ID and c is a constant, then the plots should always be close to $\log c$. Thus the best methods are those that give roughly the same value regardless of the query radius and index used.

As on the synthetic spaces, Fastmap and the Intrinsic Search Difficulty turn out to be the best predictors for both types of indices. The Distance Exponent performs generally well, except for the NASA dataset.

5 Conclusions

The intrinsic dimension (ID) of metric spaces measures their search difficulty, independently of the type of index used. Computing the ID is useful to determine whether a metric space can be indexed at all (or we must resort to sequential scanning or approximate methods), which kind of index would perform better, and what search performance to expect.

In this paper we have analyzed several ID estimators in a practical perspective. Some were defined for D-dimensional coordinate spaces, and we have adapted them to the more general metric spaces. We compared their predictions with the actual search cost using various synthetic and real-life metric spaces, so as to verify which are better at predicting the search difficulty.



(a) Retrieving 0.01%. (b) Retrieving 0.1%.

Fig. 4. Comparison of ID estimators for real metric spaces, using List of Clusters.

Although our results are preliminary, they suggest that the best performing measures in practice are *Fastmap* [10] and the simple measure proposed by Chávez et al. [6]. Instead, the Distance Exponent [22] and our generalization of Box Counting [17] did not perform so well.

The reason for the failure of Box Counting may be that it needs an extremely large number of objects to correctly estimate D. An estimation [2] is $D < 2 \log_{10} N$, which in our case implies that the method could have worked well up to $D = 10$ only. However, in our experiments the adapted method failed even in this case. It may be that our adaptation to computing it using the List of Clusters [5] is too crude (as other clustering methods may cover the dataset with fewer balls). In any case, this shows that the method is not easy to apply, but we plan to further study this issue with more points and other clustering methods. The reason for the failure of the distance exponent, which does not present issues to be adapted, is also unclear and deserves further research.

We also plan to analyze other estimators. For instance, we can study the correlation dimension [3], the concentration dimension [19], or the classical Principal Component Analysis (PCA) method [15] (which is defined on vector spaces).

References

1. S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conf. on Very Large Databases (VLDB'95)*, pages 574–584, 1995.
2. F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945–2954, 2003.
3. F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE TPAMI*, 24(10):1404–1407, 2002.
4. E. Chávez and J. Marroquín. Proximity queries in metric spaces. In *Proc. 4th South American Workshop on String Processing (WSP'97)*, pages 21–36. Carleton University Press, 1997.
5. E. Chávez and G. Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.
6. E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

7. P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. In *Proc. 23rd VLDB*, pages 426–435, 1997.
8. Paolo Ciaccia, Marco Patella, and Pavel Zezula. A cost model for similarity queries in metric spaces. In *PODS*, pages 59–68, 1998.
9. J. P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617, 1985.
10. C. Faloutsos and K.-I. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proc. 1995 ACM SIGMOD Intl. Conf. on Management of Data*, pages 163–174. ACM Press, 1995.
11. K. Figueroa, G. Navarro, and E. Chávez. Metric spaces library, 2007. Available at `http://www.sisap.org/Metric_Space_Library.html`.
12. K. Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.
13. H. V. Jagadish. A retrieval technique for similar shapes. In *SIGMOD Conference*, pages 208–217. ACM Press, 1991.
14. A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
15. I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd edition, 2002.
16. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
17. B. Mandelbrot. *Fractals: Form, Chance and Dimension*. W. H. Freeman, San Francisco, 1977.
18. E. Ott. *Chaos in dynamical systems*. Cambridge University Press, Cambridge, New York, 1993.
19. V. Pestov. Intrinsic dimension of a dataset: what properties does one expect? In *2007 Intl. Joint Conf. on Neural Networks (IJCNN)*, pages 2959–2964, Aug 2007.
20. V. Pestov. An axiomatic approach to intrinsic dimension of a dataset. *Neural Networks*, 21(23):204 – 213, 2008. Advances in Neural Networks Research: 2007 International Joint Conference on Neural Networks (IJCNN).
21. H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
22. C. Traina Jr., A. J. M. Traina, and C. Faloutsos. Distance exponent: a new concept for selectivity estimation in metric trees. Research Paper 99-110, School of Computer Science, Carnegie Mellon University, 03/1999 1999.
23. P. Yianilos. Excluded middle vantage point forests for nearest neighbor search. In *DIMACS Implementation Challenge, ALENEX'99*, Baltimore, MD, 1999.
24. P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.