Information Systems ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Q2 An empirical evaluation of intrinsic dimension estimators ☆

Q1 Gonzalo Navarro [a], Rodrigo Paredes [b,*], Nora Reyes [c,*], Cristian Bustos [c]

[a] Center of Biotechnology and Bioengineering, Department of Computer Science, University of Chile, Chile
[b] Departamento de Ciencias de la Computación, Universidad de Talca, Chile
[c] Departamento de Informática, Universidad Nacional de San Luis, Argentina

A R T I C L E I N F O

Article history:
Received 16 December 2015
Accepted 10 June 2016

Keywords:
Intrinsic dimension
Complexity of searching
Metric spaces

A B S T R A C T

We study the practical behavior of different algorithms and methods that aim to estimate the intrinsic dimension (IDim) in metric spaces. Some of them 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 IDim estimations with the actual difficulty of searching in metric spaces, we compare two representatives of each of the broadest families of metric indices: those based on pivots and those based on compact partitions. Our conclusions are that the estimators Distance Exponent and Correlation 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* [1–4].

In some applications, the metric spaces are of a particular kind called "vector spaces" of finite *explicit* or *representational* dimension, where the elements consist of D coordinates of real numbers. In this case, we can use some Minkowski metric or any other metric appropriate to the specific case (for instance, the cosine distance) as the dissimilarity measure between two objects. 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 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

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

* Corresponding authors.
E-mail addresses: gnavarro@dcc.uchile.cl (G. Navarro), raparede@utalca.cl (R. Paredes), nreyes@unsl.edu.ar (N. Reyes), cjbustos@unsl.edu.ar (C. Bustos).

http://dx.doi.org/10.1016/j.is.2016.06.004
0306-4379/

from q, formally $(q,r)_d = \{x \in S, d(x,q) \le r\}$; whereas the *nearest neighbor query* $kNN_d(q)$ retrieves the k elements of S that are closest to q, that is, $kNN_d(q)$ is a set such that for all $x \in kNN_d(q)$ and $y \in S \setminus kNN_d(q)$, $d(q,x) \le d(q,y)$, and $|kNN_d(q)| = k$.

A naïve way to answer similarity queries is to compare all the database elements with the query q and return those elements that are close enough to q. 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. Moreover, there are some relatively recent works [5–10] that try to get jointly the goals of reducing the number of distance evaluations and the number of I/O operations performed.

In vector spaces with uniformly distributed data, the *curse of dimensionality* describes the well-known exponential increase of the cost of all existing search algorithms as the dimension grows. Non-uniformly distributed 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* (*IDim*) of a metric space, as a measure of the difficulty of searching it. A reliable measure of IDim has been elusive, despite the existence of several formulae.

Computing the IDim of a metric space is useful, for example, to determine whether it is amenable to indexing at all. If the IDim 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 IDim helps decide which kind of index to use and how to tune it. For example, in lower dimension spaces, a pivot-based method works fine using a small set of pivots; whereas in higher dimensions we need to use a large set of pivots [1], which also implies a large amount of memory for the index. Alternatively, if we do not have enough extra memory for the index, we can switch to the *List of Clusters* [11], which has reasonable performance in high dimension spending little space in the index.

In this work we aim to empirically study the fitness of various IDim 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 four indexing methods that are representatives of the major families of indices: two based on pivots and two based on compact partitions. Our comparison between real and estimated search difficulty yields that *Distance Exponent* [12,13] and *Correlation* [14] are currently the best predictors in practice, however all the estimators behave relatively well.

The rest of this paper is organized as follows. In Section 2, we review some relevant issues of IDim estimators for vector spaces. Next, in Section 3, we survey four methods for estimating IDim in vector spaces and show how to adapt them to the metric case. We also include three new IDim estimators for general metric spaces. The experimental evaluation for the seven methods is presented in Section 4. We finally draw our conclusions and future work directions in Section 5. An early version of this work appeared in [15].

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 [16]. Therefore, data are embedded in $\mathbb{R}^D$, even though this does not imply that its *intrinsic* dimension is D.

There are many definitions of IDim. For instance, the IDim of a given dataset is the minimum number of free variables needed to represent the data without loss of information [17]. In general terms, a dataset $\mathbb{X} \subseteq \mathbb{R}^D$ has IDim $M \le D$, if its elements fall completely within an M-dimensional manifold of $\mathbb{R}^D$ [18]. 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 IDim 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 IDim of $\mathbb{X}$ is $M \le D$, 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. Also, as the amount of available information increases, compressing the data storage becomes even more important. Secondly, as the asymptotic complexity of the algorithms is monotonically increasing with respect to the dataset dimensionality, a dimensionality reduction (to the actual dataset IDim) can produce an important CPU time reduction. For instance, in the case of data classification or pattern recognition, producing reliable classifiers is difficult when the dataset dimensionality is high (*curse of dimensionality* [19]); and according to the theoretical approximation of statistical learning [20], the classifier generalization capability depends on the IDim of the space.

There are two approximations to estimate the IDim of a vector space [16,17], 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. Unlike the local methods that only use the information contained in the neighborhood of each data sample, global methods use whole information of the dataset.

Please cite this article as: G. Navarro, et al., An empirical evaluation of intrinsic dimension estimators, Information Systems (2016), http://dx.doi.org/10.1016/j.is.2016.06.004