The Effects of Dimensionality Curse in High Dimensional kNN Search

Nikolaos Kouiroukidis, Georgios Evangelidis
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
Email: {kouiruki, gevan}@uom.gr

***Abstract*—The dimensionality curse phenomenon states that in high dimensional spaces distances between nearest and farthest points from query points become almost equal. Therefore, nearest neighbor calculations cannot discriminate candidate points. Many indexing methods that try to cope with the dimensionality curse in high dimensional spaces have been proposed, but, usually these methods end up behaving like the sequential scan over the database in terms of accessed pages when queries like k-Nearest Neighbors are examined. In this paper, we experiment with state of the art multi-attribute indexing methods and try to investigate when these methods reach their limits, namely, at what dimensionality a kNN query requires visiting all the data pages. In our experiments we compare the Hybrid Tree, the R*-tree, and, the iDistance Method.**

***Keywords*-high dimensional point indexing; index performance comparison; kNN search**

I. Introduction

In many modern applications, like image and video content indexing and retrieval or time series data mining, similarity search plays an important role. Images, for example, are transformed into high dimensional feature vectors (points) that describe their most interesting features. Then, multi-dimensional indexes can be used to index these vectors and answer similarity queries ([1], [2]). A similarity query, or kNN query, searches for all the points that are "close" to a given query point. kNN queries are a special case of range queries, where, given a query region the index locates all points contained in it. Since those indexes are external memory structures, there have been developed sophisticated and efficient searching algorithms in terms of both time and accessed pages to answer such queries. But, unfortunately, all these indexes are affected by the dimensionality curse problem. According to this problem, above a certain dimensionality, searching for the answer points becomes inefficient because it costs the same as a plain sequential scan of the entire dataset ([3], [4]).

In this paper, we explore the degree that the dimensionality curse problem affects various well known and established indexes. We are interested in determining which indexes are better suited for kNN queries when k varies. Also, we are interested in testing the resistance of the indexes as the dimensionality increases. According to the literature, all indexes suffer above 10 dimensions [5], but unfortunately, the papers that introduce new multi-attribute indexes do not present any experimental results in high dimensions.

The rest of this paper is organized as follows: Section II briefly describes some popular multidimensional indexes, and, Section III discusses the k-Nearest Neighbor problem in high dimensional spaces. Section IV illustrates the dimensionality curse phenomenon, whereas Section V contains experimental results on k-NN searching in high dimensions with various indexes. Finally, Section VI concludes the paper.

II. Review of Popular Multidimensional Indexes

We briefly introduce some of the most promising indexing methods that have been proposed in the literature.

A. Hybrid Tree

This multidimensional indexing structure combines the advantages of space-based and data-based partitioning methods. It is very similar to space partitioning methods (like the kDB-tree [6] and the hB-tree [7]) in that its index nodes use kd-trees, instead of arrays of bounding rectangles, to compactly and efficiently represent space partitioning. It always splits nodes using a single attribute (1-d splitting), but contrary to space partitioning methods, it allows overlapping subspaces (bounding rectangles) the way data partitioning methods do (like the R*-tree) and it avoids cascading node splits (like the KDB-tree).

In addition to the usual metric distance functions like the L_p Norms, one can employ different distance functions when range and kNN queries must be implemented [8].

B. R*-tree

This structure is the most successful variant of the R-tree family of indexing structures [9]. It indexes Minimum Bounding Rectangles (MBRs) and its purpose is to minimize both the coverage and the overlap of the MBRs. It uses a revised node splitting algorithm and a forced reinsertion policy in case of a node split. Thus, it uses a method of incremental tree optimization when reinsertion after a split takes place. It can index both multidimensional point data and data with spatial extent. Due to the concentration of measure phenomenon the discrimination of points is minimized, and finding kNN points in high dimensionality looses its meaning and the R*-tree's performance degrades to that of the sequential scan.

C. *iDistance*

The design of this structure was motivated by the following three observations [10]:

- The (dis)similarity between data points can be derived with reference to a chosen reference or representative point.
- Data points can be ordered based on their distances to a reference point.
- Distance is essentially a single dimensional value.

The iDistance method maps high-dimensional data in single dimensional space, thereby enabling reuse of existing single dimensional indexes, such as the B+tree. This is done using the following procedure: a data point $p(x_0, x_1, \cdots, x_{d-1})$, $0 \leq x_j \leq 1$, $0 \leq j < d$, has an index key, y, based on the distance from the nearest reference point O_i as follows:

$$y = i \times c + dist(p, O_i)$$

where, c is a constant used to stretch the data ranges, and, O_i is a pre-defined set of reference points. Essentially, c serves to partition the single dimension space into regions so that all points in partition P_i are mapped to the range $[i \times c, (i+1) \times c)$.

III. K-Nearest Neighbor Queries

Given a set P of points in a high dimensional space and a query point q, find the k points in P closest to q. This is the k-nearest neighbor search problem. It has significant importance to several areas of computer science, like in searching in multimedia data, pattern recognition, and, data mining in general. These applications involve very large data sets and the dimensionality of the data set is usually high as well. Multidimensional point indexes can be used to speed up searching is such data sets. Thus, it is very important to design algorithms that scale well with the database size as well as with the dimensionality of the data.

The nearest neighbor problem ([11], [12], [5]) is an example of a large class of proximity problems whose definition involves the notion of distance between the input points like the closest pair problem. Many of these problems have been firstly investigated in the field of computational geometry, for example, when the points lie in a space of constant dimension. The nearest neighbor problem of this kind can be solved in $O(logN)$ time per query using $O(N)$ storage. Unfortunately, as the dimensionality grows the complexity of any index converges to $O(N)$, that is, all points must be accessed in order to evaluate a NN-query [5].

The exponential dependence of space or time on the dimensionality, called the "dimensionality curse", has been observed in many application settings. The lack of success in removing the exponential dependence on the dimensionality led many researchers to deduce that no efficient solution exists for these problems when the dimensionality is sufficiently large. But, as many researchers showed, in many cases this exponential dependence on the dimensionality can be reduced to polynomial if we allow the answers to be approximate.

This notion of approximation is best explained as follows: instead of reporting a point p closest to q, the algorithm is allowed to report any point within distance $(1+\varepsilon)$ times the distance from q to p. This is similar to designing efficient approximation algorithms for NP-hard problems.

There are two main algorithms that calculate the nearest neighbor points to a given point in multidimensional spaces using indexing structures like the R*-tree and its variants, the SR-tree [13], the Hybrid tree [8], and many more.

The first one is the Branch and Bound algorithm proposed by Rousopoulos et al [14] that uses the MINDIST and MINMAXDIST metrics to put points for visiting and examination in an order and to determine the nodes that will not be further considered for examination (pruning). Node pruning is accomplished through the use of heuristic methods. A depth first search is implemented so that when the current search path cannot obtain all the nearest neighbors, backtracking is used to examine a neighboring path.

Hjaltason and Samet [15] proposed the second major algorithm for nearest neighbor searching with its incremental form applied to R-trees. Candidate objects are found with an order that is determined from their distance to the query point, thus, they are ranked. This process is called distance browsing. Also, this incremental algorithm differs from the classic kNN in the sense that the number of k nearest neighbors is not known in advance.

IV. Dimensionality curse

The dimensionality curse in applied mathematics refers to the problem caused by the exponential increase in volume associated with adding extra dimensions to a mathematical space ([3], [4], [16]). For example, consider a unit 1-dimensional interval with 100 evenly-spaced sample points, i.e., each point is 0.01 distance units away from its neighbors. An equivalent sampling of a 10-dimensional unit hypercube with a lattice with a spacing of 0.01 between adjacent points would require 10^{20} sample points: thus, in some sense, the 10-dimensional unit hypercube can be said to be a factor of 10^{18} "larger" than the unit 1-dimensional interval. Another way to illustrate the "vastness" of high-dimensional Euclidean space is to compare the proportion of a hypersphere with radius r and dimension d, to that of a hypercube with sides of length $2r$, and equivalent dimension. The volume of such a sphere is:

$$\frac{2r^d \pi^{d/2}}{d\Gamma(d/2)}$$

The volume of the cube would be: $(2r)^d$.

As the dimensionality d increases, the hypersphere's volume becomes insignificant relative to that of the hypercube. This can clearly be seen by computing their ratio as the dimension d goes to infinity:

$$\frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} \rightarrow 0$$

where Γ is the Gamma function: $\Gamma(n) = (n-1)!$, $d \rightarrow \infty$. Thus, in some sense, nearly all of the high-dimensional space is "far away" from the centre, or, to put it another way, the high-dimensional unit space can be said to consist almost entirely of the "corners" of the hypercube, with almost no "middle".

The curse of dimensionality can also be considered from another perspective [17]. In high dimensional space, most of the volume is close to the surface of the data space, as shown in the following example. Let us have the d-dimensional unit hypercube. In order to consider the region close to the surface, let us assume we only consider the locus of points with distance ≤ 0.05 from the surface. This defines a hollow cube with a volume $V = 1-(1-2\times 0.05)^d$. For $d = 3$, $V = 1-0.9^3 = 0.27$, for $d = 10$, $V = 1-0.9^{10} = 0.65$, and, for $d = 15$, $V = 1-0.9^{15} = 0.79$. Actually, in any size and any shape of data space, the above property is satisfied. Because of this property, if data objects are uniformly distributed in space, progressively more of them will be close to the surfaces of the data spaces when the number of dimensions increases. Let the center of the cube be the query object. We can see that the above property causes two effects on the nearest neighbor search. First, if the number of objects is fixed, the average distance of the nearest neighbor will be increased as the dimension increases. Second, the distances of all objects from the center are becoming more and more similar when the number of dimensions increases.

In the multidimensional indexing data literature, dimensionality curse also appears as "the concentration of measure" or "the effect on distance functions" ([18], [19]).

Given a single distribution, the minimum and the maximum occurring distances have been shown to become indiscernible, since the ratio of the difference of the minimum and maximum values and the minimum value converges to 0:

$$\lim_{d\rightarrow\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \rightarrow 0$$

This is often cited as "distance functions losing their usefulness in high dimensionality". However, recent research indicates that the mere number of dimensions is not the problem, since relevant additional dimensions can also increase the contrast. In addition, the resulting ranking remains useful in discerning close and far neighbors. However, irrelevant ("noise") dimensions reduce the contrast, as expected.

The dimensionality curse in high dimensional indexing methods is related to what we call L_p norm. The definition of L_p norm is:

$$\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$$

The L_2 norm, or Euclidean distance metric, is the most commonly used metric, while the L_1 norm is the Manhattan distance metric. Distance functions that rely on these norms in metric spaces have the following three common properties. The distance functions are defined as $d : U \times U \rightarrow R^+$, where U is the universe of the objects.

1) $d(x, y) = 0 \Leftrightarrow x = y$
2) $d(x, y) = d(y, x)$, symmetry
3) $d(x, z) \leq d(x, y) + d(y, z)$, triangle inequality

These three properties are valid for many reasonable similarity functions. Recent research shows that, in high dimensional spaces, the validity of the L_p norm in measuring the similarity between data points is sensitive to the value of p. For example, the Manhattan distance metric (L_1 norm) is consistently more preferable than the Euclidean distance metric (L_2 norm) for high dimensional data mining applications ([16], [20]). Furthermore, a natural extension of the L_p norm to fractional distance metrics is introduced and examined from both the theoretical and empirical perspectives. The fractional distance metric is defined in the same manner with the exception that p belongs to $(0, 1)$.

From the theoretical perspective, fractional distance metrics provide better divergence between the maximum and minimum distances to a given query point than integral distance metrics. This feature makes a proximity query more meaningful and stable. Empirical studies also demonstrate that fractional distance metrics can significantly improve the effectiveness of some standard classification and clustering algorithms such as kNN and k-means on high dimensional datasets. In the meantime, fractional distance measures have been applied to content-based image retrieval and the experiments also show that retrieval performances of these measures consistently outperform the Manhattan and Euclidean distance metrics when used with a wide range of high-dimensional visual features.

In the following section, we demonstrate how some of the most promising multidimensional indexing structures behave in high dimensionality, and, especially, in what dimension given a specific dataset size, kNN searches access as many data pages as a plain sequential scan.

V. Experimental Evaluation

The implementations of the three indexes we tested are available on the web. Since we were interested in testing kNN performance only, we fixed certain parameters in our experiments. We chose to insert 100K uniformly distributed points and we varied the dimensionality from 2 to 24. We also chose a fixed page size of 4KB for all indexes. In some cases we had to modify the original code in order to collect the required statistics. For example, in the case of iDistance,

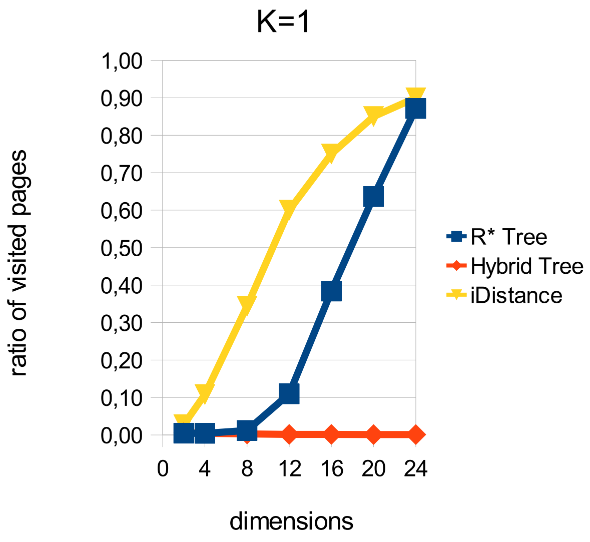

Figure 1. 1NN performance

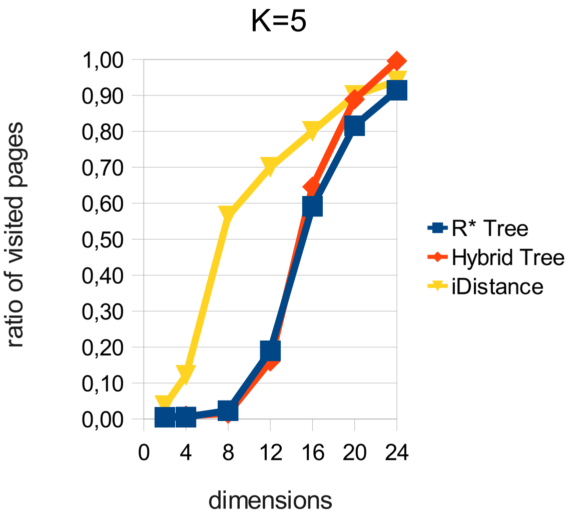

Figure 2. 5NN performance

we chose 64 reference points and we also added counters to compute the size of the B+-tree and the number of retrieved nodes during the queries. The experiments were conducted on a plain Pentium IV computer with 512MB of RAM and a 120GB hard disk running Debian GNU/Linux.

We present the results of our experiments for $k = 1$, $k = 5$ and $k = 10$. In all cases, we plot the ratio of visited data nodes over the total number of data nodes versus the dimensionality when answering the respective kNN queries. For $k = 1$, the Hybrid Tree outperforms all indexes since by design it minimizes node overlapping (Figure 1). For $k = 5$ and $k = 10$ we obtain similar results, with the R*-tree slightly outperforming the Hybrid tree in higher dimensions. The iDistance is always worse that the other two indexes in low to medium dimenions, but it achieves similar performance above 20 dimensions. This is attributed to the fact that the iDistance was designed to perform well in high dimensions. The iDistance is always better than sequential scan albeit by only 5%–10% in high dimensions [10] (Figures 2 and 3).

Finally, we confirm the findings of previous researchers and we observe that the "dimensionality curse" phenomenon renders all indexes unusable in medium dimensionality. The performance of all indexes deteriorates rapidly above eight dimensions. More specifically, all indexes have to visit almost all data pages in order to answer a 24NN query.

VI. Conclusions

In this paper, we demonstrate the inefficiency of the existing popular multidimensional indexing methods when dealing high dimensional spaces. In particular, our experiments showed that the popular k-Nearest Neighbor search suffers greatly above eight dimensions regardless of the chosen indexing method. This is attributed to the fact that in high dimensions there is little discrimination between the nearest and the farthest points from a given query point. A solution to this problem is the design of new distance functions that fit the needs of modern applications ([21], [20]) and the development of new indexes that exploit the new distance functions.

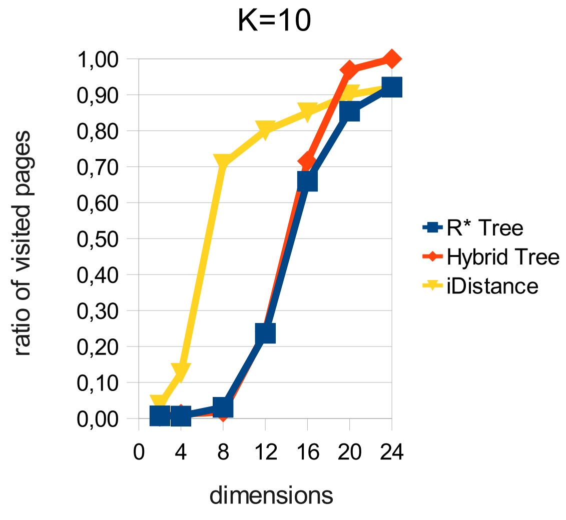

Figure 3. 10NN performance

References

[1] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, pp. 170–231, June 1998. [Online]. Available: http://doi.acm.org/10.1145/280277.280279

[2] R. R. Ciferri and A. C. Salgado, "Performance evaluation of multidimensional access methods," in *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*, ser. GIS '00. New York, NY, USA: ACM, 2000, pp. 183–184. [Online]. Available: http://doi.acm.org/10.1145/355274.355302

[3] R. E. Bellman, *Adaptive control processes - A guided tour*. Princeton, New Jersey, U.S.A.: Princeton University Press, 1961.

[4] D. Francois, V. Wertz, and M. Verleysen, "The concentration of fractional distances," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, pp. 873–886, July 2007. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2007.1037

[5] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, ser. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 194–205. [Online]. Available: http://portal.acm.org/citation.cfm?id=645924.671192

[6] J. T. Robinson, "The K-D-B-tree: a search structure for large multidimensional dynamic indexes," in *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '81. New York, NY, USA: ACM, 1981, pp. 10–18. [Online]. Available: http://doi.acm.org/10.1145/582318.582321

[7] D. B. Lomet and B. Salzberg, "The hB-tree: a multiattribute indexing method with good guaranteed performance," *ACM Trans. Database Syst.*, vol. 15, pp. 625–658, December 1990. [Online]. Available: http://doi.acm.org/10.1145/99935.99949

[8] K. Chakrabarti and S. Mehrotra, "The Hybrid Tree: An index structure for high dimensional feature spaces," in *ICDE*, 1999, pp. 440–447.

[9] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '90. New York, NY, USA: ACM, 1990, pp. 322–331. [Online]. Available: http://doi.acm.org/10.1145/93597.98741

[10] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang, "iDistance: An adaptive B^+-tree based indexing method for nearest neighbor search," *ACM Trans. Database Syst.*, vol. 30, no. 2, pp. 364–397, 2005.

[11] U. Shaft and R. Ramakrishnan, "Theory of nearest neighbors indexability," *ACM Trans. Database Syst.*, vol. 31, pp. 814–838, September 2006. [Online]. Available: http://doi.acm.org/10.1145/1166074.1166077

[12] A. Hinneburg, C. C. Aggarwal, and D. A. Keim, "What is the nearest neighbor in high dimensional spaces?" in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 506–515. [Online]. Available: http://portal.acm.org/citation.cfm?id=645926.671675

[13] N. Katayama and S. Satoh, "The SR-tree: an index structure for high-dimensional nearest neighbor queries," in *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '97. New York, NY, USA: ACM, 1997, pp. 369–380. [Online]. Available: http://doi.acm.org/10.1145/253260.253347

[14] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '95. New York, NY, USA: ACM, 1995, pp. 71–79. [Online]. Available: http://doi.acm.org/10.1145/223784.223794

[15] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, pp. 265–318, June 1999. [Online]. Available: http://doi.acm.org/10.1145/320248.320255

[16] D. François, V. Wertz, and M. Verleysen, "Non-euclidean metrics for similarity search in noisy datasets," in *ESANN*, 2005, pp. 339–344.

[17] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," in *Proceedings of the 8th International Conference on Database Theory*, ser. ICDT '01. London, UK: Springer-Verlag, 2001, pp. 420–434. [Online]. Available: http://portal.acm.org/citation.cfm?id=645504.656414

[18] R. J. Durrant and A. Kabn, "When is 'nearest neighbour' meaningful: A converse theorem and implications," *Journal of Complexity*, vol. 25, no. 4, pp. 385 – 397, 2009.

[19] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbor' meaningful?" in *Proceedings of the 7th International Conference on Database Theory*, ser. ICDT '99. London, UK: Springer-Verlag, 1999, pp. 217–235. [Online]. Available: http://portal.acm.org/citation.cfm?id=645503.656271

[20] C.-M. Hsu and M.-S. Chen, "On the design and applicability of distance functions in high-dimensional data space," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 523–536, 2009.

[21] C. C. Aggarwal, "Re-designing distance functions and distance-based applications for high dimensional data," *SIGMOD Rec.*, vol. 30, pp. 13–18, March 2001. [Online]. Available: http://doi.acm.org/10.1145/373626.373638