Indexing Issues in Supporting Similarity Searching[⋆]

Hanan Samet

Computer Science Department, Center for Automation Research, Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742
hjs@cs.umd.edu, www.cs.umd.edu/~hjs

Abstract. Indexing issues that arise in the support of similarity searching are presented. This includes a discussion of the curse of dimensionality, as well as multidimensional indexing, distance-based indexing, dimension reduction, and embedding methods.

1 Introduction

The representation of multidimensional points and objects, and the development of appropriate indexing methods that enable them to be retrieved efficiently is a well-studied subject (e.g., [1,2]). Most of these methods were designed for use in application domains where the data usually has a spatial component which has a relatively low dimension. Examples of such application domains include geographic information systems (GIS), spatial databases, solid modeling, computer vision, computational geometry, and robotics. However, there are many application domains where the data is of considerably higher dimensionality, and is not necessarily spatial. This is especially true in multimedia databases where the data is a set of objects and the high dimensionality is a direct result of trying to describe the objects via a collection of features (also known as a *feature vector*). In the case of images, examples of features include color, color moments, textures, shape descriptions, etc. expressed using scalar values. The goal in these applications is often expressed more generally as one of the following:

1. Find objects whose feature values fall within a given range or where the distance from some query object falls into a certain range (range queries).
2. Find objects whose features have values similar to those of a given query object or set of query objects (nearest neighbor queries).

These queries are collectively referred to as *similarity searching*, and the issues involved in supporting them is the subject of this paper, which is organized as follows. Section 2 mentions the use of Voronoi diagrams, while Section 3 describes the curse of dimensionality. Sections 4 and 5 discusses multidimensional indexing and distance-based indexing, respectively, while Section 6 briefly touches on dimension reduction and embedding methods. Concluding remarks are drawn in Section 7.

[⋆] This work was supported in part by the National Science Foundation under grants EIA-99-00268, IIS-00-86162, and EIA-00-91474, and Microsoft Research.

2 Voronoi Diagrams

An apparently straightforward solution to finding the nearest neighbor is to compute a Voronoi diagram (e.g., [3]) for the data points (i.e., a partition of the space into regions where all points in the region are closer to the region's associated data point than to any other data point), and then locate the Voronoi region corresponding to the query point. The problem with this solution is that the combinatorial complexity of the Voronoi diagram in high dimensions is prohibitive — that is, it grows exponentially with its dimension k so that for N points, the time to build and the space requirements can grow as rapidly as $\Theta(N^{k/2})$ [3]. This renders its applicability moot.

3 Curse of Dimensionality

The above is typical of the problems that we must face when dealing with high-dimensional data. Generally speaking, multidimensional queries become increasingly more difficult as the dimensionality increases. The problem is characterized as the *curse of dimensionality*. This term was coined by Bellman [4] to indicate that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with the number of variables (i.e., dimensions) that comprise it. For similarity searching (i.e., finding nearest neighbors), this means that the number of objects (i.e., points) in the data set that need to be examined in deriving the estimate grows exponentially with the underlying dimension.

The curse of dimensionality has a direct bearing on similarity searching in high dimensions as it raises the issue of whether or not nearest neighbor searching is even meaningful in such a domain. In particular, letting d denote a distance function which need not necessarily be a metric, Beyer et al. [5] point out that nearest neighbor searching is not meaningful when the ratio of the variance of the distance between two random points p and q, drawn from the data and query distributions, and the expected distance between them converges to zero as the dimension k goes to infinity — that is,

$$\lim_{k\to\infty} \frac{\mathrm{Variance}[d(p,q)]}{\mathrm{Expected}[d(p,q)]} = 0.$$

In other words, the distance to the nearest neighbor and the distance to the farthest neighbor tend to converge as the dimension increases. Formally, Beyer et al. demonstrate that when the data and query distributions satisfy this ratio, the probability that the farthest neighbor distance is smaller than $1+\epsilon$ of the nearest neighbor distance is 1 in the limit as the dimension k goes to infinity and ϵ is a positive value. For example, they show that this ratio holds whenever the coordinate values of the data and the query point are independent and identically distributed as is the case when they are both drawn from a uniform distribution.

Assuming that d is a distance metric and hence that the triangle inequality holds, an alternative way of looking at the curse of dimensionality is to observe that when dealing with high-dimensional data, the probability density function (analogous to a histogram) of the distances of the various elements is more concentrated and has a larger mean value. This means that similarity searching algorithms will have to perform more work.

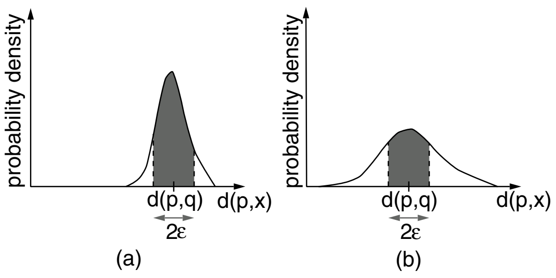

Fig. 1. A probability density function (analogous to a histogram) of the distances d(p,x) with the shaded area corresponding to $|\mathsf{d}(\mathsf{q},\mathsf{p}) - \mathsf{d}(\mathsf{p},\mathsf{x})| \le \epsilon$. (a) indicates a density function where the distance values have a small variation, while (b) indicates a more uniform distribution of distance values thereby resulting in a more effective use of the triangle inequality to prune objects from consideration as satisfying the range search query.

In the worst case, we have the situation where $d(x,x) = 0$ and $d(x,y) = 1$ for all $y \neq x$, which means that a similarity query must compare the query object with every object of the set. One way to see why more concentrated probability densities lead to more complex similarity searching is to observe that this means that the triangle inequality cannot be used so often to eliminate objects from consideration. In particular, the triangle inequality implies that every element x such that $|d(q,p) - d(p,x)| > \epsilon$ cannot be at a distance of ϵ or less from q (i.e., from $d(q,p) \le d(p,x) + d(q,x)$). Thus if we examine the probability density function of $d(p,x)$ (i.e., on the horizontal axis), we find that when ϵ is small while the probability density function is large at $d(p,q)$, then the probability of eliminating an element from consideration via the use of the triangle inequality is the remaining area under the curve, which is quite small (see Figure 1a in contrast to Figure 1b where the density function of the distances is more uniform).

These observations mean that nearest neighbor searching may be quite inefficient as it is difficult to differentiate between the nearest neighbor and the other elements. Moreover, seemingly appropriate indexing methods, such as k-d trees [6] and R-trees [7] which are designed to make it easier to avoid examining irrelevant elements, may not be of use in this case. In fact, the experiments of Beyer et al. [5] show that the curse of dimensionality becomes noticeable for dimensions as low as 10 to 15 for the uniform distribution. The only saving grace is that real world high-dimensional data (say of dimension k) is not likely to be uniformly distributed as their volume is much smaller than $O(c^k)$ for some small constant $c > 2$. Thus we can go on with our discussion despite the apparent pall of the curse of dimensionality which tends to cast a shadow on any arguments or analyses that are based on uniformly-distributed data or queries.

4 Multidimensional Indexing

Assuming that the curse of dimensionality does not come into play, query responses are facilitated by sorting the objects on the basis of some of their feature values and building appropriate indexes. The high-dimensional feature space is indexed using some multidimensional data structure (termed *multidimensional indexing*) with appropriate modifications to fit the high-dimensional problem environment. Similarity search which finds objects similar to a target object can be performed with a range search or a nearest neighbor search in the multidimensional data structure. However, unlike applications in spatial databases where the distance function between two objects is usually Euclidean, this is not necessarily the case in the high-dimensional feature space where the distance function may even vary from query to query on the same feature (e.g., [8]).

Searching in high-dimensional spaces is time-consuming. Performing range queries in high dimensions is considerably easier, from the standpoint of computational complexity, than performing similarity queries as range queries do not involve the computation of distance. In particular, searches through an indexed space usually involve relatively simple comparison tests. However, if we have to examine all of the index nodes, then the process is again time-consuming. In contrast, computing similarity in terms of nearest neighbor search makes use of distance and the process of computing the distance can be computationally complex. For example, computing the Euclidean distance between two points in a high-dimensional space, say d, requires d multiplication operations and $d-1$ addition operations, as well as a square root operation (which can be omitted). Note also that computing similarity requires the definition of what it means for two objects to be similar, which is not always so obvious.

5 Distance-Based Indexing

Often, the only information that we have available is a distance function that indicates the degree of similarity (or dis-similarity) between all pairs of the N given objects. Usually the distance function d is required to obey the triangle inequality, be non-negative, and be symmetric, in which case it is known as a *metric* and also referred to as a *distance metric*. Sometimes, the degree of similarity is expressed by use of a similarity matrix which contains interobject distance values, for all possible pairs of the N objects

Given a distance function, we usually index the data (i.e., objects) with respect to their distance from a few selected objects. We use the term *distance-based indexing* to describe such methods (e.g., [9]). A number of such methods have been proposed over the past few decades, some of the earliest being due to Burkhard and Keller [10]. These methods generally assume that we are given a finite set S of N objects and a distance metric d indicating the distance values between them (collectively termed a *finite metric space*) Typical of distance-based indexing structures are *metric trees* [11,12], which are binary trees that result in recursively partitioning a data set into two subsets at each node. Uhlmann [12] identified two basic partitioning schemes, *ball partitioning* and *generalized hyperplane partitioning*.

In ball partitioning, the data set is partitioned based on distances from one distinguished object, sometimes called a *vantage point* [13], into the subset that is inside and the subset that is outside a ball around the object (e.g., see Figure 2a). In generalized hyperplane partitioning, two distinguished objects p_1 and p_2 are chosen and the data set is partitioned based on which of the two distinguished objects is the closest — that is, all the objects in subset A are closer to p_1 than to p_2, while the objects in subset B are closer to p_2 (e.g., see Figure 2b). The asymmetry of ball partitioning (which is

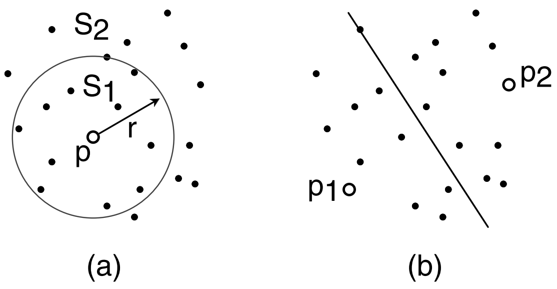

Fig. 2. Possible top-level partitionings of a set of objects (depicted as two-dimensional points) in a metric tree using (a) ball partitioning and (b) generalized hyperplane partitioning.

evident from Figure 2a) is a potential drawback of this method as the outer shell tends to be very narrow for metric spaces typically used in similarity search (e.g., see [14]). In contrast, generalized hyperplane partitioning is more symmetric, in that both partitions form a "ball" around an object (see Figure 2b). The vp-tree [13] is an example of a ball partitioning tree while the gh-tree [12] is an example of a generalized hyperplane partitioning tree.

An alternative way of distinguishing between some of the different distance-based indexing methods is on the basis of whether they are pivot-based or clustering-based (e.g., [15]). Pivot-based methods choose a subset of the objects in the data set to serve as distinguished objects, termed *pivot objects* (or more generally *pivots*), and classify the remaining objects in terms of their distances from the pivot objects. Pivot-based similarity searching algorithms make use of the known distances from the objects to different pivot objects to reduce the number of distance computations involving the query object that will be needed to respond to the query. The pivot objects, assuming without loss of generality that there are k of them, can often be viewed as coordinates in a k-dimensional space and the result of the distance computation for object x is equivalent to a mapping of x to a point $(x_0, x_1, \ldots, x_{k-1})$ where coordinate value x_i is the distance $d(x, p_i)$ of x from pivot p_i. The result is similar to embedding methods which are discussed further below. Ball partitioning methods are examples of pivot-based methods. In addition, methods that make use of distance matrices which contain precomputed distances between some or all of the objects in the data set (e.g., [16]) are also examples of pivot-based methods. Note that distance matrix methods differ from ball partitioning methods in that they do not form a hierarchical partitioning of the data set.

Clustering-based methods partition the underlying data set into spatial-like zones called *clusters* that are based on proximity to a distinguished object known as the *cluster center*. In particular, once a set of cluster centers has been chosen, the objects that are associated with each cluster center c are those that are closer to c than to any other cluster center. Although the cluster centers play a similar role as the pivot objects, the principal difference is that an object o is associated with a particular pivot p on the basis of the distance from o to p and not because p is the closest pivot to o, which would be the case if p was a cluster center. Generalized-hyperplane partitioning methods are examples of clustering-based methods. The *sa-tree* [17,18], inspired by the Voronoi diagram, is another example of a clustering-based method. It records a portion of the Delaunay graph of the data set, which is a graph whose vertices are the Voronoi cells, with edges between adjacent cells. Although many of the clustering-based methods are hierarchical, this need not necessarily be the case.

It is interesting to observe that both pivot-based and clustering-based methods achieve a partitioning of the underlying data set into spatial-like zones. However, the difference is that the boundaries of the zones are more well-defined in the case of pivot-based methods as they can be expressed explicitly using a small number of objects and a known distance value. In contrast, in the case of clustering-based methods, the boundaries of the zones are usually expressed implicitly in terms of the cluster centers, instead of explicitly, which may require quite a bit of computation to determine. In fact, very often, the boundaries cannot be expressed explicitly as, for example, in the case of an arbitrary metric space (in contrast to a Euclidean space) where we do not have a direct representation of the 'generalized hyperplane' that separates the two partitions.

The advantage of distance-based indexing methods is that distance computations are used to build the index, but once the index has been built, similarity queries can often be performed with a significantly lower number of distance computations than a sequential scan of the entire dataset. Of course, in situations where we may want to apply several different distance metrics, then the drawback of the distance-based indexing techniques is that they require that the index be rebuilt for each different distance metric, which may be nontrivial. This is not the case for the multidimensional indexing methods which have the advantage of supporting arbitrary distance metrics (however, this comparison is not entirely fair, since the assumption, when using distance-based indexing, is that often we do not have any feature values as for example in DNA sequences).

6 Dimension Reduction and Embedding Methods

There are many problems with indexing high-dimensional data. In particular, it can be shown that, assuming uniformly-distributed high-dimensional data, most of the data lies at or near the boundary of the data space [19] (e.g., for 20 dimensions, 98.85% of data lies in the outermost 10% of the hypercube of the data space). Therefore, only rarely is the data volume so high that every dimension is split even once when using an index such as a k-d tree. Thus a typical query region often overlaps all of the leaf node regions of the index which means that the cost of performing queries using the index is often higher than a sequential scan of the entire data (e.g., [5,20]). In fact, this is another manifestation of the curse of dimensionality.

Nevertheless, the "inherent dimensionality" of a data set is often much lower than the dimensionality of the underlying space. For example, the values of some of the features may be correlated in some way. Thus there has been a considerable amount of interest in techniques to reduce the dimensionality of the data using methods such as Singular Value Decomposition (SVD) [21], Karhunen-Loève Transform (KLT) [22], and Principal Component Analysis (PCA) [22]. Another motivation for the development of many dimension-reduction techniques has been a desire to make use of disk-based spatial indexes which are based on object hierarchies such as members of the R-tree family [7, 23,24]. The performance of these methods decreases with an increase in dimensionality due to the decrease in the fanout of a node of a given capacity since usually the amount of storage needed for the bounding boxes is directly proportional to the dimensionality of the data thereby resulting in longer search paths.

In situations where no features are defined for the objects but only a distance function, there exists an alternative to using distance-based indexes. In particular, methods have been devised for deriving "features" purely based on the inter-object distances [25,26, 27,28]. Thus, given N objects, the goal is to choose a value of k and find a set of N corresponding points in a k-dimensional space so that the distance between the N corresponding points is as close as possible to that given by the distance function for the N objects. In particular, if the methods are contractive (i.e., the distance in the embedding space is always less than the distance in the original space) [29], then we can now index the points using multidimensional data structures while guaranteeing 100% recall (i.e., that we will not miss any objects). These methods are known as *embedding methods* and can also be applied to objects represented by feature vectors as alternatives to the traditional dimension-reduction methods. Not all embedding methods are contractive for all distance metrics.

7 Concluding Remarks

Providing indexing support for similarity searching is an important area where much work remains to be done. Some of the more promising research directions lie in developing techniques to identify the important features in the applications so that the dimension of the problem domain can be reduced thereby enabling us to properly utilize the vast array of existing indexing and nearest neighbor techniques.

References

1. Samet, H.: Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley, Reading, MA (1990)
2. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading, MA (1990)
3. Aurenhammer, F.: Voronoi diagrams — a survey of a fundamental geometric data structure. ACM Computing Surveys **23** (1991) 345–405
4. Bellman, R.E.: Adaptive Control Processes. Princeton University Press, Princeton, NJ (1961)
5. Beyer, K.S., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In Beeri, C., Buneman, P., eds.: Proceedings of the 7th International Conference on Database Theory (ICDT'99), Berlin, Germany (1999) 217–235
6. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18** (1975) 509–517
7. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD Conference, Boston (1984) 47–57
8. Rui, Y., amd S. Mehrotra, T.S.H.: Content-based image retrieval with relevance feedback in MARS. In: Proceedings of the 1997 IEEE International Conference on Image Processing, Santa Barbara, CA (1997) 815–818
9. Hjaltason, G.R., Samet, H.: Index-driven similarity search in metric spaces. ACM Transactions on Database Systems **28** (2003) 517–580

10. Burkhard, W.A., Keller, R.: Some approaches to best-match file searching. Communications of the ACM **16** (1973) 230–236
11. Uhlmann, J.K.: Metric trees. Applied Mathematics Letters **4** (1991) 61–62
12. Uhlmann, J.K.: Satisfying general proximity/similarity queries with metric trees. Information Processing Letters **40** (1991) 175–179
13. Yianilos, P.N.: Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, TX (1993) 311–321
14. Brin, S.: Near neighbor search in large metric spaces. In Dayal, U., Gray, P.M.D., Nishio, S., eds.: Proceedings of the 21st International Conference on Very Large Data Bases (VLDB), Zurich, Switzerland (1995) 574–584
15. Chávez, E., Navarro, G.: An effective clustering algorithm to index high dimensional spaces. In: Proceedings String Processing and Information Retrieval (SPIRE 2000), A Coruña, Spain (2000) 75–86
16. Vidal Ruiz, E.: An algorithm for finding nearest neighbours in (approximately) constant average time. Pattern Recognition Letters **4** (1986) 145–157
17. Navarro, G.: Searching in metric spaces by spatial approximation. VLDB Journal **11** (2002) 28–46
18. Hjaltason, G.R., Samet, H.: Improved search heuristics for the SA-tree. Pattern Recognition Letters **24** (2003) 2785–2795
19. Berchtold, S., Böhm, C., Kriegel, H.P.: Improving the query performance of high-dimensional index structures by bulk-load operations. In: Advances in Database Technology — EDBT'98, Proceedings of the 6th International Conference on Extending Database Technology, Valencia, Spain (1998) 216–230
20. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Gupta, A., Shmueli, O., Widom, J., eds.: Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), New York (1998) 194–205
21. Golub, G.H., van Loan, C.F.: Matrix Computations. Third edn. Johns Hopkins University Press, Baltimore, MD (1996)
22. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Second edn. Academic Press, Boston (1990)
23. Katayama, N., Satoh, S.: The SR-tree: an index structure for high-dimensional nearest neighbor queries. In Peckham, J., ed.: Proceedings of the ACM SIGMOD Conference, Tucson, AZ (1997) 369–380
24. White, D.A., Jain, R.: Similarity indexing with the SS-tree. In Su, S.Y.W., ed.: Proceedings of the 12th IEEE International Conference on Data Engineering, New Orleans (1996) 516–523
25. Faloutsos, C., Lin, K.I.: FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: Proceedings of the ACM SIGMOD Conference, San Jose, CA (1995) 163–174
26. Hristescu, G., Farach-Colton, M.: Cluster-preserving embedding of proteins. Technical report, Rutgers University, Piscataway, NJ (1999)
27. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. Combinatorica **15** (1995) 215–245
28. Wang, J.T.L., Wang, X., Lin, K.I., Shasha, D., Shapiro, B.A., Zhang, K.: Evaluating a class of distance-mapping algorithms for data mining and clustering. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA (1999) 307–311
29. Hjaltason, G.R., Samet, H.: Properties of embedding methods for similarity searching in metric spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003) 530–549 Also University of Maryland Computer Science TR-4102.