Fundamentals of the Problem

Edgar Chávez* and Gonzalo Navarro†

* Universidad Michoacana / CICESE, Mexico. *elchavez@umich.mx*.

† Department of Computer Science, University of Chile. *gnavarro@dcc.uchile.cl*.

Spatial distance has been for decades a natural concept in applications such as computational geometry, geomatics, meshing, and others. There are several other applications, however, where a more general notion of *proximity* is necessary. To reuse the large body of knowledge built around spatial distances, researchers have tried to map non-spatial proximity measures to those. This approach is not always successful, and fails in some emblematic cases. Instead, a flexible model, called "metric spaces", captures in a natural way more general proximity concepts. We will briefly present its fundamentals; exhaustive books and surveys exist [2, 3, 1].

A *metric space* is a pair $(\mathbb{X}, d)$ where $\mathbb{X}$ is a *universe* of objects and $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}^+$ is a nonnegative *distance* function defined among the objects. This distance satisfies the properties that make it a *metric*: (1) strict positiveness: $d(x, y) > 0 \Leftrightarrow x \neq y$, (2) symmetry: $d(x, y) = d(y, x)$, (3) triangle inequality: $d(x, y)+d(y, z) \geq d(x, z)$. In a metric space coordinates are not necessarily present, thus most of the algorithms designed for coordinate spaces do not apply directly.

Some examples of metric spaces are sequences with variants of the edit distance (used, say, in computational biology, it measures the minimum number of single-character insertions, deletions and substitutions that must be carried to convert one sequence into the other); documents with cosine similarity (used in Information Retrieval, the associated distance measures the angle between unitary vectors representing the documents in a space of terms); and point clouds with Hausdorff or Earthmover's distances (used in clustering, pattern recognition, etc).

In such applications, one has a finite *dataset* $\mathbb{U} \subset \mathbb{X}$ of size n, which can be *indexed* (i.e., data structures can be built on it) so as to answer various proximity queries. The most basic ones are (a) *range query* (q, r), for $q \in \mathbb{X}$ and $r \in \mathbb{R}^+$, returns $\{u \in \mathbb{U},\ d(q, u) \leq r\}$, the dataset objects at distance within r of q; and (b) *k-nearest neighbor query* $nn(q, k)$, for $k \in \mathbb{N}^+$, a set of k elements of $\mathbb{U}$ nearest to q. Several other queries, such as proximity joins, are of interest in applications.

The simplest cost model assumes that computing d is so expensive that other costs can be disregarded. Under this setup, the problem can be stated as follows: *Given $\mathbb{U}$, build an index on it so that queries can be later answered with as few distance computations as possible.*

The key to avoid comparing q with all of the dataset is the triangle inequality. Assume $u, v \in \mathbb{U}$, so that the index has precomputed $d(u, v)$. Now, if for solving range query (q, r) we compute $d(q, u)$, we might avoid computing also $d(q, v)$ if $|d(q, u) - d(u, v)| > r$, since by the triangle inequality this implies $d(q, v) > r$. Hence inter-dataset distance information, plus the triangle inequality, allows us prove *lower bounds* on the distances between new objects q and dataset objects.

The objects of the universe are seen as black boxes (in particular without any coordinate information), so that all we can know from them are their distances to other objects. Under such

model, the most information an index can store on the dataset is the complete symmetric matrix of $n(n-1)/2$ distances. Using that much information, it has been shown theoretically and practically that $nn(q,k)$ queries can be solved with a number of distance evaluations that is independent of n.

Such a result hides, however, a dependence on what is called the *intrinsic dimensionality* of the metric space (or of a finite sample dataset). In coordinate spaces this is defined as the lowest dimensionality in which the dataset can be embedded without distorting the distances more than a threshold. In metric spaces the concept is much sloppier and no agreement exists on how to measure it (two good examples of definitions that try to carry the intuition of coordinate spaces onto metric spaces are the exponent dimension and the doubling dimension). A rule of thumb is to measure how concentrated is the histogram of distances: A sharp histogram is related to a higher intrinsic dimensionality of the dataset. The intrinsic dimensionality is known to act as a lower bound on the performance of any index for a given dataset, yet again a more precise quantification of this phenomenon, called the *curse of dimensionality*, has been elusive.

A related problem is the difficulty of carrying out any kind of formal performance analysis on metric spaces. Usually no meaningful worst-case analyses are possible, and average-case ones are difficult because they depend on the distribution of the data and queries in a nontrivial way, and researchers have not agreed on an appropriate model (simple enough to be tractable, realistic enough to have some predictive power). We believe other paths might be worthy to explore, such as competitive analysis, where a given index is compared to an "optimal" one that carries out the minimum number of distance evaluations that fully determines the correct output.

Needless to say, storing the $O(n^2)$ distances is impractical for all but very small datasets. Many alternative indexing schemes for metric spaces have been proposed along the years. All of them can be regarded as different ways to use a given amount of memory in the best possible way, by storing a subset of the inter-dataset distances, or ranges thereof. In addition, several schemes include mechanisms to reduce the extra CPU time, which can be important in practice.

For simplicity, let us focus on range search algorithms. Nearest-neighbor algorithms are usually more important in applications, yet from each range search algorithm, a *range-optimal* nearest-neighbor one can be systematically derived. Range optimality means that the cost of the query $nn(q,k)$ is exactly that of a range query (q,r) where r is the distance to the kth object returned. There are also indexing schemes specifically designed to solve nearest-neighbor queries.

An important family of indexing schemes can be directly seen as a reduced-memory version of the quadratic-space method. We choose m *pivots* $\mathbb{P} = \{p_1, p_2, \ldots, p_m\} \subset \mathbb{U}$ and precompute all the distances $d(p,u)$ for $p \in \mathbb{P}$ and $u \in \mathbb{U}$. This requires $O(mn)$ space. Now, given a range query (q,r), all the distances $d(q,p)$ for $p \in \mathbb{P}$ are computed, and the triangle inequality is used as explained to discard as many elements of $\mathbb{U} - \mathbb{P}$ as possible. Non-discarded elements, that is those $u \in \mathbb{U}$ such that $\max_{1 \le i \le m} |d(p_i,q) - d(p_i,u)| \le r$, must be directly compared with q.

This method can be seen as a contractive mapping from the metric space $(\mathbb{X}, d)$ onto the coordinate space $(\mathbb{R}^m, L_\infty)$, where L_∞ is Minkowski's maximum distance: Each $u \in \mathbb{U}$ is mapped to vector $\phi(u) = (d(u,p_1), d(u,p_2), \ldots, d(u,p_m))$. At query time q is also mapped onto $(\mathbb{R}^m, L_\infty)$ (at the cost of m computations of $d()$). Now if $L_\infty(\phi(u), \phi(q)) > r$, then it also holds $d(q,u) > r$ in the original space, and thus $d(q,u)$ needs not be directly computed. We note that the search for the candidate elements u such that $L_\infty(\phi(u), \phi(q)) \le r$ is a geometric range search problem,

on which we could wish to minimize CPU time (this time is disregarded under the simple model where only computations of $d()$ count).

The pivot-based method works well on sufficiently "easy" metric spaces and queries, where a relatively small m lets one discard most of $\mathbb{U}$. Note this m is both the cost in distance computations to map q to $\mathbb{R}^m$, and the dimension of the coordinate space where the search for candidates must be carried out. When the metric space has high intrinsic dimension, however, even using a large m the achieved pruning is very modest. This is one incarnation of the curse of dimensionality.

A related issue is the quality of the pivots. Since the mapping is contractive, $L_\infty \circ \phi \leq d$ is guaranteed. A lower bound, such as $L_\infty \circ \phi \geq \alpha \cdot d$, $0 < \alpha < 1$, would be needed to offer performance guarantees. Unfortunately such a lower bound cannot be offered in general, thus pivot selection trying to offer a good-quality mapping is done using heuristics and validated experimentally.

Many indexing schemes are built on variants of the pivot idea. From a general viewpoint, they can be seen as a quest for, given a limited amount of memory for the index, how to best use that space. One can, for example, store only some distances towards each pivot (the most "interesting" ones), store only the range where the distance falls (thus using fewer bits and storing more distances in exchange), and so on. In particular, it is possible to build a tree of pivots, so that each pivot knows the distances only to its subtree, and its children are organized into ranges of distances towards it. This allows for linear-space indexes that in addition allow one to use sublinear extra CPU time, as whole subtrees can be pruned during the search using the triangle inequality.

Another family of indexes uses the pivots in a different way. The space is partitioned into m *zones*, each zone being that of the objects closer to a pivot than to any other pivot. Such a Voronoi-like partitioning of the space can be used with the triangle inequality to discard whole zones at query time: Let (q, r) be a range query and let p be the pivot closest to q (it may cost m distance computations to determine p). Then, if $d(p, q) + r < d(p', q) - r$ for another pivot p', there cannot be points relevant to the query within the zone of p'. Tree-structured indexes can be built by dividing recursively the zones into sub-zones.

For some applications it is enough to have either an approximate answer, with proximity guarantees to the true answer, or the correct answer with some probability. This relaxation has been shown to offer excellent performance even on high-dimensional metric spaces, which are intractable for exact proximity searching. A general (but not the only) approach is to *prioritize* the traversal of the database in some "promising" order, so that after traversing a small subset of $\mathbb{U}$ one has a large chance of having reached the best answers. Any nearest-neighbor algorithm, range-optimal ones in particular, can be pruned at some point of its execution and turn it into an algorithm that gives approximate answers. A particularly successful idea is to hint promising objects by observing the proximity order in which dataset elements "see" a set of pivots. Each object is represented by a permutation and proximity is predicted by measuring the similarity between objects' and query's permutations. The dataset is then traversed by permutation proximity order.

Open challenges. The field of metric space searching has made important progress since its origins, both in theoretical and practical aspects. Yet, it is a far from solved area. A recently created conference, *Similarity Search and Applications (SISAP)*, is devoted to research on this topic (see www.sisap.org). We finish by listing which we consider the main open challenges, each of which is developed in the subsequent entries in this issue.

Intrinsic dimensionality: Finding a suitable definition of intrinsic dimensionality, consistent with the case of coordinate spaces, and capturing the experimental performance of indexes, would allow us explain why, and measure by how much, some metric spaces are intrinsically harder to search than others, and to use appropriate techniques depending on their difficulty. This entry shows that, despite the phenomenon is well-known among practitioners, there is no single theoretical definition that fits all situations, and very few cases where a lower bound on the performance of any index as a function of some definition of intrinsic dimension can be proved. It is shown that pivot-based indexes can be viewed as instances of a more general approach where the curse of dimensionality arises in the form of a phenomenon called "concentration of measure", which states that any contractive mapping (satisfying $|f(x) - f(y)| \leq d(x, y)$) tends to have almost all of its values concentrated around the median, and this leads to ineffective indexes. Finally, it shows that if a space is found to have low dimension under some particular definition, this can be used to design an effective index exploiting that feature. This is illustrated with the "Assouad dimension", related to the number of balls needed to cover the objects of a space. When this dimension is low, a metric tree effectively indexes the space. They conjecture that many spaces today considered as hard could become tractable under a convenient definition of intrinsic dimension.

Nearest-neighbor algorithms: Despite range-optimal algorithms can be systematically derived from range search ones, which are usually simpler to design, it is possible to design specific algorithms for nearest neighbor search. In many applications this is the most important type of search. This entry starts by describing four main algorithmic approaches to nearest-neighbor search: branch and bound (where sets of candidate objects are defined according to some hierarchy at indexing time, and at query time they are discarded or refined into subsets depending on lower/upper bounds obtained from successive distance computations), walks (where one starts at an object and tries to approach the target object by moving to nearby objects), mappings (where objects are mapped to a target space where the problem can be solved more easily, and from the solution in the target space a small candidate set in the original space can be derived), and epsilon-nets (where the space is hierarchically divided into clusters and only a small number of clusters per level must be examined at query time; sufficient separation between clusters lead to query-time guarantees). Then the entry focuses on the "combinatorial approach", a generalization of the metric space approach where one can only tell whether x is closer to y than z, but there are no explicit distances. A notion similar to that of intrinsic dimensionality (called a "disorder constant" D) can be defined, and several nearest-neighbor algorithms have been derived, whose performance is a function of n and D. This framework allows one to define some subclasses of problems where the nearest-neighbor problem is tractable (including some where a metric cannot be defined) and to solve proximity problems where a metric is hard to define, as is the case in many real-life situations handing complex objects. Several challenges pointing to real-life problems are posed.

Approximate and probabilistic algorithms: An exciting way out to the limits posed by intrinsic dimensionality is to relax the search problem to allow for non-exact solutions. The lower bounds arising from the relation between approximations and intrinsic dimensionality remain unclear. This entry starts by noting that in many cases the correct answer is found quickly by the search algorithm, but then much work is spent in ensuring that this is indeed the correct answer. The entry then argues that approximate solutions are acceptable in most practical applications of

metric searching. Then it proposes to classify the many existing techniques along various dimensions to facilitate comparing them. One important dimension is the approach used to obtain the approximation, where three approaches are identified: mapping the objects to another space where the problem is simpler to handle, using a mapping that preserves proximity as much as possible; not considering some objects that are "unlikely" to belong to the answer (e.g., close to be eliminated by lower bounds on the distances computed); and stopping an exact search algorithm earlier than the necessary to ensure correctness and giving the best answer found so far, trying to consider the more promising objects earlier in the process. The most important challenge posed is finding techniques that offer guarantees on performance and/or quality of the result, either deterministic or probabilistic. A couple of methods are covered, locality-sensitive hashing (LSH) and probably approximately correct queries (PAC), that offer guarantees of those kinds.

Beyond the metric space model: Many applications require even more general similarity models than those provided by metric spaces. In the absence of the triangle inequality, usually only approximate algorithms are possible. This is a largely under explored area. The entry starts by enumerating several relevant applications where some of the metric axioms are violated in the similarity functions that work best. The most frequently questioned axiom is the triangle inequality, which is on the other hand the fundamental property used by all the metric space indexes. One way to handle cases where it does not hold is that domain experts give alternative topological or statistical properties that hold in the database and that allow designing other indexing schemes. Another way is to discover or estimate those kinds of properties by studying a sample of the database. The entry mentions several concrete approaches. Apart from domain-specific ones, general techniques include mapping to a metric space (usually increasing the intrinsic dimensionality), and using approximate techniques directly on the nonmetric space. The identification of general alternative topological features (such as in the "combinatorial framework" advocated in an earlier entry) is, among others, an important challenge posed at the end.

Stronger metric operations: Range and nearest-neighbor search are the most basic queries. These fall short to incorporate a metric object type to state-of-the-art database technology. A missing operation, among others, is the metric join, i.e., find close-enough pairs of points among two sets. The entry starts by enumerating some application areas for similarity joins. Then it refers to the nested loop, where each element of a set is searched for in the other set using an index, as the basic technique, which misses opportunities for optimization due to similar queries. Algorithms improving upon this basic idea make use of spatial clustering to reduce the number of pairs of objects to consider. Then the entry covers some particular algorithms for specific metric spaces.

Real-life performance: The idealized cost model we presented is helpful to understand the fundamentals of the problem. In many applications, however, side costs cannot be disregarded. This is particularly true when the data resides on disk, whose access times are hardly negligible. Other issues are index construction and maintenance, scalability, and distributed processing. This topic is becoming of high interest with the advent of multicore processors and peer-to-peer systems. The entry starts by emphasizing the challenges and opportunities derived from the massive generation of digital content from distributed heterogeneous sources, as well as the distributed storage of digital information searchable by similarity. It covers the M-tree family as the first and best-known indexes to handle similarity search in secondary storage, a must when handling massive

data. Then it points at the problems posed by the parallelization of the M-tree (actually shared by all hierarchical structures) and at the D-index as a hash-based (or cluster-based) index for secondary storage that is, instead, easily parallelizable. The entry then turns into P2P systems, which manage to handle indexing and searching in distributed form with no centralized scheme. It covers several adaptations of existing techniques (such as cluster-based and mapping-based indexes) to P2P architectures. In a more challenging scenario, which occurs in applications, nodes retain total control on their data and will not index data from other nodes. Some systems are described that manage to operate this more restricted version of a P2P network for similarity search. Finally, the entry focuses in the challenges posed by the cost of extracting sufficiently good features from massive data so that good-quality retrieval is possible from the features. As the data volumes increase, massively parallel schemes become mandatory to achieve reasonable indexing performance, and this may reach the point where only stream processing on the data flow is possible.

Applications: The research in metric space indexing is driven by applications such as multimedia searching, computational biology and data mining. It is important to exploit insight knowledge of expert practitioners in order to design successful solutions. Two entries describe applications in computational biology and in data mining. Similarity search arises in many subfields of computational biology. The entry focuses in the case of sequence comparisons used to find homologous regions in DNA and protein sequences. Similarity measures such as PAM and BLOSUM matrices have been developed which have a strong biological meaning, but are not metric. Recently, there have been efforts in deriving metric variants that are still biologically plausible while admitting searches using metric indexes. Protein identification by similarity of mass spectra is another area where metric space methods have been applied.

The entry on data mining focuses on its applications to clustering. It first shows that several clustering algorithms are directly based on distance computations, such as k-nearest-neighbors classifiers, k-medioid clustering, and distance-based outlier detection methods. While the methods themselves usually do not require the space to be metric, faster clustering is possible with metric indexes when the spaces are metric. The entry then covers the problem, which arises in many real cases, of clustering in presence of noise. Practical solutions adapt clustering algorithms to handling uncertain objects, for example by considering mean distances. Finally, the entry considers clustering high-dimensional spaces, which is usually a consequence of the irrelevance of many of the features. This can be addressed by finding out which features are relevant in which areas of the space, via various projection techniques. Most of the problems are, however, basically open.

References

[1] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[2] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers, 2006.

[3] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.