Chapter 2

Similarity search in metric spaces: overview and preliminaries

2.1 Introduction

Traditional databases use exact match to search for data records. Retrieved records are those whose attributes exactly match the ones specified in the query. However, for several applications exact match search is not suitable and similarity search should be used instead. In these cases, data that are similar to the query are retrieved.

In some applications requiring similarity search, data are represented in a vector space. Similarity is obtained by computing the distance between vectors: the smaller the distance, the higher the similarity. Several access methods were defined to improve performance of similarity search for data represented in vector space. These, typically, allows efficient similarity search to be performed when the number of dimension of the vector space is below 15.

There are several other applications where geometric interpretation of distance between vectors cannot be effectively used to asses similarity between represented data or, simply, where data cannot be represented in a vector space. One example

is image similarity. Even though image's features are represented by vectors, for instance color histograms, sometimes vectors that are geometrically distant could be more effectively considered similar, because of the phenomenon of cross-talk between dimensions representing individual colors. An example where data cannot be even represented in a vector space is the string similarity. The similarity between two strings is typically assessed by computing the number of operations needed to transform one into another. Strings cannot be effectively represented by real valued vectors and no geometric interpretation can be inferred from their distance function.

In all these cases geometric properties and information on the coordinates are not available – only pair-wise distances can be computed. For this reason, the study of data represented in metric spaces and access methods for similarity search in metric spaces has recently attracted much attention.

In this section we justify the need of techniques for similarity search, when data are represented in a metric space, and give some preliminary definitions useful for the remainder of the thesis.

We first discuss some applications where exact match search cannot be applied and similarity search should be used instead. We compare similarity search when data are represented in metric spaces and vector spaces, and we discuss the different basic statistic techniques that can be respectively used to build optimized access methods.

Finally we describe the specific data sets that we have used as test beds for validating the techniques proposed in this thesis.

2.2 Similarity search and its applications

Traditional database systems typically manage files of homogeneous records. Each record is composed of a specified set of attributes that are used to characterize the entities they represent. Values associated with attributes are supposed to describe each record unequivocally and completely. For instance records representing customers of a bank may be characterized by the attributes `FirstName`, `FamilyName`, `SocialSecurityNum`, and probably a few others. Each record, by means of the values associated with its attributes, unambiguously represents a particular customer of the bank.

In these systems, typically *exact match search* is used to retrieve interesting records from the database. Queries specify values for a subset of record attributes and only records whose attribute values exactly match those specified in the query are retrieved. Not retrieved records contain at least one attribute, among those specified in the query, whose value differs from that of the query attribute. For instance, a typical query submitted to a traditional database system would be, informally, *retrieve the records corresponding to the customer whose name is John Smith.* In this case, the system would retrieve the records contained in the database, where the value associated with the `FirstName` field is exactly John, and the value associated with the `FamilyName` field is exactly Smith. Eventually, when there are several matching records, the query can be defined to be more selective by adding additional convenient attribute values, like for instance the date of birth, the city of birth, or the social security number.

Exact match search basically partitions the set of records, contained in the database, in two subsets. One, sometime called the *result set*, contains the records that exactly

match the query. The other contains the records where at least one attribute value does not exactly match the corresponding query attribute value. However, there are some applications where partitions distinguishing qualifying from not qualifying records cannot be defined. In some of these applications all records possibly match the query and a "degree" of matching can be computed instead, comparing each record with the query. In these circumstances, the concept of *similarity search* is introduced as opposite to the exact match search, to indicate that retrieved records are those having high degree of matching with the query. Qualifying and not qualifying records can be identified, for instance, by specifying a threshold on the degree of similarity or on the number of records retrieved – see Section 2.4 for a formal definition of similarity search queries. In the following we outline some typical applications where similarity search capabilities are needed.

Multimedia document retrieval: Probably one of the most common application of similarity search is the retrieval of multimedia documents. Application in this area are for instance face recognition, object recognition, finger print matching, voice recognition, and in general multimedia databases [ABH97, YI99]. In this case retrieval is not performed by using directly the raw content of document, for instance, comparing pixel by pixel two images. Documents of the database are processed and salient *features*, which describe the content of documents, are extracted. Search is performed by comparing features extracted from the queries with features extracted from documents of the database. Let us consider image documents for instance. In this cases, typically, the feature extracted are dominant colors, textures, and shapes [Cas96, SO95, Smi97], generally represented as vectors. In case of color features, for instance, the color spaces is

quantized in a certain number of colors, the quantity of each individual color in the whole image is stored in each dimension of the vector, obtaining a color histogram [Smi97]. Searching for images similar to a query image corresponds to retrieve images whose color histogram is similar to the query color histogram.

Computational biology: Another very interesting application comes from the computational biology. The basic objects managed and studied in this area are DNA and protein sequences. Typically they are modeled as huge sequences of symbols, that can be easily represented by very long text strings. The problem here is finding a given sequence of letters, corresponding to some particular characteristic, inside a longer sequence. In most cases it is not possible to find an exact match of the given short sequence, and computational biologist are more interested in finding parts of a longer sequence which are similar to a short sequence. In fact, there could be minor differences in genetic sequences that somehow identify the species which they belong to. In this area, the similarity among sequences is based on the probability of mutations that somehow rearrange the sequences themselves [Wat95].

Web search engines: Web search engines traverse the World Wide Web and index the Web sites that they visit. Users may express queries to search for web sites they are interested in. Suppose, for instance, that a user wants to buy a vacation package to Hawaii that costs less that 1000$. Probably, he would specify the keywords `"vacation Hawaii less than 1000$"`, or `"low price vacation package to Hawaii"`, or perhaps `"please, search for web sites that offer vacation packages to Hawaii that cost less than 1000 dollars"`, or other similar variants. Of course all previous queries do not exactly specify

what the user is interested in and similarity search is needed to retrieve web sites that are judged to be relevant. Web search engines are a particular applications of the more generic *information retrieval systems* [SM83], which allow the retrieval of text documents to be performed by specifying list of keywords as queries.

Filtering and recommender systems: Filtering systems [BC92, MMLP97] and recommender systems [RV97] build and maintain a profile of their users. This profile is built by using various strategies, ranging from analysis of the behaviour of the user [MS94] using the system, to techniques of relevance feedback [All96]. In case of filtering systems, the profile is used to filter out what is not interesting to the user and to provide him only with interesting documents (for instances sport news from new agencies). In case of recommender systems, it is used to suggest something to the users (for instance a book to buy, a web site to visit, an advertisement). In both cases what the system has to do is to compute the similarity between the profile of the user and elements to be suggested or to be filtered.

Stock quotes: A further example is suggested by systems that manage stock quotes, which are typically represented as time series [AFS93]. In most cases, in order to take reliable decisions on investments, it is useful to check if circumstances similar to the current one (represented by some time series) occurred in the past, provided that experts of the fields have defined how to compare time series representing stock quotes. In fact, if a certain situation occurred in the past, evaluating what occurred in the following periods helps to understand what could be the trend in the future.

In addition to the previous examples, several other applications might need the use of similarity retrieval. An exhaustive analysis of these applications is out of the scope of this thesis, however we can broadly identify three circumstances where exact match search provides little help and similarity search should be used instead:

i) data cannot be precisely described and represented

ii) queries cannot be precisely specified

iii) data that exactly match a query do not exist in practice

In several cases, a combination of circumstances i), ii), and iii) occur.

Let us consider case i). Sometimes, it is not possible to have a precise description of represented entities. Consider for instance, images, videos or audio documents. Typically these documents are not searched by directly using their raw content. They are associated with description of their content, generated either manually, or automatically. These description might be subjective and ambiguous, in case of those manually generated, or can be too generic and imprecise, in case of those automatically generated. In both cases, what can be retrieved, is the set of descriptions of documents that present some high degree of coincidence with the query, or simply that are similar to the query.

In case ii) the imprecision applies to the query specification. In some cases, users are not capable of precisely expressing their needs with a query. This may happen, for instance, because the system does not offer expressive and unambiguous query specification facilities, or simply because users have a foggy idea of what they are really looking for. This is the case of all techniques of *query by example*, which are largely used in image retrieval systems, for instance. In these systems, the query

image is either sketched by the user, or an arbitrary preexisting image is used as query. In both cases what the user searches for is a set of images that are somehow similar to the one used as a query.

Case iii) occurs when the probability of finding something that is exactly equal to the query is practically 0. This is, for instance, the case of stock quotes management. Of course, the probability that time series of the past exactly match the current ones is very low. However, as we said above, it is very interesting to retrieve situations similar to the current one.

All applications described above require as a basic functionality the possibility to compute the similarity between representation of data managed by the retrieval system. In the remainder of the thesis we generally refer to these representation as *objects* or *data objects*, without distinguishing and defining the nature of the representations themselves. Accordingly data used as queries will be here called *query objects*. Systems supporting previous applications should be able to efficiently retrieve data objects stored in the database which are similar to the query object.

2.3 From vector spaces to generic metric spaces

In several similarity search applications, data are represented by tuples of real numbers (for instance color histograms). In these cases, data are defined in a *vector space*, or better in a *finite dimensional vector space*, since the number of elements in a tuple is supposed to be finite. Similarity between objects can be indirectly obtained by using the *distance* between vectors: the smaller the distance the higher the similarity.

There are several possibilities to measure the distance between vectors. The most

widely used distance functions, however, are those of the family of Minkowski distances (see Section 2.3.3) where the distance between vectors is interpreted geometrically. For instance, the Euclidean distance belongs to the family of Minkowski distances.

In order to speed up retrieval in large collections of data, with respect the trivial sequential scan, access methods have been developed. The basic idea, which they are based on, is the definition of convenient partitions of objects such that only few subset of objects have to be accessed in order to answer to similarity queries (see Chapter 3 for details). The use of vectors and Minkowski distances offers a great advantage to this purpose since it is possible to exploit geometric and coordinate information to obtain optimal partition and to achieve high performance. In fact, access methods for vector spaces such as k-d-Trees [Ben75, Ben79], R-Trees [Gut84, BKSS90], quad-Trees [Sam95], X-Trees [BKK96], to mention some, rely heavily on geometric and coordinate information. In this context, optimal algorithms have been proposed both for the average and the worst case [BWY80] to answer to nearest neighbor queries. An excellent survey of access methods for data represented in vector spaces can be found in [GG98] and [BBK01]. However, it has been seen that all these techniques degrade their performance when the number of dimensions of the vector space increases. In fact, these access methods suffer of the so called *dimensionality curse*. That is, they have an exponential dependency on the dimensionality of the space. When the number of dimensions is greater than 10–15, a linear scan over the whole data set would perform better [BGRS99, WSB98].

However, in many cases, it has been observed that, even if objects are represented in a high dimensional vector space, fewer dimensions are in fact needed to represent

effectively their geometric distributions [Fal96]. For instance, let suppose that we have objects represented in a 100 dimensional vector space, analyzing them in most case we might notice that they are in fact mostly distributed on an hyper-plane so that, by some ad-hoc transformations, we may preserve the geometric relationships among objects, but we use less dimensions to represent them. This is also the direction followed by some dimensionality reduction techniques that use, for instance, Karhunen-Loeve Transformation (KLT) [Fuk90], Discrete Fourier Transform (DFT) [OS89], Discrete Cosine Transform (DCT) [Kai85], or Discrete Wavelet Transform (DWT) [Cas96], to transform data from a vector space with a high number of dimensions to another with fewer dimensions. All these methods assume that a few dimensions are enough to retain the most important information regarding the represented data objects. However, they might introduce some approximation since not always the geometric relationships are exactly preserved.

In the past, given these established techniques that can be used in vector spaces, the problem of similarity search has always been approached by enforcing the use of vectors to represent objects. However, this strategy, that somehow guarantees that similarity searches can be executed efficiently, does not guarantee effectiveness as well. In fact, in several cases, vectors and Minkowski distances fail to accurately represent similarity relationships among objects. For instance, in the image processing field, several functions used to compare two images, cannot be easily expressed as distance between two vectors. An important example is the *quadratic form distance* [FSA+95, HSE+95, NBE+93] (see Section 2.3.3) that takes into account the so called *cross-talk* between different dimensions. The quadratic form distance does not belong to the family of Minkowski distances and it does not preserve geometric properties

of the vector space which it is applied to. Other important examples that cannot be handled by using vector distances are the string similarity, used for instance to compare DNA sequences in genetic databases, or the set similarity (see again Section 2.3.3).

Notice that when arbitrary distance functions are used to compute similarity between objects (even if represented as vectors), the geometric properties and the coordinates cannot be exploited and, as a consequence, the access methods for vector spaces indicated above cannot be used as well. Recently, the study of objects represented in generic *metric spaces* and consequently the design of access methods for metric spaces, as M-Trees [CPZ97], Slim-Trees [TTSF00], Vantage Point Trees [Chi94, Uhl91, Yia93, Yia99, BÖ99], or GNAT [Bri95] has attracted much attention in the field of similarity search. In metric space the only information that can be obtained about objects is their distance from other objects. Formal definition of metric spaces is given at the end of this section. Of course these new approaches are also able to deal with vector spaces and the Minkowski distances, since these are special cases of metric spaces: only the distances between vectors are considered and the fact that they are vectors is ignored, so no geometric information on coordinates are exploited. By doing this, an immediate advantage is the independence on the dimensionality of objects. In fact, in [CPZ97] is shown that M-Trees perform better than R*-Trees [BKSS90] when applied to high dimensional vector spaces. An excellent survey of techniques for searching in metric spaces is provided in [CNBYM01].

In this thesis we suppose that objects are represented in generic metric spaces, given their generality, their wide applicability, and their increasing demand in the similarity search area. In this way, we are not tied to specific cases of similarity

search problems, as with vector spaces, even if our results can also be applied to them.

In the following we give some definitions that are needed in the remaining of the thesis.

2.3.1 Metric spaces

Let $\mathcal{D}$ be a domain of objects, for instance image features or DNA sequences. Let d a function defined as follows:

$$d : \mathcal{D} \times \mathcal{D} \rightarrow \Re$$

that for each triple of objects $O_x, O_y, O_z \in \mathcal{D}$ satisfy the following properties:

(1) $d(O_x, O_y) = d(O_y, O_x)$ *(symmetry)*

(2) $0 < d(O_x, O_y) < \infty, O_x \neq O_y$ *(positiveness)*

(3) $d(O_x, O_x) = 0$ *(reflexivity)*

(4) $d(O_x, O_y) \leq d(O_x, O_z) + d(O_z, O_y)$ *(triangle inequality)*

We define

$$\mathcal{M} = (\mathcal{D}, d)$$

a *metric space*. $\mathcal{D}$ is the domain of the metric space, while d is the distance function of the metric space. For practical reasons, in this thesis, we assume that the maximum possible distance between objects of the considered domain $\mathcal{D}$ never exceeds a value d_m, thus we consider a *bounded metric space*. This is not a restriction, since all practical applications, which we have dealt with up to now, consider distances to be bounded.

Notice that there are no assumptions on the nature of the objects of the domain $\mathcal{D}$. There are only some constraints on the distance function d. It is easy to show that the family of Minkowski distances satisfies these properties, so a vector space is also a metric space when they are used to measure the vector distances.

In the rest of the thesis, when we use the the term *distance function* we suppose that it always satisfies the properties defined above. We also use simply $\mathcal{M}$ to indicate a metric space, that implicitly is defined by a domain $\mathcal{D}$ of objects and by a distance function d.

2.3.2 Metric ball regions

Access methods for vector spaces, typically partition the data set and bound elements of the partition by hyper-rectangular region. In metric space, the notion of hyper-rectangle cannot be used, since it requires the use of coordinates. However we may use the notion of ball regions. Given an object O in a metric space $\mathcal{M}$, we define a *ball region* $\mathcal{B}(O, r)$ as

$$\mathcal{B}(O, r) = \{O_i \in \mathcal{D} \mid d(O, O_i) \leq r\}$$

A ball region $\mathcal{B}(O, r)$ is determined by a center $O \in \mathcal{D}$ and a radius $r \geq 0$. It includes such objects from $\mathcal{D}$ so that their distances from O are less than or equal to r.

2.3.3 Similarity and distance functions

Computing similarity between objects can be done in several ways, depending on their specific representation. Similarity functions are generally defined as follows:

$$sim : \mathcal{D} \times \mathcal{D} \rightarrow [0, 1]$$

given two objects of the considered domain, their similarity is a real value in the interval between 0 and 1.

However, as previously observed, a suitable formalization for the similarity search problem is to represent objects in a metric space and to search for objects whose distance from the query object is small. Accordingly to this formalization, a distance function that satisfies the metric postulates should be provided instead of the similarity function. In several cases, the distance function is obtained naturally from the definition of the specific similarity search application. In other cases, where only the similarity function is available, the distance function should be obtained from it. Intuitively, the higher the similarity the smaller the distance, therefore, given a similarity function sim, a distance function d can be obtained as follows:

$$d(O_1, O_2) = 1 - sim(O_1, O_2), \tag{2.3.1}$$

where O_1 and O_2 are two objects of $\mathcal{D}$. When d satisfies the metric postulates, $\mathcal{M} = (\mathcal{D}, d)$ is a metric space.

The similarity function can be obtained from a distance function as well, provided the metric space is bounded, that is, provided that a distance between two objects never exceeds a fixed distance d_m:

$$sim(O_1, O_2) = \frac{(d_m - d(O_1, O_2))}{d_m}.$$

Relationships between similarity and distance function are discussed in [CPZ98b].

In the following we discuss some of the most important metric distance functions and similarity functions that result in metric distance functions when Equation 2.3.1 is applied.

Minkowski distances: Let us first, consider the case where the objects are real

valued vectors. Let us suppose that we have two vectors v_x and v_y defined in a vector space with *dim* dimensions. This implies that both v_x and v_y are tuples composed of *dim* real values. The traditional way of computing similarity among objects represented in a vector space is to use a function of the family of the Minkowski distances. This set of distance function is often designated as the L_p distance and it is defined for vectors v_x and v_y as

$$L_p(v_x, v_y) = \left(\sum_{j=1}^{dim} \mid v_x[j] - v_y[j] \mid^p \right)^{1/p}, p \geq 1.$$

Varying the value of p, we may enumerate the various function of this family. For instance, L_1 is known as the *city-block* or *Manhattan* distance, and it corresponds to the sum of the distances along each individual dimension of the considered vector space. L_2 is the famous and intuitive *Euclidean* distance. Another interesting function of this family is the L_∞, which corresponds to compute the limit of L_p, when p goes to ∞. L_∞ is the maximum among the distances along each individual coordinate:

$$L_\infty = (v_x, v_y) = \max_{j=1}^{dim} \{ \mid v_x[j] - v_y[j] \mid \}.$$

Figure 2.1 sketches how the shape of a ball region $\mathcal{B}(O, r)$ changes when different L_p's are used.

Quadratic form: The family of the Minkowski distances considers different vector coordinates to be independent, thus distances are strictly related to the closeness of vectors in a multi-dimensional space. However, in many cases, vector coordinates can be dependent or correlated. Good examples of such type data

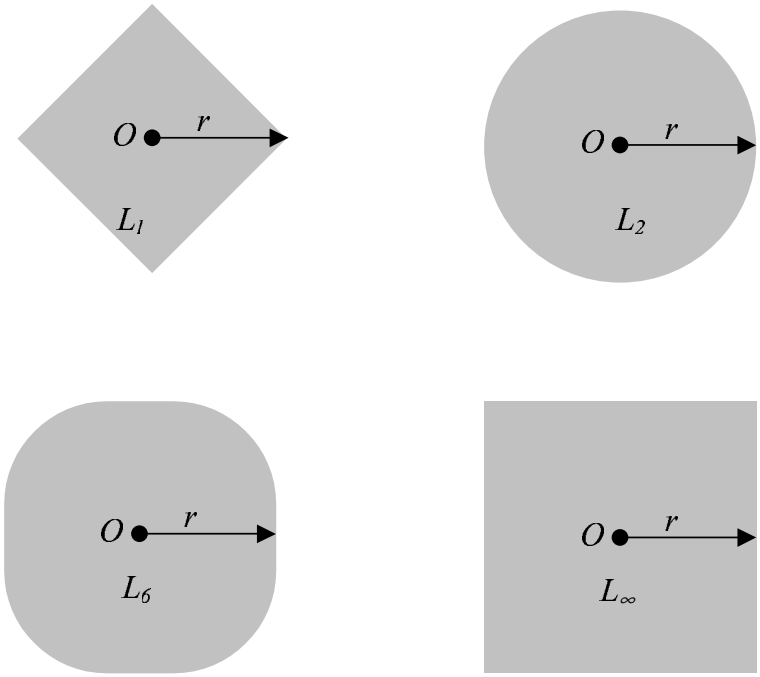


Figure 2.1: Shape of a ball region $\mathcal{B}(O, r)$, in a two dimensional vector space, when respectively L_1, L_2, L_6, L_∞, are used as distance functions.

are color histograms [Smi97] with each dimension representing a different color. Obviously, orange and pink are certainly more similar than red and blue. In order to effectively measure the distance between color histograms, this natural (though also subjective) *cross-talk* of dimensions should be taken into account properly [Fal96]. One way to handle this problem is to use the *quadratic-form* distance [FSA+95, HSE+95, NBE+93].

$$d_{qf}(v_x, v_y) = \sqrt{(v_x - v_y)^T \mathbf{A} (v_x - v_y)},$$

where $\mathbf{A} = [a_{i,j}]$ is a correlation matrix between dimensions of vectors v_x and v_y, and the superscript T indicates the vector transposition. Naturally, there is no direct correspondence between distances and positions of vectors in the space. If the correlation matrix $\mathbf{A}$ is symmetric and its diagonal is 1, that is

when $a_{i,j} = a_{j,i}$ and $a_{i,i} = 1$, then d_{qf} satisfies the metric properties.

String distance: There are several cases where objects can be effectively represented by strings of arbitrary length and retrieval is performed by searching for string that partially match a query string [HD80, Nav01]. This is quite common in computational biology [Wat95], signal processing [Lev65], text retrieval [Dam64], for instance. In all these applications distance among two strings s_1 and s_2 is defined as the minimal cost of a sequence of operations that transform s_1 into s_2. The cost is ∞ when that transformation sequence does not exist. The cost of a sequence of operations is computed as the sum of the cost of the individual operations. The most common distance functions are defined as sequences of some of the following operations:

op_1: insert a letter

op_2: delete a letter

op_3: replace a letter

op_4: swap adjacent letters

Depending on which of the previous operation are allowed, different distance functions are defined:

- *Levenshtein or Edit distance* [Lev65]: allows insertions and replacements.
- *Humming distance* [SK83]: allows only replacements.
- *Longest Common Subsequence distance* [NW70, AG87]: allows only insertions.

It is easy to show that these distance functions satisfy the metric postulates, when all individual operations is given the same cost.

Set similarity: Another interesting case is the *similarity of sets*. Given two sets $\mathcal{S}_1$ and $\mathcal{S}_2$, which contains arbitrary elements, a largely used similarity function is defined as the ratio of the number of their common elements to the number of all different elements

$$S_T(\mathcal{S}_1, \mathcal{S}_2) = \frac{\#(\mathcal{S}_1 \cap \mathcal{S}_2)}{\#(\mathcal{S}_1 \cup \mathcal{S}_2)},$$

where $\#(\mathcal{S})$ is the number of elements in set $\mathcal{S}$.

Tanimoto similarity: A generalization of the set similarity is the *Tanimoto* similarity measure [Koh84]. It is used for instance to compute the similarity between structural features of biological data [GPR97]. Let us suppose that we have vectors v_x and v_y defined in a vector space. Let us indicate with $v_x \cdot v_y$ the *scalar product* of v_x and v_y, and with $\|v_x\|$ the *Euclidean norm* of v_x. The *Tanimoto* similarity is defined as

$$S_T(v_x, v_y) = \frac{v_x \cdot v_y}{\|v_x\|^2 + \|v_y\|^2 - (v_x \cdot v_y)}.$$

When vectors contain only 0's and 1's, the similarity of sets is obtained. In this case 0 means that an element, identified by a vector position, does not belong to the set, while 1 means that it belongs to it. Notice that the distance function, obtained in correspondence of the Tanimoto similarity, satisfies the metric properties but it does not preserve the geometric closeness of vectors.

Hausdorff distance: As a final example, consider the *Hausdorff distance*, which is used to compare shapes of images [HKR93]. Here the compared objects are sets

of relevant points, for example high curvature points. Given two finite point sets $A = \{a_1, a_2, \ldots, a_p\}$ and $B = \{b_1, b_2, \ldots, b_q\}$, the Hausdorff distance is

$$H(A, B) = \max(h(A, B), h(B, A)),$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \| a - b \|,$$

and $\| \cdot \|$ is some underlying norm on the points of A and B, for example the L_2 or Euclidean norm. Also this distance function satisfy the metric postulates.

2.3.4 Statistical information on data

The use of statistical information on data has always played a key role for the performance optimization of database systems. Statistical information is in fact at the basis of many cost models used for the optimization of query execution. In addition, it is also used to better organize the structure of some access methods for performance improvement. There are some differences on the type of statistics that can be obtained depending on the fact that geometric information can be used, as with vector spaces and distance of the Minkowski family, or no geometric information is available, as in generic metric spaces. In the first case, statistics on the position of the objects in the space can be exploited. In the second case, only information on the distances can be used and no geometric information on data can be exploited. In the following we first introduce the probabilistic notions of density and distribution functions then we discuss how these applies to our scenario to characterize data represented in metric spaces.

Distribution and density functions

Let suppose that V is a continuous random variable [HPS71]. Mathematically it is a real-valued function, defined on a probability space, which takes real values, depending on some event whose probability is 0.

The *distribution function* F_V of a random variable V is the following probability:

$$F_V(v) = \mathtt{Pr}\{V \leq v\}$$

For instance, let us suppose that V is a continuous random variable associated with the height of people, then $F_V(v)$ is the probability that chosen a random person (the event), its height is smaller than v. Notice that, chosen a random person, the probability that its height is *exactly* v is 0 of course, given that v is a real value.

The *density function* f_V of a random variable V is a function such that

$$\int_{-\infty}^{+\infty} f_V(v)dv = 1$$

Of course, we have that

$$F_V(x) = \int_{-\infty}^{x} f_V(v)dv$$

When the random variables are more than one, for instance V_1 and V_2, then we have respectively the *joint distribution*, $F_{V_1V_2}(v_1, v_2)$, and the *joint density*, $f_{V_1V_2}(v_1, v_2)$. The joint distribution is defined as follows

$$F_{V_1V_2}(v_1, v_2) = \mathtt{Pr}\{V_1 \leq v_1 \wedge V_2 \leq v_2\}$$

The joint density $f_{V_1V_2}(v_1, v_2)$ is a function such that

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_{V_1 V_2}(v_1, v_2) dv_1 dv_2 = 1$$

As before, we have that

$$F_{V_1 V_2}(a, b) \int_{-\infty}^{b} \int_{-\infty}^{a} f_{V_1 V_2}(v_1, v_2) dv_1 dv_2 = 1$$

This approach can be extended to an arbitrary number of random variables.

Distance distribution versus data distribution

When a data set contains objects represented in a vector space, a useful property that can be obtained by analyzing it is the *data distribution*. By using it is possible to know, for each arbitrary region of the space, what is expected amount of object contained in it or, in probabilistic terms, what is the probability that a random object belongs to the region. This can be obtained by integrating the *data density function* over the given region. For instance, Figure 2.2 shows the data density function, say $f_{XY}(x, y)$, in a two dimensional vector space, where X and Y are continuous random variables corresponding to the x and y coordinates of objects. In the Figure, dark values correspond to high values of $f_{XY}(x, y)$, while light values correspond to low values. Data density, or data distribution, has been used on vector spaces to characterize data sets for various purposes, as for instance to develop effective cost models [BKK97, FK97, KF93, PM97, TS96] for multi-dimensional access methods such as R-Trees [Gut84] or R*-Trees [BKSS90].

In this thesis we assume that data objects are represented in a generic metric space, and vector spaces are just a special case. As discussed in Section 2.3.1, metric spaces allow only to measure distance between two objects, and in particular no

coordinate systems can be used. This, among other things, has the implication that no information on data distribution can be exploited. In fact, data distribution cannot be obtained since an object in a metric space does not have an identifiable position and the only thing that can be known is its distance from other objects.

The characterization of generic metric spaces requires a novel approach. In [CPZ98a] the use of the *distance distribution* is proposed as a counterpart of data distribution used for vector spaces, since it is the "natural" way to characterize metric spaces. Given an object O_i, the distribution of distances from O_i, characterize how the other objects are distributed around O_i. For instance if we report these observations to a two dimensional vector space the result is what is sketched in Figure 2.3. That is given the distance distribution or the distance density with respect to an object O_i, we might know what is the amount of objects whose distance from O_i is smaller than a certain value or, in probabilistic terms, what is the probability that a random object has a distance from O_i smaller than a certain value. However no information on the "dense" zones of the space is available, since an object whose distance from O_i is d, may be on any point on the circumference with center O_i and radius d. If the vector space has three dimensions the circumference become a sphere and so on. In particular, in a "pure" metric space, no guesses on the position of the object can be made, since there is not position. The distribution of distances from O_i is sometime called the O_i's *viewpoint*.

In addition to the distribution of distances from a specific object, another property that can be easily computed is the *overall distance distribution*. Given a distance d the overall distance distribution says what is the probability that distances smaller than d exist, that is, what is the probability that given two random objects their

distance is smaller than d.

Managing the overall distance distribution is far more easy than managing the viewpoint of every possible object of the data set. In [CPZ98a] it is shown that the overall distance distribution can be used, instead of the viewpoints, when the data set is probabilistically *homogeneous*, that is when there is no significant *discrepancy* between the various viewpoints. In case of highly non homogeneous spaces, the viewpoint of a certain individual object is significantly different that that of a different object. However, it was found that real and "realistic" synthetic data sets are highly homogeneous so the overall distance distribution can be reliably used in practice to characterize them. These notions are more formally discussed in the following.

Let D_O be a continuous random variable corresponding to the distance $d(O, \mathbf{O_1})$, where $\mathbf{O_1}$ is a random object. The *distance density* $f_{D_O}(x)$ represents the probability[1] of distance x from object O. The corresponding *distance distribution* $F_{D_O}(x)$, that is the probability of a distance to O of being at most x, can be determined as

$$F_{D_O}(x) = \int_0^x f_{D_O}(t)\, dt \tag{2.3.2}$$

Given two different objects $O_x, O_y \in \mathcal{D}$, the corresponding viewpoints $F_{D_{O_x}}$ and $F_{D_{O_y}}$ are different functions.

In the remaining of this thesis we will simply use F_{O_i}, to indicate the distance distribution or the viewpoint with respect to the object O_i.

The overall distribution of distances over $\mathcal{D}$ can be defined as

$$F(x) = \mathtt{Pr}\{d(\mathbf{O_1}, \mathbf{O_2}) \leq x\}, \tag{2.3.3}$$

[1]We are using continuous random variables so, to be rigorous, the probability that they take a specific value is by definition 0. However, in order to simplify the explanation, we slightly abuse the terminology and use the term probability to give an intuitive idea of the behavior of the density function being defined.

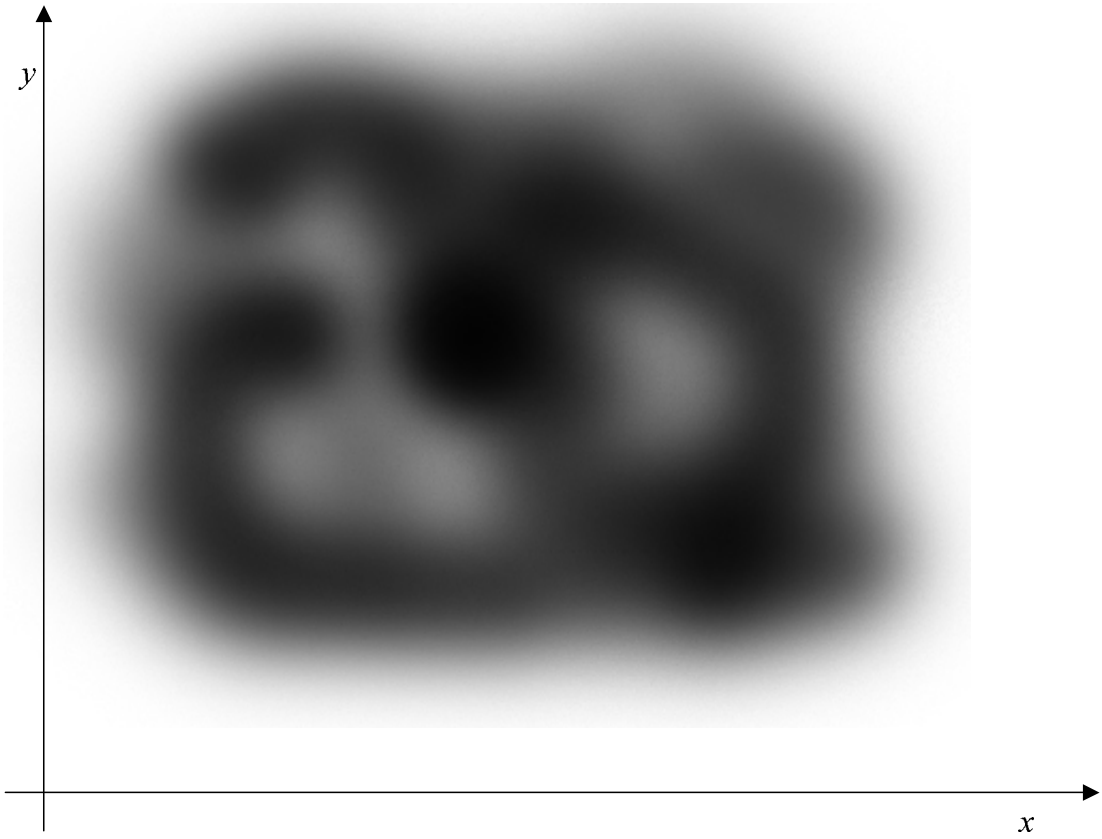


Figure 2.2: Density of data in a two dimensional vector space

where $\mathbf{O_1}$ and $\mathbf{O_2}$ are two independent random objects of $\mathcal{D}$.

Th *discrepancy* between two viewpoints is defined as

$$\delta(F_{O_i}, F_{O_j}) = Avg[\mid F_{O_i}(\mathbf{x}) - F_{O_j}(\mathbf{x}) \mid] \tag{2.3.4}$$

where $\mathbf{x}$ is a random distance in the interval $[0, d_m]$, and d_m is the maximum distance between two objects of the dataset. The discrepancy between two viewpoints is the average difference of values that they assume varying the distance.

The *index of homogeneity of viewpoints* HV of a metric space $\mathcal{M}$ is defined as

$$HV(\mathcal{M}) = 1 - Avg[\delta(F_{\mathbf{O_1}}, F_{\mathbf{O_2}})]$$

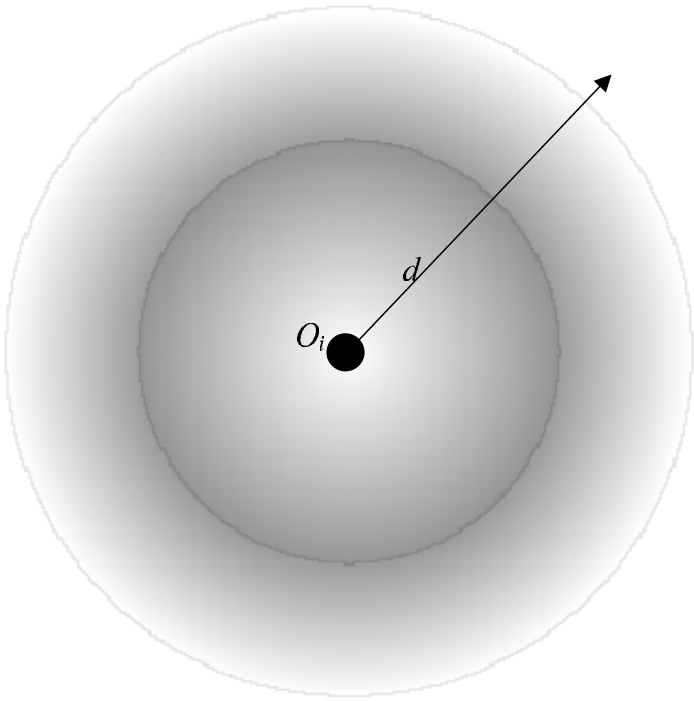


Figure 2.3: Density of distances from the object O_i

where $\mathbf{O_1}$ and $\mathbf{O_2}$ are random points of $\mathcal{U}$. The index of homogeneity of viewpoints of a metric space is 1 minus the average discrepancy of the viewpoints.

When $HV(\mathcal{M}) \approx 1$, two different relative distance distributions are likely to behave similarly and in [CPZ98a] was shown that the overall distance distribution $F(x)$ can be used as the representative instead of any F_{O_i}.

As we said before, for real and realistic synthetic data sets, including the ones that we have used in this thesis 2.5, the index of homogeneity was found to be close to 1. Accordingly, we use the overall distance distribution $F(x)$, in our experiments, for any reference object O_i.

Notice that, even if we neglect the computational complexity of a procedure that would be able to determine $F(x)$, all objects from $\mathcal{D}$ are usually not known. Therefore,

an estimation of $F(x)$ is computed, by considering distances between objects from a representative sample of $\mathcal{D}$.

2.4 Similarity queries

The most frequently used types of similarity queries are *similarity range* and the *nearest neighbors* queries. Accordingly, in this thesis we concentrate our efforts on these two, even if other forms of similarity queries exist, as for instance the *all pairs* or the *all nearest neighbors* queries.

In similarity search applications, in principle, it is not possible to distinguish the set of objects that qualify from those that do not qualify. In fact, all objects qualify to a similarity query and the only possibility is to asses the degree of similarity between each object and the query. The result is that, theoretically, all objects should be retrieved in correspondence of a similarity query. Of course this is not realistic for practical applications, and the possibility of distinguishing between qualifying and not qualifying objects should be somehow enforced. There are two basic techniques to obtain this, in correspondence of which similarity range and nearest neighbors queries are defined.

The first possibility, which results in similarity range queries, is to use a threshold on the degree of similarity with the query, or correspondingly on the distance from the query. Qualifying objects are those whose similarity with the query is above the specified threshold, or correspondingly, whose distance is below the specified threshold.

The second possibility, which results in nearest neighbors queries, is to specify the maximum number of objects to be retrieved. Let us suppose that k is the number of

object to be retrieved. Qualifying objects are the k most similar objects to the query, or correspondingly, the k objects closest to the query.

In the first case, the response set is determined by the maximal dissimilarity with respect to the query, while in the second case the response is constrained by the maximal number of closest objects. In the following a formal definition of similarity range and nearest neighbors queries is given.

Let us suppose that we have a data set $\mathcal{DS} \subseteq \mathcal{D}$ in metric space $\mathcal{M} = (\mathcal{D}, d)$ and a query object $O_q \in \mathcal{D}$. The similarity range query $\mathbf{range}(O_q, r_q)$ for query object O_q and search radius r_q is defined as

$$\mathbf{range}(O_q, r_q) = \{O_i \in \mathcal{DS} \mid d(O_q, O_i) \leq r_q\}. \tag{2.4.1}$$

Notice that a range queries is in fact defined by the ball region $\mathcal{B}(O_q, r_q)$, that we will call the *query region*.

The nearest neighbors query $\mathbf{nearest}(O_q, k)$ for query object O_q and k nearest neighbors is defined as

$$\begin{aligned} \mathbf{nearest}(O_q, k) \quad = \{O_i \in \mathcal{DS} \mid \quad & 1 \leq i \leq k \wedge \\ & \forall j < k, d(O_j, O_q) \leq d(O_{j+1}, O_q) \wedge \\ & \forall j > k, d(O_k, O_q) \leq d(O_j, O_q)\}. \end{aligned} \tag{2.4.2}$$

Similarity response sets are ordered (sorted, ranked) sets, where the position of an object in the set is determined by its distance with respect to O_q.

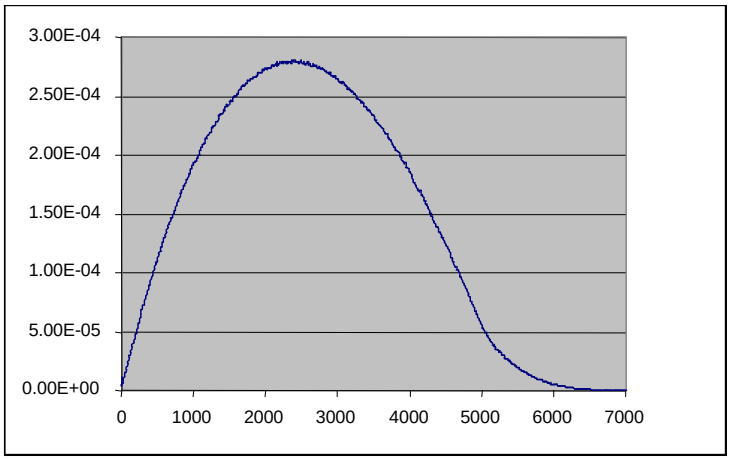
Some access methods for similarity search partition the data set into disjoint subsets which are characterized by bounding regions. Such aggregate information is used for pruning regions (subsets) that cannot contain qualifying objects. When a range query is considered, it is easy to imagine that regions having no overlap with the query region can safely be pruned.

In fact, the situation is quite similar even in case of the nearest neighbor queries. By considering the distance to the k-th nearest neighbor, we can transform the nearest neighbors query to a range query. The only problem is that this radius is not known in advance, so the query region. However, good nearest neighbor search strategies only access the regions that have an overlap with this theoretic query region. In order to achieve this behaviour, typically a priority queue of candidate regions is maintained, and the regions are accessed starting with the most promising one.

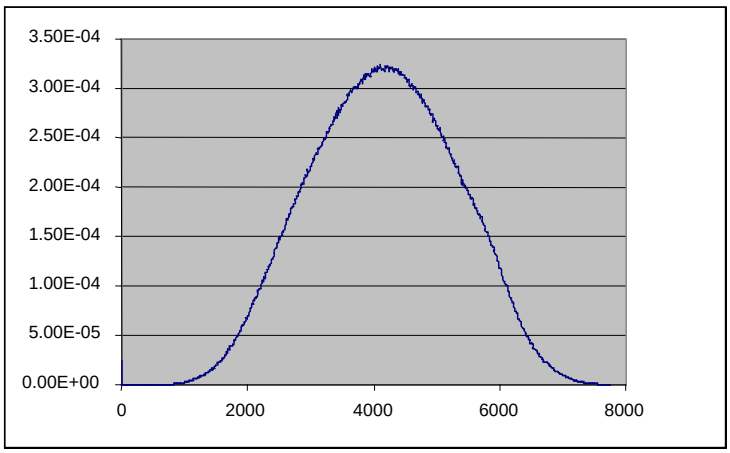
Details of similarity search algorithms that implement Equation 2.4.1 and 2.4.2 using access methods are discussed in Chapter 3.

2.5 Data sets used in this thesis

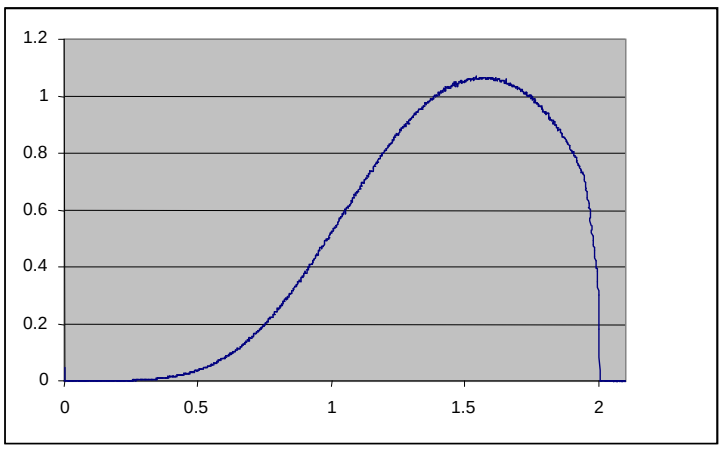
We have conducted our experiments using three data sets: one synthetic and two real-life data sets of image color features. The synthetic data set, called UV, is a set of vectors uniformly distributed in a 2-dimensional space where vectors are compared through the Euclidean (L_2) distance. The second data set, designated as HV1, represents color features of images. In this case, the features are represented as 9-dimensional vectors containing the *average*, *standard deviation*, and *skewness of pixel values* for each of the red, green, and blue channels [SO95]. An image is divided into five overlapping regions, each one represented by a 9-dimensional color feature vector. This results in a 45-dimensional vector as a descriptor of one image. The distance function to compare two feature vectors is again the Euclidean (L_2) distance. The third data set, called HV2, contains color histograms represented in 32-dimensions. This data set was obtained from the UCI Knowledge Discovery in Databases Archive, [HB99]. The color histograms were extracted from the Corel



UV Data set



HV1 Data set



HV2 Data set

Figure 2.4: Overall distance density functions of the data sets used for the experiments

image collection as follows. The HSV space is divided into 32 subspaces (colors), using 8 ranges of hue and 4 ranges of saturation. The value for each dimension of a vector is the density of each color in the entire image. The distance function used to compare two feature vectors is the histogram intersection implemented as L_1. Data sets UV and HV1 contain 10,000 objects, while HV2 contains 50,000 objects.

The range of distances and the corresponding overall distance density functions can be seen in Figure 2.4. Note the differences in densities of the individual data collections that were selected to test proximity for qualitatively different metric spaces, i.e. spaces characterized by different data (distance) distributions.

Notice that even if these data sets are defined in a vector space, we ignore this and in the remaining of the thesis we deal with them as if they where defined in a generic metric space. For instance, we never make any assumption on data distribution, and only distance distribution is used when needed.