Proc. 20th Int. Conf. on Scientific and Statistical Database Management (SSDBM'08), Hong Kong, China, 2008.

ProUD: Probabilistic Ranking in Uncertain Databases

Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz
{bernecker,kriegel,renz}@dbs.ifi.lmu.de

Institute for Informatics, Ludwig-Maximilians-Universität München, Germany

Abstract. There are a lot of application domains, e.g. sensor databases, traffic management or recognition systems, where objects have to be compared based on vague and uncertain data. Feature databases with uncertain data require special methods for effective similarity search. In this paper, we propose an effective and efficient probabilistic similarity ranking algorithm that exploits the full information given by inexact object representations. Thereby, we assume that the objects are given in form of discrete probabilistic object locations in particular several object snapshots with confidence values. Based on the given object representations, we suggest diverse variants of probabilistic ranking schemes. In a detailed experimental evaluation, we demonstrate the benefits of our probabilistic ranking approaches. The experiments show that we can achieve high quality query results while keeping the computational cost quite small.

1 Introduction

Similarity ranking is one of the most important query types in feature databases. A similarity ranking query iteratively reports objects in descending order of their similarity to a given query object. The iterative computation of the answers is very suitable for retrieving the results the user could have in mind. This is a big advantage of ranking queries against the most prominent similarity queries, the distance-range (ε-range) and the k-nearest neighbor query, in particular if the user does not know how to specify the query parameters ε and k.

Many modern applications have to cope with uncertain or imprecise data. Example applications are location determination and proximity detection of moving objects, similarity search and pattern matching in sensor databases or personal identification and recognition systems based on video images or scanned image data. The importance of this topic in the context of database systems is demonstrated by the increasing interest of the database research community in this topic. Several approaches coping with uncertain objects have been proposed [2, 3, 8, 1]. All these approaches use continuous probability density functions (pdfs) for the description of the spatial uncertainty while the approaches proposed in [5, 6] use discrete representations of uncertain objects. The approach proposed in [5] supports probabilistic distance range queries on uncertain objects. In [6] efficient methods for probabilistic nearest-neighbor queries are proposed. However, in fact only one-nearest neighbor queries are supported.

Similarity search in conjunction with multimedia data like images, music, or data from personal identification systems like face snapshots or fingerprints commonly involves distance computations within the feature space. If exact features cannot be generated from uncertain objects, we have to cope with *positionally* uncertain vectors in the feature space (i.e. objects are represented by ambiguous feature vectors). Basically, there exist two forms of representations of *positionally* uncertain data: Uncertain positions represented by a probability density function (pdf) or uncertain positions drawn by samples. In this paper we concentrate on uncertain objects represented by a set of sample positions, each associated with a confidence value. The confidence values indicate how well the corresponding sample matches the exact object. This form of representation is motivated by the fact that we often have only discrete but ambiguous object information as usually returned by common sensor devices, e.g. discrete snapshots of continuously moving objects.

A probabilistic ranking on uncertain objects computes for each object $o \in \mathcal{D}$ the probability that o is the K-th nearest neighbor ($1 \leq K \leq |\mathcal{D}|$) of a given query object q. In the context of probabilistic ranking queries we propose diverse forms of ranking outputs which differ in the order the objects are reported to the user. Furthermore, we suggest diverse forms in which the results are reported (i.e. which kind of information is assigned to each result).

2 Problem Definition

In this section, we formally introduce the problem of probabilistic ranking queries on uncertain objects. We first start with the definition of (positionally) uncertain objects.

2.1 Positionally Uncertain Objects

Objects of a d-dimensional vector space $\mathbb{R}^d$ are called *positionally uncertain*, if they do not have a unique position in $\mathbb{R}^d$, but have multiple positions associated with a probability value. Thereby, the probability value assigned to a position $p \in \mathbb{R}^d$ of an object o indicates the likelihood that p is the best of all representations for o. A formal definition is given in the following:

Definition 1 (positionally uncertain object). *Let $\mathcal{D}$ be a database of objects located in a d-dimensional feature space $\mathbb{R}^d$. An object $o_i \in \mathcal{D}$ is called* positionally uncertain*, iff the object cannot be assigned to a unique position in $\mathbb{R}^d$. A positionally uncertain object o_i is represented by a set of M sample points $\mathcal{S}(o_i) = \{o_{i,1}, .., o_{i,M}\}$, where $o_{i,j} \in \mathbb{R}^d$ ($1 \leq j \leq M$).*

Let us note that in many applications the positionally uncertain objects are already given in the discrete representation, i.e. by a set of sample points, in particular if the objects are derived from a sequence of sensor signals, e.g. in object tracking systems. Otherwise, we use the generally applicable concept of Monte-Carlo sampling to generate the set of samples according to a given continuous probability density function.

In the remainder, we call *positionally uncertain objects* simply *uncertain objects* and use both notions alternately.

2.2 Distance Computation for Uncertain Objects

Positionally uncertain objects involve uncertain distances between them. Like the uncertain position, the distance between two uncertain objects (or between two objects where at least one of them is an uncertain object) can be described by a probability density function (pdf) that reflects the probability for each possible distance value. However for uncertain objects with discrete uncertainty representations we need another form of distance.

Definition 2 (uncertain distance).
Let $o_i = \{o_{i,1}, \ldots, o_{i,M}\} \in \mathcal{D}$ *and* $o_j = \{o_{j,1}, \ldots, o_{j,M}\} \in \mathcal{D}$ *be two uncertain objects, each represented by* M *sample points and let* $dist : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_0^+$ *be a distance function. Then an* uncertain distance $d_{uncertain}$ *between two uncertain objects* o_i *and* o_j *is a collection of* M^2 *distance samples as defined below*

$$d_{uncertain}(o_i, o_j) = \{dist(o_{i,m}, o_{j,n}) | 1 \leq m \leq M, 1 \leq n \leq M\},$$

where $dist()$ is a L_p-norm based similarity distance.

The probability that the distance $d_{uncertain}(o_i, o_j)$ between two uncertain objects o_i and o_j is smaller than a given range $\varepsilon \in \mathbb{R}_0^+$ can be estimated by:

$$P(d_{uncertain}(o_i, o_j) \leq dist) = \frac{|\{d \in d_{uncertain}(o_i, o_j) | d \leq dist\}|}{|d_{uncertain}(o_i, o_j)|}.$$

Since distance computations between uncertain objects are very expensive, we need computationally inexpensive distance approximations to reduce the candidate set in a filter step. For this reason, we introduce distance approximations that lower and upper bound the uncertain distance between two uncertain objects.

Definition 3 (minimal object distance).
Let $o_i = \{o_{i,1}, o_{i,2}, .., o_{i,M}\}$ *and* $o_j = \{o_{j,1}, o_{j,2}, .., o_{j,M'}\}$ *be two uncertain objects. Then the distance* $d_{min}(o_i, o_j) = \min_{s=1..M, s'=1..M'}\{dist(o_{i,s}, o_{j,s'})\}$ *is called minimal distance between the objects* o_i *and* o_j.

Likewise, we can define an upper distance bound for uncertain objects.

Definition 4 (maximal object distance).
Let $o_i = \{o_{i,1}, o_{i,2}, .., o_{i,M}\}$ *and* $o_j = \{o_{j,1}, o_{j,2}, .., o_{j,M'}\}$ *be two uncertain objects. Then the distance* $d_{max}(o_i, o_j) = \max_{s=1..M, s'=1..M'}\{dist(o_{i,s}, o_{j,s'})\}$ *is called maximal distance between the objects* o_i *and* o_j.

2.3 Probabilistic Ranking on Uncertain Objects

The output of probabilistic queries is usually in form of a set of result objects, each associated with a probability value indicating the likelihood that the object fulfills the query predicate. However, in contrast to ε-range queries and k-nn queries, ranking queries do not have such an unique query predicate, since the query predicate changes with each ranking position. In case of a ranking queries, to each result object a set of probability values is assigned, one for each ranking position. We call this form of ranking output *probabilistic ranking*.

Definition 5 (probabilistic ranking). *Let q be an uncertain query object and $\mathcal{D}$ be a database containing $N = |\mathcal{D}|$ uncertain objects. An* uncertain ranking *is a function $prob\_ranked_q : (\mathcal{D} \times \{1, .., N\}) \rightarrow [0..1]$ that reports for a database object $o \in \mathcal{D}$ and a ranking position $k \in \{1, .., N\}$ the probability which reflects the likelihood that o is at the k^{th} ranking position according to the uncertain distance $d_{uncertain}(o, q)$ between o and the query object q in ascending order.*

If the result of the probabilistic ranking is reported to the user in its raw form, the user could be overstrained with ambiguous ranking results. For this reason, we suggest an unambiguous ranking based on the information given by the probabilistic ranking. The following proposed unambiguous ranking can be built in a post processing step. Our unambiguous ranking *PRQ_MAC* assigns each object o a unique ranking position k by aggregating over the confidences of all prior ranking positions $i < k$ according to o.

Definition 6. *A* probabilistic ranking query based on maximal aggregated confidence *(PRQ_MAC) incrementally retrieves for the next ranking position $i \in \mathcal{I}_N$ a result tuple of the form $(o, \sum_{j=1..i} prob\_ranked_q(o, j))$, where $o \in \mathcal{D}$ has not been reported at previous ranking iterations (i.e. at ranking positions $j < i$) and $\forall p \in \mathcal{D}$ which have not been reported at previous ranking iterations, the following statement holds:*

$$\sum_{j=1..i} prob\_ranked_q(o, j) \geq \sum_{j=1..i} prob\_ranked_q(p, j).$$

3 Probabilistic Ranking Algorithm

The computation of the probabilistic ranking is very expensive and is the main bottleneck of the probabilistic ranking queries proposed in the previous section. In the following, we assume that each object is represented by M sample points. Furthermore, we assume that the object samples are stored in a spatial index structure like the R*-tree [7], in order to organize the uncertain objects such that proximity queries can be efficiently processed.

In the following, we concentrate on the computation of the probabilistic ranking query according to one sample point $q_j \in \mathbb{R}^d$ of the query object q. The computation is done for each sample point of the query object separately and, in a postprocessing step, the results are then easily merged by building the average, to obtain the final result.

3.1 Iterative Probability Computation

Initially, an iterative computation of the nearest neighbors of q_j w.r.t. the sample points of all objects $o \in \mathcal{D}$ (sample point ranking $rank_s(q_j)$) is started using the ranking algorithm proposed in [4]. Then, we iteratively pick object samples from the sample point ranking $rank_s(q_j)$ according to the query sample point q_j. For each sample point $o_{i,s}$ ($1 \leq s \leq M$) returned from $rank_s(q_j)$, we immediately compute the probability that $o_{i,s}$ is the k^{th} nearest neighbor of q_j for all k ($1 \leq k \leq i$). Thereby, all other samples $o_{i,t}$ ($t \neq s$) of object o_i have to be ignored due to the sample dependency within an object as mentioned above.

For the computation of the probabilistic ranking we need a table called *probability table* (PT) which is used to maintain the intermediate results w.r.t. $o_{i,s}$ and which finally contains the overall results of the probabilistic ranking.

Probability Table (PT) The *probability table* stores for each object o_i and each $k \in \mathbb{N}$ ($1 \leq k \leq N$) the actual probability that o_i is the k^{th}-nearest neighbor of the query sample point q_s. The entries of PT according to the s^{th} sample point of object o_i are defined as follows:

$$PT[k][i][s] = P((k{-}1) \text{ objects } o \in \mathcal{D}, (o \neq o_i), \text{ are closer to } q_j \text{ than the sample point } o_{i,s}).$$

We assume that object o_i is the i^{th} object for which $rank_s(q_j)$ has reported at least one sample point. The same assumption is made for the sample points of an uncertain object (i.e., sample point $o_{i,s}$ is the s^{th}-closest sample point of object o_i according to q_j). These assumptions hold for the remainder of this paper.

Now, we show how to compute an entry $PT[k][i][s]$ of the probability table using an additional structure called sample table (ST). The sample table stores for each accessed object l separately the portion of samples already returned from $rank_s(q_j)$ denoted by $ST[l][1]$, whereas $ST[l][0]$ denotes the portion of the remaining not yet returned samples, i.e. $ST[l][0] = 1 - ST[l][1]$. Let ST be a sample table of size N (i.e. ST stores the information corresponding to all N objects of the database $\mathcal{D}$). Let $\sigma_k(i) \subseteq \{o \in \mathcal{D} | o \neq o_i\}$ denote the set, called *k-set* of o_i, containing exactly (k-1) objects. If we assume $k < N$, obviously $\binom{N}{k}$ different k-set permutations $\sigma_k(i)$ exist. For the computation of $PT[k][i][s]$, we have to consider the set S_k of all possible k-set permutations according to o_i. The probability that exactly (k-1) objects are closer to the query-sample point q_j than the sample point $o_{i,s}$, can be computed as follows:

$$PT[k][i][s] = \sum_{\sigma_k(i) \in S_k} \prod_{\substack{l = 1..N \\ l \neq i}} \begin{cases} ST[l][1] & \text{,if } o_l \in \sigma_k(i) \\ ST[l][0] & \text{,if } o_l \notin \sigma_k(i) \end{cases}$$

Let us assume that we actually process the sample point $o_{i,s}$. Since the object samples are processed in ascending order according to their distance to q_j, the sample table entry $ST[l][1]$ reflects the probability, that object o_l is closer to q_j than the sample point $o_{i,s}$. On the other hand, $ST[l][0]$ reflects the probability that $o_{i,s}$ is closer to q_j than o_l.

In the following, we show how the entries of the probability table can be computed by fetching iteratively the sample points from $rank_s(q_j)$. Thereby, we assume that all entries of the probability table are initially set to zero. Then the iterative ranking process $rank_s(q_j)$ which reports one sample point of an uncertain object in each iteration, is started. Each reported sample point $o_{i,s}$ is used to compute for all k ($1 \leq k \leq N$) the probability value that corresponds to the table entry $PT[k][i][s]$. After filling the (i-s)-column of the probability table, we proceed with the next sample point fetched from $rank_s(q_j)$ in the same way as we did with $o_{i,s}$. This procedure is repeated until all sample points are fetched from $rank_s(q_j)$.

```
ALGORITHM probability($\hat{ST}$,$MIN$,$MAX$,$k$) {
    result = 0;
    $N = MAX - MIN + 1$;
    IF $(k = 0)$ THEN result = $\prod_{i=MIN..MAX} \hat{ST}[i][0]$;
    ELSE IF $(k \geq N)$ THEN result = $\prod_{i=MIN..MAX} \hat{ST}[i][1]$;
    ELSE
        $MID = \lceil (MIN + MAX)/2 \rceil$;
        FOR $(i = 0..min(\lceil (MAX - MIN)/2 \rceil, k))$ DO
            left = probability($\hat{ST}, MIN, MID - 1, i$);
            right = probability($\hat{ST}, MID, MAX, (k - i)$);
            result = result + (left * right);
        END FOR
    END IF
    RETURN result;
}
```

Fig. 1. The sample point probability computation algorithm.

3.2 Accelerated Probability Computation

The computation of the probability table can be very costly in space and time. One reason is the size of the table that grows drastically with the number of objects and the number of samples for each object. The table size can be reduced as, in fact we need only one value per object and ranking position which aggregates the results over the object samples. Another problem is the very expensive computation of the probability table entries PT[k][i][s]. In the following, we propose methods that reach a considerable reduction of the overall query cost.

In fact, at a time we explicitly have to maintain table entries for those objects from which at least one sample point has been reported from $rank_s(q_j)$, whereas we can skip those from which we already fetched all sample points.

The computational bottleneck of our probabilistic ranking algorithm is the computation of each table entry. for each computation of $PT[k][i][s]$ we have to compute the probabilities according to $\binom{N}{k}$ different k-set permutations which have to be summed up to the final probability value. For example, if $N = 100$ and $k = 20$ we need to consider about $1.73 \cdot 10^{13}$ k-set permutations.

In the case of subsequently fetching samples belonging to the same object, the ranking probabilities according to this object doesn't change. Hence, obviously only one computation of the probability value is required. However, often the case where two adjacent sample points reported from the ranking belong to different objects occurs. For this case we suggest a divide and conquer method which is able to drastically reduce the number of k-set permutations to be computed. Instead of considering all k of N' permutations, we first split the k-set into two subsets of equal size. Then we only need to consider (k-i) of $\frac{N'}{2}$ permutations for $i = 1..k$ for the one subset, combined with the i of $\frac{N'}{2}$ permutations of the other subset. As a consequence, instead of considering

$\binom{N'}{k}$ k-set permutations, the number of k-set permutations to be considered can be reduced to

$$\sum_{i=0..k} \left(\binom{\frac{N'}{2}}{k-i} + \binom{\frac{N'}{2}}{i} \right).$$

The k-set split can be recursively repeated for each subset. The recursive decomposition of a subset, from which we have to compute k ($0 < k < N'$) out of N' permutations stops if $k \geq N'$. Otherwise, there exists only one permutation that can be immediately computed and reported to the calling function of the recursion. The algorithm for the computation of the sample point probability is depicted in Figure 1.

4 Experimental Evaluation

Due to space limitations, in this section we can only give a coarse summary of the experimental evaluation of our ranking methods. We applied our ranking methods on real world datasets as well as on artificial datasets. The artificial datasets which are used for the efficiency experiments contain 10 to 1000 3-dimensional uncertain objects. For the evaluation of the effectiveness of our methods we used three real-world datasets O_3, NSP_h and NSP_{frq}. The O_3 dataset is an environmental dataset consisting of 30 uncertain time series, each composing a set of measurements of O_3 concentration in the air measured within one month. The NSP datasets NSP_h and NSP_{frq} are chronobiologic datasets describing the cell activity of Neurospora[1] within sequences of day cycles. These datasets are used to investigate endogenous rhythms.

In the first experiments, we evaluated the quality of our probabilistic ranking query (*PRQ_MAC*) proposed in Section 2.3. We compare its quality with the quality of a non-probabilistic ranking (*MP*) which ranks the objects based on the distance between their mean positions. For these experiments, we used the three real-world datasets O_3, NSP_h and NSP_{frq}. The ranking quality is shown by the average precision over all recall values for each dataset. The avg. precisions according to the dataset O_3 are prec(*PRQ_MAC*)=0.65 and prec(*MP*)=0.63, to the dataset NSP_h are prec(*PRQ_MAC*)=0.43 and prec(*MP*)=0.35 and to the dataset NSP_{frq} are prec(*PRQ_MAC*)=0.70 and prec(*MP*)=0.60. Obviously, the *PRQ_MAC* approach outperforms the non-probabilistic ranking approach.

In the next experiment, we evaluate the performance of our probabilistic ranking acceleration strategies proposed in Section 3.2 w.r.t. query processing time. The results of the experiments showed that the strategies are able to reduce the query cost by several orders of magnitude. Interestingly, the recursive computation of the probability permutations alone (i.e. without other strategies) yields an speed up of up to two orders of magnitude compared to the other strategies.

5 Conclusions

In this paper, we proposed an approach that efficiently computes probabilistic ranking queries on uncertain objects represented by sets of sample points. In particular,

[1] Neurospora is the name of a fungal genus containing several distinct species. For further information see *The Neurospora Home Page*: http://www.fgsc.net/Neurospora/neurospora.html.

we proposed methods that are able to break down the high computational complexity required to compute for an object o the probability, that o has the ranking position k ($1 \leq$k$\leq N$) according to the distance to a query object q. We theoretically and experimentally showed that against straightforward solutions our approach is able to speed-up the query by factors of several orders of magnitude. In the future we plan to apply probabilistic ranking queries to improve data mining applications.

6 Acknowledgments

We would like to thank Jan Remis and Roselyn Santos from the Institute of Medical Psychology (IMP) at the Ludwig-Maximilians University of Munich for making the neurospora dataset NSP available to us. Furthermore, we would like to thank U. Böllmann and M. Meindl for providing us with the environmental dataset TEMP from the Bavarian State Office for Environmental Protection, Augsburg, Germany.

References

1. C. Böhm, A. Pryakhin, and M. Schubert. "Probabilistic Ranking Queries on Gaussians". In *Proc. of the 18th Int. Conf. on Scientific and Statistical Database Management (SSDBM'06)*, pages 169–178, 2006.
2. R. Cheng, D. Kalashnikov, and S. Prabhakar. "Evaluating Probabilistic Queries over Imprecise Data". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA)*, pages 551–562, 2003.
3. R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter. "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data". In *Proc. 30th Int. Conf. on Very Large Databases (VLDB'04), Toronto, Canada*, pages 876–887, 2004.
4. G. Hjaltason and H. Samet. "Ranking in Spatial Databases". In *Proc. 4th Int. Symposium on Large Spatial Databases, SSD'95, Portland, USA*, volume 951, pages 83–95, 1995.
5. H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. "Probabilistic Similarity Join on Uncertain Data". In *Proc. 11th Int. Conf. on Database Systems for Advanced Applications (DASFAA'06), Singapore, Singapore, pp. 295-309*, 2006, (Best paper).
6. H.-P. Kriegel, P. Kunath, and M. Renz. "Probabilistic Nearest-Neighbor Query on Uncertain Objects". In *Proc. 12th Int. Conf. on Database Systems for Advanced Applications (DASFAA'07), Bangkok, Thailand, pp. 337-348*, 2007.
7. H.-P. Kriegel, B. Seeger, R. Schneider, and N. Beckmann. "The R*-tree: An Efficient Access Method for Geographic Information System". In *Proc. Int. Conf. on Geographic Information Systems, Ottawa, Canada*, 1990.
8. Y. Tao, R. Cheng, X. Xiao, W. Ngai, B. Kao, and S. Prabhakar. "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions". In *Proc. 31th Int. Conf. on Very Large Data Bases (VLDB'05), Trondheim, Norway*, pages 922–933, 2005.