Probabilistic Skyline Queries

Christian Böhm
University of Munich
Munich, Germany
boehm@ifi.lmu.de

Frank Fiedler
University of Munich
Munich, Germany
fiedler@dbs.ifi.lmu.de

Annahita Oswald
University of Munich
Munich, Germany
oswald@dbs.ifi.lmu.de

Claudia Plant
Technische Universität München
Munich, Germany
plant@lrz.tum.de

Bianca Wackersreuther
University of Munich
Munich, Germany
wackersreuther@dbs.ifi.lmu.de

ABSTRACT

The ability to deal with uncertain information is becoming increasingly important for modern database applications. Whereas a conventional (certain) object is usually represented by a *vector* from a multidimensional feature space, an uncertain object is represented by a multivariate probability density function (*PDF*). This PDF can be defined either discretely (e.g. by a histogram) or continuously in parametric form (e.g. by a Gaussian Mixture Model). For a database of uncertain objects, the users expect similar data analysis techniques as for a conventional database of certain objects. An important analysis technique for certain objects is the skyline operator which finds maximal or minimal vectors with respect to any possible attribute weighting. In this paper, we propose the concept of probabilistic skylines, an extension of the skyline operator for uncertain objects. In addition, we propose efficient and effective methods for determining the probabilistic skyline of uncertain objects which are defined by a PDF in parametric form (e.g. a Gaussian function or a Gaussian Mixture Model). To further accelerate the search, we elaborate how the computation of the probabilistic skyline can be supported by an index structure for uncertain objects. An extensive experimental evaluation demonstrates both the effectiveness and the efficiency of our technique.

Categories and Subject Descriptors

H.3 [**Information Storage And Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Performance

Keywords

Gaussian Mixtrue Model, Skyline Query, Uncerain Data

1. INTRODUCTION

Skyline queries and probabilistic queries on uncertain data are two vital areas of current database research which recently have attracted increasing attention. The classical example of a skyline query is a user looking for cheap hotels which are close to the beach. Typically, it is unknown to the system how important the two conditions, *distance* and *price*, to the user actually are. If there is a conflict between these conditions (available are expensive hotels close to the beach and cheap hotels far away from the beach), it is unclear how the user's trade-off between these conditions would really look like. The system could ask the user to express his preferences by a weighting factor. However, such a weighting factor is often difficult to estimate, especially since the user typically has no information on the database content. Moreover, complex decision making on real world data usually involves several dimensions of interest. As an alternative, the skyline query returns *all* offers which may be of interest. This means, the skyline does not only contain the cheapest hotel and the hotel closest to the beach but also all hotels providing an outstanding combination of price and distance, which makes them more attractive to the user than any other hotel in the database. If one hotel A is both, closer to the beach *and* cheaper than hotel B, we say A *dominates* B (in symbols $A \prec B$), because A is better than B with respect to any possible weighting of the criteria *distance* and *price*. More generally, the *skyline* contains all objects which are not dominated by any other object in the data set. Such conventional skyline queries on exact data have attracted a huge volume of attention in the recent years, e.g. [3, 18, 9, 14, 11] where most papers focus on efficient algorithms for query processing. More recent approaches study skyline queries for special applications, e.g. time series data [10] and skyline queries from the perspective of a moving query object [20].

Uncertainty is a natural factor in many applications, which is not restricted to moving objects [8, 20]. The classical example of uncertain data modelled by probability density functions (PDFs) are sensor networks. In numerous applications, sensor networks are collecting data to monitor complex systems, for example the behavior of a certain species in its natural habitat [13], the environmental conditions in a region [12], the medical conditions of a patient [16], or the production of a chemical reactor [21]. A common characteristic of all sensor networks is that the individual sensors collect uncertain data. Every obtained piece of information is associated with some measurement error. Usually there is a trade-off between the accuracy, size and price of a sensor. Very precise sensors are typically more expensive and also larger in size. Wearable sensors for example need to be small and the measurement error is partly due to inaccurate placement and other individual conditions such as the electrical conductance of the skin. Based on extensive testing, manufacturers of sensors can usually provide close bounds on the measurement error.

Most complex decision processes based on real world data involve uncertainty which could be effectively supported by **skyline queries on uncertain data**: As a concrete application scenario of a sensor network with a large amount of sensors considers a chemical reactor producing *polyvinyl chloride* (PVC). Depending on the aiming application area of the PVC the reactor has to have a certain temperature. The temperature can either be controlled by heating up the reactor, or by adding some substances cooling down the whole production process. Due to the size of the reactor adding some chemical product in order to increase the temperature needs a certain time to take effect. Furthermore the pressure inside the reactor is also another critical and monitored indicator in quality assurance of the resulting product. Gas can be injected into the reactor or valves can be opened in order to control the pressure.

Producing PVC is only possible if the temperature of the reactor at any point may not drop below a certain value. Furthermore the pressure is not allowed to drop below some critical threshold. Of course the combination of low temperature and decreased pressure is also a critical combination. The reactor is equipped with hundreds of sensors, which measure different inaccurate phenomena of the production process. In general all outstanding combinations of the monitored sensors can be critical to the production phase. Those extraordinary values can be recognized as a skyline. Therefore an efficient skyline computation on uncertain data is necessary.

Uncertainty is not only restricted to sensor networks and moving objects. Due to space limitation, we can only mention a few examples. In privacy preserving data mining for example, uncertainty is often deliberately introduced to mask sensitive information. In all these applications, decision making can effectively be supported by our technique for processing probabilistic skyline queries.

The remainder of the paper is organized as follows: Section 2 introduces the related work on skylines and probabilistic query processing. In Section 3, the mathematical foundations for probabilistic skyline computation on uncertain data are derived, first for general PDFs and then for the important case of Gaussian PDFs and Gaussian Mixture Models. Section 4 gives the details on efficient processing of probabilistic skyline queries in presence of an index structure for uncertain objects in a filter-and-refinement architecture. Section 5 is dedicated to the algorithms for computing the skyline objects with and without index structures. Section 6 gives an extensive experimental evaluation and Section 7 concludes our paper.

2. RELATED WORK

Skyline Queries on Certain Data. The skyline operator has been introduced to the database community by Börzsönyi *et al.* [3]. Triggered by the practical relevance of skyline queries in many domains including e.g. customer segmentation, decision support and bio-medical applications, a huge volume of papers focus on efficient query processing, e.g. [6, 18, 9] and [11]. Most of these approaches can be classified into three major categories: nested-loop-based, divide-and-conquer-based and index-based methods. The Block-Nested-Loops algorithm (BNL) and the Extended Divide&Conquer algorithm (D&C) by Börzsönyi *et al.* [3] have been the fundamental approaches in the first two categories. More efficient approaches have recently been proposed, e.g. the Sort-Filter-Skyline algorithm (SFS) by Chomicki *et al.* [6] which improves BNL by pre-sorting the data set. Index-based approaches focus on pruning strategies based on approximations to avoid scanning the whole data set. The Bitmap and the Index technique by Tan *et al.* [18] fall into this category. The first method uses bit-vectors as approximations, whereas the second applies a transformation into one-dimensional space to exploit the B^+-tree for efficient retrieval. Another index-based algorithm is presented by Kossmann *et al.* [9], which is based on nearest neighbor search. This method does not compute the skyline in a batch-oriented way but progressively outputs results and thus allows the user to change preferences during runtime. Papadias *et al.* [14] developed a progressive algorithm BBS (branch-and-bound skyline), based on a nearest neighbor search technique supported by an R-tree. Only nodes that may contain skyline points are accessed, and each node is processed only once. Lin *et al.* [11] developed an efficient skyline computation method by determining the skyline for the most recent *N* elements in a data stream. All of these methods aim at the skyline computation of certain data.

Uncertain Data. In many applications, e.g. sensor networks [8] and moving objects, the features of the objects are not exactly known, but can be specified with some uncertainty, e.g. the current location of a moving object. A lot of work focuses on efficient indexing and query processing on uncertain data, e.g. [7, 17, 4, 5, 19, 2], just to mention a few. Typically, uncertain objects are modelled by probability density functions. Most query types on conventional databases, especially K-nearest neighbor and range queries have been extended to the probabilistic case. Cheng *et al.* [5] introduce R-tree-based index structures to support probabilistic threshold queries on uncertain objects. An uncertain object is represented by a so-called uncertainty interval which covers the unknown true appearance of the object and is associated with a PDF. All objects which are answers with a probability exceeding a threshold are returned to the user. Similarly, Tao *et al.* [19] represent uncertain objects by an uncertainty region associated with a PDF. The authors propose the U-tree, a hierarchical index structure to efficiently support probabilistic range queries. Hyper-rectangular core parts of the uncertainty regions can be associated with bounds for the appearance probability of an uncertain object, which allows for effective pruning. In [2], uncertain objects are represented by Gaussians, which allows for probabilistic identification queries. Given an uncertain query object, the identification query returns those uncertain objects which most likely represent the same object (e.g. for a database of facial images: Find the images most likely showing the same person as portrayed on the query image). The Gauss-tree is a hierarchical index structure on the parameter space of Gaussians: Gaussians with similar mean and variance are stored in a common MBR. The Gauss-tree, the U-tree and most other index structures for the efficient management of uncertain data are fully dynamic index structures which can efficiently handle insertions and deletions of uncertain objects. These index structures are very general and support many types of queries (e.g. identification queries). Therefore, it can be assumed that in a database system for uncertain objects, such an index structure is available, and it is not required to construct a particular index for each single query.

Probabilistic Skylines on Multi-Instance Data. In contrast to common query types, such as range or identification queries, probabilistic skyline queries have been nearly untouched so far. To the best of our knowledge, only the work of Pei *et al.* [15] goes into this direction. The authors propose a probabilistic skyline model for multi-instance data, where each object is part of the skyline with a certain probability. Each object consists of multiple instances. The skyline probability of an object is determined by counting the number of its instances which are not dominated by the instances of any other object in the database. This is the first approach dealing with uncertain data in the area of skyline queries, but it is not directly adaptable to objects which are initially represented by probability density functions. Approaches involving uncertain data represent uncertainty by a PDF because the true representation of an uncertain object is not available.

3. THE SKYLINE PROBABILITY OF UNCERTAIN OBJECTS

In this section, we introduce the basic definitions of probabilistic skylines. In the following we consider uncertain objects, each given by a PDF denoted by $f(\vec{x})$ or $g(\vec{x})$, and the complete database consists of a set of such PDFs, $DB = \{f_1(\vec{x}), ..., f_n(\vec{x})\}$ where $n = |DB|$ is the number of uncertain objects in DB. The vector $\vec{x} = (x_1, ..., x_d)^{\mathbf{T}} \in \mathbb{R}^d$ is a stochastic variable of the corresponding uncertain object that follows the given distribution function ($\vec{x} \sim f(\vec{x})$). Unlike in [15], $\vec{x}$ is not an instance given from the application.

3.1 The Skyline Probability of a General PDF

The original concept of skylines (for d-dimensional certain feature vectors) defines the notion of dominance in the following way: Object $\vec{x}$ dominates object $\vec{y}$ (in symbols $\vec{x} \prec \vec{y}$) if the following two conditions hold:

(1) $x_i \le y_i$ for all coordinates $i : 1 \le i \le d$, and
(2) $x_i < y_i$ for at least one coordinate $i : 1 \le i \le d$.

The generalization from non-probabilistic skylines on certain objects to probabilistic skylines on uncertain objects is given in the following definitions. We start with the generalization of the notion of the *dominance*:

DEFINITION 1 (PROBABILISTIC DOMINANCE). *Let $f(\vec{x}), g(\vec{y}) \in DB$ be uncertain objects. The probability that $f(\vec{x})$ dominates $g(\vec{y})$ corresponds to the probability that $\vec{x}$ generated from the probability density function $f(\vec{x})$ dominates $\vec{y}$ generated by $g(\vec{y})$. In symbols:*

$$P(\vec{x} \prec \vec{y}) = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} f(\vec{x}) \cdot g(\vec{y}) \cdot \begin{cases} 1 & \text{if } \vec{x} \prec \vec{y} \\ 0 & \text{otherwise} \end{cases} \mathbf{d}\vec{x}\, \mathbf{d}\vec{y}.$$

Instead of $P(\vec{x} \prec \vec{y})$ we will also often write $P(f(\vec{x}) \prec g(\vec{y}))$. The symbol $\int_{\mathbb{R}^d} f(\vec{x})\mathbf{d}\vec{x}$ stands for the d-dimensional infinite volume integral evaluated over the complete data space $\mathbb{R}^d$:

$$\int_{\mathbb{R}^d} f(\vec{x})\, \mathbf{d}\vec{x} = \int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} f((x_1, ..., x_d)^{\mathbf{T}})\, \mathbf{d}x_1\, ...\, \mathbf{d}x_d.$$

Similarly to the dominance, also the skyline property of an uncertain object in the database can only be defined in terms of a probability, the so-called *skyline probability* $P_S(f(\vec{x}))$:

DEFINITION 2 (SKYLINE PROBABILITY). *The skyline probability of an uncertain object $f(\vec{x}) \in DB$ corresponds to the probability with which $\vec{x}$ taken from the PDF $f(\vec{x})$ is not dominated by any of the $\vec{y}$ each of which is independently taken from one of the remaining uncertain objects stored in the database $g(\vec{y}) \in DB'$ where $DB' = DB \backslash \{f(\vec{x})\}$:*

$$P_S(f(\vec{x})) = P(\bigwedge_{g(\vec{y}) \in DB'} g(\vec{y}) \not\prec f(\vec{x})) =$$

$$= \int_{\mathbb{R}^d} f(\vec{x}) \prod_{g(\vec{y}) \in DB'} (1 - \int_{\mathbb{R}^d} g(\vec{y}) \cdot \begin{cases} 1 & \text{if } \vec{y} \prec \vec{x} \\ 0 & \text{otherwise} \end{cases} \mathbf{d}\vec{y})\, \mathbf{d}\vec{x}.$$

Most of the considered PDFs (such as Gaussians) have infinite support (i.e., the domain in which the probability density is different from 0). Hence in principle every object is in the skyline, though the skyline probability of most objects is very close to zero. In practice, usually only objects with a skyline probability significantly greater than zero are of interest. This is formalized by a user-defined threshold τ:

DEFINITION 3 (τ-SKYLINE). *Let $\tau \in [0..1]$ be a threshold. The τ-Skyline of a database of uncertain objects is the set of objects for which the following property holds:*

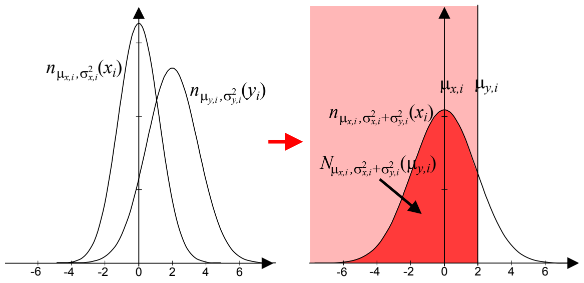


Figure 1: Transformation of the Probability.

$$S_\tau = \{f(\vec{x}) \in DB | P_S(f(\vec{x})) \ge \tau\}.$$

In the next section, we will derive the closed formulas for the dominance and skyline probability of uncertain objects, provided that the PDF is a normal distribution.

3.2 Gaussian Distribution Functions

The Gaussian distribution (or *normal* distribution) is by far the most important and most commonly used distribution function in statistics. Its extarordinary role is mainly due to the *central limit theorem*. The probability density function $f(\vec{x})$ corresponds to the well-known multivariate Gaussian function with mean $\vec{\mu}_x = (\mu_{x,1}, ..., \mu_{x,d})^{\mathbf{T}}$ and a diagonal (in principle, our work can also be extended to the non-diagonal case) covariance matrix $\Sigma_x = \text{diag}(\sigma^2_{x,1}, ..., \sigma^2_{x,d})$:

$$f(\vec{x}) = n_{\vec{\mu}_x, \Sigma_x}(\vec{x}) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_x}} \cdot \mathbf{e}^{-\frac{1}{2}(\vec{x}-\vec{\mu}_x)^{\mathbf{T}} \cdot \Sigma_x^{-1} \cdot (\vec{x}-\vec{\mu}_x)}.$$

A second object $g(\vec{y})$ can be defined applying separate parameters $\vec{\mu}_y$ and Σ_y. Since Σ_x and Σ_y are diagonal, the distribution functions are independent in the coordinates, and we can write them in the following way:

$$f(\vec{x}) = \prod_{1 \le i \le d} n_{\mu_{x,i},\sigma^2_{x,i}}(x_i) = \prod_{1 \le i \le d} \frac{1}{\sqrt{2\pi\sigma^2_{x,i}}} \mathbf{e}^{-\frac{(x_i - \mu_{x,i})^2}{\sigma^2_{x,i}}}.$$

The dominance probability can be rewritten as follows:

$$P(\vec{x} \prec \vec{y}) =$$

$$= \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \left(\prod_{1 \le i \le d} n_{\mu_{x,i},\sigma^2_{x,i}}(x_i) \cdot \prod_{1 \le i \le d} n_{\mu_{y,i},\sigma^2_{y,i}}(y_i) \right) \cdot$$

$$\cdot \begin{cases} 1 & \text{if } x_j \le y_j\ \forall j : 1 \le j \le d \\ 0 & \text{otherwise} \end{cases} \mathbf{d}\vec{x}\mathbf{d}\vec{y}.$$

Since all the factors in the products of this formula correspond to independent variables x_i and since the conditions in the case distinctions are independent, we can rearrange the terms in the following way and then exchange product and integral:

$$\begin{aligned} P(\vec{x} \prec \vec{y}) &= \prod_{1 \le i \le d} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} n_{\mu_{x,i},\sigma^2_{x,i}}(x_i) \cdot n_{\mu_{y,i},\sigma^2_{y,i}}(y_i) \\ &\quad \cdot \begin{cases} 1 & \text{if } x_i \le y_i \\ 0 & \text{otherwise} \end{cases} \mathbf{d}x_i \mathbf{d}y_i \\ &= \prod_{1 \le i \le d} P(x_i \le y_i). \end{aligned}$$

This means, we can do the whole determination of the dominance probability *independently* in each dimension (the two-fold integral) and then combine the individual dimension-wise probabilities to an overall probability (the product symbol in the above formula). Now we are going to derive the solution for each of the dimensions. In Figure 1, we see on the left side two different Gaussians. The task is to determine the probability with which $x_i \leq y_i$.

We substitute a new variable ϵ_i such that $y_i = x_i + \epsilon_i$. Then, the half space which we have to integrate (in which the case distinction yields 1) is exactly defined by $\epsilon_i \geq 0$. Then, our dominance probability corresponds to:

$$\begin{aligned} &P(x_i \leq y_i) = \\ &\quad = \int_0^{\infty} \int_{-\infty}^{+\infty} n_{\mu_{x,i},\sigma_{x,i}^2}(x_i) \cdot n_{\mu_{y,i},\sigma_{y,i}^2}(x_i + \epsilon_i)\, \mathbf{d}x_i\ \mathbf{d}\epsilon_i \\ &\quad = \int_0^{+\infty} \underbrace{\int_{-\infty}^{+\infty} n_{\mu_{x,i},\sigma_{x,i}^2}(x_i) \cdot n_{\mu_{y,i}-\epsilon_i,\sigma_{y,i}^2}(x_i)\, \mathbf{d}x_i}_{=\ n_{\mu_{x,i},\sigma_{x,i}^2+\sigma_{y,i}^2}(\mu_{y,i}-\epsilon_i) \quad (*)}\ \mathbf{d}\epsilon_i. \end{aligned}$$

where the formula (*) for the convolution product has already been proven in [2]. This can be simply rewritten using the cumulative distribution function, in the following form:

$$P(x_i \leq y_i) = N_{\mu_{x,i},\sigma_{x,i}^2+\sigma_{y,i}^2}(\mu_{y,i}).$$

We can now conclude that integrating the product of the two Gaussians exactly corresponds to integrating *one* Gaussian with the added variances of the two. As Figure 1 demonstrates, the original problem on the left side has been transformed into an easy-to-solve problem on the right side, where only one Gaussian needs to be integrated. The integral is marked in color.

3.3 Gaussian Mixture Models

Now, we want to derive an analogous formula, provided that the probability density functions $f_x(\vec{x})$ and $f_y(\vec{y})$ are given by Gaussian Mixture Models rather than simple Gaussian functions. Gaussian Mixture models provide a highly flexible and accurate representation of uncertain objects. Gaussian Mixture Models are for example suitable to represent sensor data with non-Gaussian error distributions [1]. A Gaussian Mixture Model M_x is given as a set of triplets

$$M_x = \{(w_x, \vec{\mu}_x, \Sigma_x)_{(1)}, ..., (w_x, \vec{\mu}_x, \Sigma_x)_{(k)}\}$$

where $k = |M_x|$ is the number of components. Each component is defined by its weight w_x and its means $\vec{\mu}_x$ and its diagonal variance matrix Σ_x. The sum of the weights equals one:

$$\sum_{(w_x,\vec{\mu}_x,\Sigma_x)\in M_x} w_x = 1.$$

The probability density function of the GMM is given by:

$$\begin{aligned} f_x(\vec{x}) &= \sum_{(w_x,\vec{\mu}_x,\Sigma_x)\in M_x} w_x \cdot n_{\vec{\mu}_x,\Sigma_x}(\vec{x}) \\ &= \sum_{(w_x,\vec{\mu}_x,\Sigma_x)\in M_x} w_x \cdot \prod_{1\leq i\leq d} n_{\mu_{x,i},\sigma_{x,i}^2}(x_i). \end{aligned}$$

The GMM M_y and its probability density function is defined analogously. The number of components $l = |M_y|$ may differ from that of M_x.

The dominance probability can now be written as:

$$\begin{aligned} &P(\vec{x} \prec \vec{y}) = \\ &\quad = \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \left(\sum_{(w_x,\vec{\mu}_x,\Sigma_x)\in M_x} w_x \cdot \prod_{1\leq i\leq d} n_{\mu_{x,i},\sigma_{x,i}^2}(x_i) \right) \cdot \\ &\quad \cdot \left(\sum_{(w_y,\vec{\mu}_y,\Sigma_y)\in M_y} w_y \cdot \prod_{1\leq i\leq d} n_{\mu_{y,i},\sigma_{y,i}^2}(y_i) \right) \cdot \\ &\quad \cdot \begin{cases} 1 & \text{if } x_j \leq y_j\ \forall j : 1 \leq j \leq d \\ 0 & \text{otherwise} \end{cases} \mathbf{d}\vec{x}\mathbf{d}\vec{y}. \end{aligned}$$

We can apply the distributive law and then exchange summation and integration, and include our results obtained in Section 3.2 to obtain:

$$= \sum_{(w_x,\vec{\mu}_x,\Sigma_x)\in M_x} \sum_{(w_y,\vec{\mu}_y,\Sigma_y)\in M_y} w_x w_y \cdot \prod_{1\leq i\leq d} N_{\mu_{x,i},\sigma_{x,i}^2+\sigma_{y,i}^2}(\mu_{y,i}).$$

4. CONSERVATIVE ESTIMATION OF THE DOMINANCE PROBABILITY

When uncertain objects are stored in an index structure such as the U-tree [19] or the Gauss-tree [2], then it is an interesting question how to accelerate the search for the skyline by exploiting this index structure. If, for instance, a few objects belonging to the skyline are already known (or at least a few good candidates for skyline objects) then these objects may dominate not only other single database objects but also complete branches (subtrees) of the index. On the other hand, to determine the exact probability of an uncertain object $f(\vec{x})$ to belong to the skyline, theoretically we have to examine all objects $g(\vec{y}) \in DB$ which have a probability $P(\vec{y} \succ \vec{x}) > 0$ of dominating the object. However, many objects $g(\vec{y})$ may have a probability to dominate $f(\vec{x})$ which is close to zero, and the same may be true for *all* objects $g(\vec{y})$ in a complete *subtree.* This leads us to the question, how to estimate the highest and lowest possible probability with which an uncertain object dominates an arbitrary object stored in a subtree, and, inversely, what is the highest and lowest probability that an arbitrary object stored in a subtree dominates another uncertain object.

Note that index structures such as the U-tree or the Gauss-tree are very general indexing methods efficiently supporting different kinds of queries (such as probabilistic identification queries) and efficiently supporting insertion of new uncertain objects and deletion of uncertain objects. Therefore, it can be assumed that such an index is available in a database system for uncertain objects. The details how insertions and deletions can be processed and how the tree is maintained (e.g. to keep balance) for the U-tree can be found in [19], for the Gauss-tree in [2].

An important principle for indexing probability density functions is the principle of *parameter space indexing* where the parameters (in the case of Gaussian PDFs $(\mu_1, \sigma_1, ..., \mu_d, \sigma_d)$) are stored like points from a $(2 \cdot d)$-dimensional data space rather than spatially extended objects. For Gaussian Mixture Models, each uncertain object is represented by a set of such $(2 \cdot d)$-dimensional points. The most prominent example for such a *parameter space indexing method* is the Gauss-tree [2]. Like in R-trees and similar tree structures for high-dimensional data, objects which are close to each other in the $(2 \cdot d)$-dimensional data space are grouped together into a common subtree, and the whole subtree is represented by an axis-parallel minimum bounding rectangle of the stored points. Before accessing a node, query processing algorithms have to check according to the minimum bounding rectangle, whether or not the corresponding subtree is able to contain objects to answer the query.

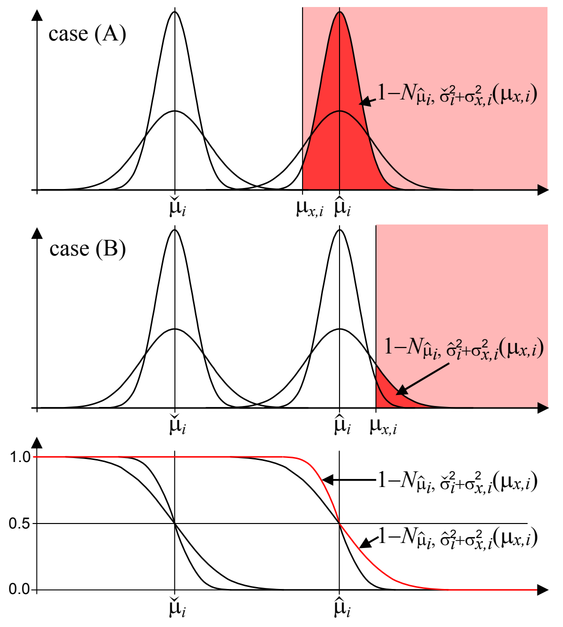


Figure 2: Maximum Probability that Object $n_{\mu_{x,i},\sigma^2_{x,i}}$ Dominates M_y.

In our context, we need to determine the maximum and minimum probability of an object stored in a tree ($\equiv$ contained in the rectangle) to *dominate* a given uncertain object, and to *be dominated* by a given uncertain object, respectively.

In the following sections we restrict the formulas to the simple Gaussian functions only for better readability, the exact same formulas hold for the Mixture Model as well, only the weighting factor is missing.

4.1 Uncertain Object Dominates MBR

To clearly define the problem, we are given an uncertain object as a probability density function

$$f(\vec{x}) = \prod_{1\le i\le d} n_{\mu_{x,i},\sigma_{x,i}}(x_i)$$

and a minimum bounding rectangle

$$MBR = (\check{\mu}_1, \hat{\mu}_1, \check{\sigma}_1, \hat{\sigma}_1, ..., \check{\mu}_d, \hat{\mu}_d, \check{\sigma}_d, \hat{\sigma}_d).$$

which defines limits for the parameters of other probability density functions $g(\vec{y})$ which may be stored in the subtree. Therefore, $g(\vec{y})$ may be an arbitrary Gaussian function

$$g(\vec{y}) = \prod_{1\le i\le d} n_{\mu_{y,i},\sigma_{y,i}}(y_i)$$

where the parameters fulfill the constraints

$$\begin{array}{lll} \check{\mu}_1 \le \mu_{y,1} \le \hat{\mu}_1, & ..., & \check{\mu}_d \le \mu_{y,d} \le \hat{\mu}_d \\ \check{\sigma}_1 \le \sigma_{y,1} \le \hat{\sigma}_1, & ..., & \check{\sigma}_d \le \sigma_{y,d} \le \hat{\sigma}_d. \end{array}$$

We call the functions $g(\vec{y})$ fulfilling the above constraints the *allowed* functions in the subtree. Note that a function does not have to be stored in the subtree to be allowed. The allowed functions are rather those uncertain objects which potentially *could* be stored in the subtree.

We want to determine among all allowed functions those which are dominated by $f(\vec{x})$ with maximum and minimum probability, respectively. In symbols:

$$\hat{P}(f(\vec{x}) \prec MBR) = \max_{\substack{\check{\mu}_i \le \mu_{y,i} \le \hat{\mu} \\ \check{\sigma}_i \le \sigma_{y,i} \le \hat{\sigma}}} \left\{ \prod_{1\le j\le d} N_{\mu_{x,j},\sigma^2_{x,j}+\sigma^2_{y,j}}(\mu_{y,j}) \right\},$$

($\forall i : 1 \le i \le d$) is the maximum dominance probability, and

$$\check{P}(f(\vec{x}) \prec MBR) = \min_{\substack{\check{\mu}_i \le \mu_{y,i} \le \hat{\mu} \\ \check{\sigma}_i \le \sigma_{y,i} \le \hat{\sigma}}} \left\{ \prod_{1\le j\le d} N_{\mu_{x,j},\sigma^2_{x,j}+\sigma^2_{y,j}}(\mu_{y,j}) \right\},$$

($\forall i : 1 \le i \le d$) is the minimum dominance probability. Our first observation is that here the maximization (minimization) can be done individually in each dimension because the terms are again independent in the dimensions. We can add the variance of $f(\vec{x})$ to that of $g(\vec{y})$:

$$\hat{P}(f(\vec{x}) \prec MBR) =$$

$$= \prod_{1\le i\le d} \max_{\substack{\check{\mu}_i \le \mu_{y,i} \le \hat{\mu} \\ \check{\sigma}_i \le \sigma_{y,i} \le \hat{\sigma}}} \left\{ N_{\mu_{x,j},\sigma^2_{x,j}+\sigma^2_{y,j}}(\mu_{y,j}) \right\}$$

(and analogously for the minimum dominance probability). It is convenient to exchange $\mu_{x,j}$ and $\mu_{y,j}$ in the above equation by following the rule $N_{\mu,\sigma^2}(x) = 1 - N_{x,\sigma^2}(\mu)$:

$$\hat{P}(f(\vec{x}) \prec MBR) =$$

$$= \prod_{1\le i\le d} \max_{\substack{\check{\mu}_i \le \mu_{y,i} \le \hat{\mu} \\ \check{\sigma}_i \le \sigma_{y,i} \le \hat{\sigma}}} \left\{ 1 - N_{\mu_{y,j},\sigma^2_{x,j}+\sigma^2_{y,j}}(\mu_{x,j}) \right\}.$$

Now, we can draw the four extreme curves of the MBR with the added variance of $f(\vec{x})$, as depicted in the top and middle diagram of Figure 2. We have also marked for two possible positions of $\mu_{x,i}$, in the top diagram a position which is less than $\hat{\mu}_i$ and in the middle diagram a position greater than $\hat{\mu}_i$. In the next paragraph we will see that these are two possibilities in a case distinction. In both diagrams, we have marked the dominated area, i.e. right from $\mu_{x,i}$. The probability corresponds to the area under the Gaussian, starting from $\mu_{x,i}$ until $+\infty$ which exactly corresponds to $1 - N...(\mu_{x,i})$. We will also show in the next paragraph how the parameters of N must be set in order to maximize (minimize) the dominance probability.

No matter at what position $\mu_{x,i}$ in Figure 2 is, the maximally dominated function is always one of the rightmost possible functions, i.e. $\mu_{y,i} = \hat{\mu}_i$. The reason is that if we want to determine that curve, which has most of its area right from the point $\mu_{x,i}$, then we have to shift the curve as far as possible to the right side. This intuition is also confirmed by the bottom diagram in Figure 2 which depicts for each of the extreme functions of the MBR the corresponding function $1 - N...(\mu_{x,i})$ which gives for each position of $\mu_{x,i}$ the area under the Gaussian right from $\mu_{x,i}$: The two functions centered at $\hat{\mu}_i$ are everywhere greater than the corresponding functions centered at $\check{\mu}_i$ and all other allowed functions (with same variance). A little bit more complex is the determination of the variance $\sigma^2_{y,i}$ of the function $g(\vec{y})$ maximizing the dominance probability. We have to distinguish the two cases where $\mu_{x,i}$ is left (case A) or right (case B) from $\hat{\mu}_i$.

Case (A) is depicted in the top diagram of Figure 2. In this case, we can see that the maximizing variance is defined by the most narrow (and highest) function with $\sigma_{y,i} = \check{\sigma}_i$, because the integral of a Gaussian is monotonically increasing (positive derivative) with increasing variance σ^2 if (and only if) the upper integration boundary is less than the mean, and monotonically decreasing (negative derivative) if (and only if) the upper integration boundary is greater than the mean. If the integration boundary is equal to the means, then the integral is constantly 0.5 no matter how large the variance is (and its derivative is equal to zero). Formally:

$$\frac{\mathbf{d}}{\mathbf{d}\sigma} N_{\mu,\sigma^2}(x) \begin{cases} < 0 & \text{if } x < \mu \\ = 0 & \text{if } x = \mu \\ > 0 & \text{if } x > \mu. \end{cases}$$

Therefore, in case (A), $1 - N...(\mu_{x,i})$ is maximal for the smallest possible variance (i.e. the most narrow Gaussian), in this case $(\check{\sigma}_i^2 + \sigma_{x,i}^2)$. This intuition is again confirmed by the bottom diagram of Figure 2 where the maximum among all allowed functions is $1 - N_{\hat{\mu}_i, \check{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i})$ if $\mu_{x,i} \leq \hat{\mu}_i$.

Case (B) is depicted in the middle diagram of Figure 2. In this case, we see that $\sigma_{y,i} = \hat{\sigma}_i$ defines the maximum because the marked area becomes larger the more σ_{y_i} is increased. For the case $\mu_{x,i} = \hat{\mu}_i$, we would mark 50 percent of the curve, no matter how large σ_{y_i} is. Summarizing, we obtain the following formula for an upper bound estimation of the maximum dominance probability:
$\hat{P}(f(\vec{x}) \prec MBR) =$

$$= \prod_{1 \leq i \leq d} \begin{cases} 1 - N_{\hat{\mu}_i, \check{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{if } \mu_{x,i} \leq \hat{\mu}_i \\ 1 - N_{\hat{\mu}_i, \hat{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{otherwise.} \end{cases}$$

We have $1 - N...(\mu_{x,i})$ because here we have to integrate from $\mu_{x,i}$ to $+\infty$ rather than the integration boundaries from Section 3.2 (from $-\infty$ to $\mu_{x,i}$). The overall function $\hat{P}(f(\vec{x}) \prec MBR)$ is depicted in the bottom diagram of Figure 2. Here we can see the four integral functions corresponding to the four extreme Gaussians representing the corner points of the MBR. The maximum (which changes at $\hat{\mu}_i$ from $1 - N_{\hat{\mu}_i, \check{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i})$ to $1 - N_{\hat{\mu}_i, \hat{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i})$) is marked in color. Analogously, the minimum dominance probability is given by:
$\check{P}(f(\vec{x}) \prec MBR) =$

$$= \prod_{1 \leq i \leq d} \begin{cases} 1 - N_{\check{\mu}_i, \hat{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{if } \mu_{x,i} \leq \check{\mu}_i \\ 1 - N_{\check{\mu}_i, \check{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{otherwise.} \end{cases}$$

In the bottom diagram of Figure 2, the minimum function is not particularly marked but it can also be recognized that minimal are those functions which are centered by $\check{\mu}_i$ and that the case distinction of the above formula is also correct.

4.2 MBR Dominates a Given Uncertain Object

Inversely, we are also interested in the question what is the maximum and minimum probability with which an arbitrary function from the MBR may prune a given uncertain object. This information will later be used to avoid unnecessary node refinements when confirming that the uncertain object is indeed a skyline object. The derivation of the following equations is analogous to those in Section 4.1:
$\hat{P}(MBR \prec f(\vec{x})) =$

$$= \max_{\substack{\check{\mu}_i \leq \mu_{y,i} \leq \hat{\mu} \\ \check{\sigma}_i \leq \sigma_{y,i} \leq \hat{\sigma}}} \left\{ \prod_{1 \leq j \leq d} N_{\mu_{y,j}, \sigma_{x,j}^2 + \sigma_{y,j}^2}(\mu_{x,j}) \right\}$$
$$= \prod_{1 \leq i \leq d} \begin{cases} N_{\check{\mu}_i, \hat{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{if } \mu_{x,i} \leq \check{\mu}_i \\ N_{\check{\mu}_i, \check{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{otherwise.} \end{cases}$$

$\check{P}(MBR \prec f(\vec{x})) =$

$$= \min_{\substack{\check{\mu}_i \leq \mu_{y,i} \leq \hat{\mu} \\ \check{\sigma}_i \leq \sigma_{y,i} \leq \hat{\sigma}}} \left\{ \prod_{1 \leq j \leq d} N_{\mu_{y,j}, \sigma_{x,j}^2 + \sigma_{y,j}^2}(\mu_{x,j}) \right\}$$
$$= \prod_{1 \leq i \leq d} \begin{cases} N_{\hat{\mu}_i, \check{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{if } \mu_{x,i} \leq \hat{\mu}_i \\ N_{\hat{\mu}_i, \hat{\sigma}_i^2 + \sigma_{x,i}^2}(\mu_{x,i}) & \text{otherwise.} \end{cases}$$

4.3 MBR Dominates Other MBR

We can define the dominance probability also at the level of two minimum bounding rectangles MBR_x and MBR_y and determine the maximum and minimum probability with which an arbitrary function stored in MBR_x dominates an arbitrary curve stored in MBR_y, respectively.
$\hat{P}(MBR_x \prec MBR_y) =$

$$= \max_{\substack{\check{\mu}_{x,i} \leq \mu_{x,i} \leq \hat{\mu}_{x,i} \\ \check{\sigma}_{x,i} \leq \sigma_{x,i} \leq \hat{\sigma}_{x,i} \\ \check{\mu}_{y,i} \leq \mu_{y,i} \leq \hat{\mu}_{y,i} \\ \check{\sigma}_{y,i} \leq \sigma_{y,i} \leq \hat{\sigma}_{y,i}}} \left\{ \prod_{1 \leq j \leq d} N_{\mu_{y,j}, \sigma_{x,j}^2 + \sigma_{y,j}^2}(\mu_{x,j}) \right\}$$
$$= \prod_{1 \leq i \leq d} \begin{cases} 1 - N_{\hat{\mu}_{y,i}, \check{\sigma}_{y,i}^2 + \check{\sigma}_{x,i}^2}(\check{\mu}_{x,i}) & \text{if } \check{\mu}_{x,i} \leq \hat{\mu}_{y,i} \\ 1 - N_{\hat{\mu}_{y,i}, \hat{\sigma}_{y,i}^2 + \hat{\sigma}_{x,i}^2}(\check{\mu}_{x,i}) & \text{otherwise.} \end{cases}$$

$\check{P}(MBR_x \prec MBR_y) =$

$$= \prod_{1 \leq i \leq d} \begin{cases} 1 - N_{\check{\mu}_{y,i}, \hat{\sigma}_{y,i}^2 + \hat{\sigma}_{x,i}^2}(\hat{\mu}_{x,i}) & \text{if } \hat{\mu}_{x,i} \leq \check{\mu}_{y,i} \\ 1 - N_{\check{\mu}_{y,i}, \check{\sigma}_{y,i}^2 + \check{\sigma}_{x,i}^2}(\hat{\mu}_{x,i}) & \text{otherwise.} \end{cases}$$

4.4 Bounding the Skyline Probability

Until now, we have only given the details how the dominance probability can be determined for Gaussian PDFs but not for the skyline probability. An obvious idea to determine the skyline probability of an uncertain object $f(\vec{x})$ would be directly to use the results of Section 4.2 to 4.3 and to multiply the probabilities of all other uncertain objects $g(\vec{y}) \in DB \backslash \{f(\vec{x})\}$, not to dominate $f(\vec{x})$. However, this approach requires the independence of the dominance property. As we will show in this section, the probability that one object $g(\vec{y})$ dominantes $f(\vec{x})$ and the probability that another object $h(\vec{z})$ dominates $f(\vec{x})$ may be positively (but not negatively) correlated. As we will see, negative correlations are excluded due to the transitivity of the dominance. The consequence is, that we could under-estimate (but not over-estimate) the skyline probability by ignoring this correlation.

LEMMA 1 (POSITIVE CORRELATION OF DOMINANCE). *Let $f(\vec{x}), g(\vec{y})$, and $h(\vec{z}) \in DB$ be uncertain objects. The conditional probability that $f(\vec{x})$ is dominanted by $h(\vec{z})$ under the condition that $f(\vec{x})$ is dominated by $g(\vec{y})$ is greater or equal to the unconditional probability that $f(\vec{x})$ is dominated by $h(\vec{z})$. Likewise, the conditional probability that $f(\vec{x})$ is not dominated by $h(\vec{z})$ under the condition that $f(\vec{x})$ is not dominated by $g(\vec{y})$ is greater or equal*

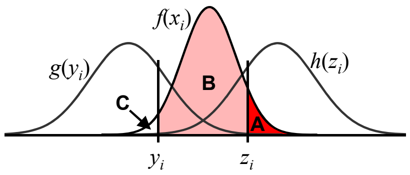


Figure 3: Correlation of Dominance.

to the unconditional probability:

$$\begin{aligned} P(h(\vec{z}) \prec f(\vec{x}) | g(\vec{y}) \prec f(\vec{x})) &\geq P(h(\vec{z}) \prec f(\vec{x})), \\ P(h(\vec{z}) \not\prec f(\vec{x}) | g(\vec{y}) \not\prec f(\vec{x})) &\geq P(h(\vec{z}) \not\prec f(\vec{x})). \end{aligned}$$

PROOF. Consider Figure 3 where we have two stochastic variables $y_i \sim g(y_i)$ and $z_i \sim h(z_i)$ Those values of x_i which are dominated by z_i, are in the area marked by $A = \int_{z_i}^{\infty} f(x_i)\mathbf{d}x_i$. The values which are dominated by y_i are in the areas $A \cup B$ with $B = \int_{y_i}^{z_i} f(x_i)\mathbf{d}x_i$. The sum of areas $A + B + C = 1$ (the whole Gaussian PDF). The unconditional probability with which x_i is dominated by z_i is:

$$P(z_i \leq x_i) = \frac{A}{A+B+C} = A.$$

In contrast, if we know that x_i is dominated by y_i (conditional probability), then the event space only consists of the areas $A \cup B$, and, therefore, the conditional probability is:

$$P(z_i \leq x_i | y_i \leq x_i) = \frac{A}{A+B} \geq P(z_i \leq x_i).$$

If the roles of y_i and z_i are exchanged, then the conditional probability even becomes 1 because all instances which are dominated by z_i are also dominated by y_i. But also in this case the statement of the lemma holds (trivially). It also holds in the multivariate case ($h(\vec{z}) \prec f(\vec{x})$), because in the conditional probability, the area of negative events is always less or equal compared to the unconditional probability whereas the positive events are unchanged. The same applies for the conditional probability for $h(\vec{z}) \not\prec f(\vec{x})$. □

This lemma provides us some possibilities for the estimation of upper and lower boundaries of the skyline probability, mostly due to the *product rule of probabilites* ($P(A \wedge B) = P(A) \cdot P(B|A)$ for arbitrary, not necessarily independent stochastic events A and B which is related to *Bayes' theorem*) which allows the decomposition of the formula for the skyline probability. Let $Q \subseteq DB'$ (where again $DB' = DB \backslash \{f(\vec{x})\}$) be a suitable subset of the database, (e.g. those objects that have the largest probabilities to dominate $f(\vec{x})$). Then according to the product rule of probabilities we have:

$$P_S(f(\vec{x})) = P(\bigwedge_{g(\vec{y}) \in DB'} g(\vec{y}) \not\prec f(\vec{x})) =$$

$$= P(\bigwedge_{g(\vec{y}) \in Q} g(\vec{y}) \not\prec f(\vec{x})) \cdot P(\bigwedge_{h(\vec{z}) \in DB' \backslash Q} h(\vec{z}) \not\prec f(\vec{x}) | \bigwedge_{g(\vec{y}) \in Q} g(\vec{y}) \not\prec f(\vec{x}))$$

of which we can get an upper bound by completely cancelling the conditional term (which is ≤ 1), and a lower bound by replacing it by the unconditional term (which is a lower bound due to Lemma 1):

$$\begin{aligned} P_S(f(\vec{x})) &\leq \hat{P}_S^Q(f(\vec{x})) = P(\bigwedge_{g(\vec{y}) \in Q} g(\vec{y}) \not\prec f(\vec{x})), \\ P_S(f(\vec{x})) &\geq \check{P}_S^Q(f(\vec{x})) \\ &= P(\bigwedge_{g(\vec{y}) \in Q} g(\vec{y}) \not\prec f(\vec{x})) \cdot \prod_{h(\vec{z}) \in DB' \backslash Q} P(h(\vec{z}) \not\prec f(\vec{x})). \end{aligned}$$

We call $\hat{P}_S^Q(f(\vec{x}))$ the *partial skyline estimation with respect to* Q and the term

$$P_{IASP}^Q(f(\vec{x})) = \prod_{h(\vec{z}) \in Q} P(h(\vec{z}) \not\prec f(\vec{x}))$$

the *independence assumption skyline estimation with respect to* Q. Note that, if Q corresponds to the complete database DB', then

$$\check{P}_S^{DB'}(f(\vec{x})) = P_S(f(\vec{x})) = \hat{P}_S^{DB'}(f(\vec{x})),$$

and the independence assumption estimation (which can be computed analytically in a very inexpensive way) can e.g. be used to rank the objects of the database (cf. Section 3.2). For the partial skyline estimation $\hat{P}_S^{DB'}(f(\vec{x}))$, we can use the following partial Montecarlo integration method: Let $X = \{\vec{x}_1, ..., \vec{x}_s\}$ be a set (with a number $s = |X|$ of elements) of values i.i.d. generated from $f(\vec{x})$. Then:

$$\hat{P}_S^Q(f(\vec{x})) \approx P_{MC}^Q(f(\vec{x})) = \frac{1}{s} \sum_{\vec{x} \in X} \prod_{n_{\vec{\mu},\Sigma}(\vec{y}) \in Q} (1 - N_{\vec{\mu},\Sigma}(\vec{x})).$$

This Montecarlo integration is not very expensive either since (1) it generates instances $\vec{x}$ only for the single PDF $f(\vec{x})$, and not for the (many) other functions $g(\vec{y}) \in Q$ (i.e. our method requires $O(s)$ computations), and (2) this integration method is updatable whenever Q changes (e.g. when a new uncertain object is inserted into Q). (e.g. when a new uncertain object is inserted into Q). Our method has no demand to store generated instances permanently.

5. ALGORITHMS

In the following sections we describe algorithms for efficient computation of the τ-skyline.

5.1 Baseline Algorithm for Skyline Computation

The baseline algorithm for determining the τ-skyline iterates over the number n of objects in the database, and determines the skyline probability of each object by generating s instances. The skyline probability is then computed according to the formula for $P_{MC}^{DB'}(f(\vec{x}))$, where $DB' = DB \backslash \{f(\vec{x})\}$ as described in Section 4.4. An object $f(\vec{x})$ belongs to the τ-skyline if

$$P_{MC}^{DB'}(f(\vec{x})) \geq \tau$$

holds. Since each database object has to be compared to each other, the runtime complexity of the baseline algorithm is $O(s \cdot n^2)$, with s being the sample rate. The baseline algorithm has only a linear runtime complexity with respect to the number of instances. But no filtering steps from Section 3 are applied to accelerate the skyline operator.

5.2 Priority Algorithm for Efficient τ-Skyline Computation

In order to avoid the computation of the exact skyline probability for each object, we introduce our priority algorithm. The main idea of the priority algorithm is to use the equations derived in Section 4.4 to exclude as many objects as possible from further consideration. Compared to the numerically determined skyline probability $P_{MC}^{DB'}(f(\vec{x}))$, the approximation $P_{IASP}^{DB'}(f(\vec{x}))$ can be computed extremly fast even on large databases. This fact is used to determine for object $f(\vec{x})$ the approximation $P_{IASP}^{DB'}(f(\vec{x}))$ by comparing the object $f(\vec{x})$ with all other objects $g(\vec{y})$ of the database. The resulting dominance probability $1 - P(g(\vec{y}) \prec f(\vec{x}))$ from this comparison step is stored together with a reference to $g(\vec{y})$ in a priority queue in descending order. In the following *iteration step*, in order to decide whether $f(\vec{x})$ belongs to the τ-skyline, we use the priority queue and get one object after the other out of the queue. For each dequeued object $h(\vec{z})$, we update the skyline probability $P_{MC}^{SF}(f(\vec{x}))$ of the set SF of all *S*o *F*ar processed objects of the priority queue. When processing object $h(\vec{z})$ we update $P_{IASP}^{DB'}(f(\vec{x}))$ by cancelling out the dominance probability which is stored in the priority queue. This results in the probability $P_{IASP}^{DB' \backslash SF}(f(\vec{x}))$,

```
01 algorithm PrioritySkyline(threshold $\tau$)
02   Set resultSet := {};
03   for all $f(\vec{x}) \in DB$ do:
04     PriorityQueue pq = new PriorityQueue(descending);
05     for all $g(\vec{y}) \in DB'$ do:
06       update $P_{IASP}^{DB'}(f(\vec{x}))$;
07       pq.insert($g(\vec{y}), (1 - P(g(\vec{y}) \prec f(\vec{x})))$ );
08     end for;
09     Set $SF$ :={};
10     while(pq.isNotEmpty()) do:
11       $h(\vec{z})$ := pq.removeFirst();
12       $SF := SF \cup \{h(\vec{z})\}$
13       update $P_{IASP}^{DB' \setminus SF}(f(\vec{x}))$
14       update $P_{MC}^{SF}(f(\vec{x}))$
15       if ($P_{MC}^{SF}(f(\vec{x})) < \tau$)
16         break while; //$f(\vec{x})$ does NOT belong to skyline
17       if ($P_{MC}^{SF}(f(\vec{x})) \cdot P_{IASP}^{DB' \setminus SF}(f(\vec{x})) \geq \tau$)
18         resultSet.add($f(\vec{x})$); // A belongs to skyline
19         break while;
19     end while;
20   end for;
21   return resultSet;
```

Figure 4: Pseudocode of the Priority Algorithm for Skyline Construction.

representing the maximal possible influence of the remaining objects in the priority queue on object $f(\vec{x})$. This means, the more objects $h(\vec{z})$ are processed, the smaller $P_{MC}^{SF}(f(\vec{x}))$ gets. Object $f(\vec{x})$ can be returned if one of the following conditions are fulfilled:

- if $P_{MC}^{SF}(f(\vec{x})) < \tau$ then $f(\vec{x})$ can definitely be excluded from the τ-skyline.

- if $P_{MC}^{SF}(f(\vec{x}))) \cdot P_{IASP}^{DB' \setminus SF}(f(\vec{x})) \geq \tau$ then $f(\vec{x})$ definitely belongs to the τ-skyline.

If we are only interested in objects belonging to the τ-skyline, we can stop the computation as soon as we know that $f(\vec{x})$ belongs to the skyline. But then the exact skyline probability value of $f(\vec{x})$ can not be returned.

We can typically exclude object $f(\vec{x})$ from the skyline very early, i.e. after comparing it with only a few other objects $h(\vec{z})$ of the priority queue, since the queue is sorted in descending order according to $P_{IADP}(h(\vec{z}) \prec f(\vec{x}))$. The objects $h(\vec{z})$ which probably have the highest influence on the skyline probability of $f(\vec{x})$ are processed first.

The pseudocode for the priority τ-skyline algorithm is depicted in Figure 4. Here, the computation of $P_{IASP}^{DB'}(f(\vec{x}))$ in line 06 refers to the formula in Section 4.4. The worst case runtime complexity of the priority algorithm is:

$$O\left(n^2 + s \cdot n \cdot |SF|\right),$$

with s being the sample rate, n the number of objects in the database, and $|SF|$ being the maximum number of objects considered in the set SF. In Section 6 we will demonstrate the superiority of the priority algorithm over the baseline approach. The main disadvantage of the priority algorithm is its I/O costs.

5.3 Indexed Algorithm for Efficient τ Skyline Computation

When objects are stored in a database a common indexing technique uses minimal bounding rectangles (MBR). The indexed approach uses the MBR technique as described in section 4. By using the index the τ-skyline can prune several MBRs without looking at the exact objects.

```
01 algorithm IndexedSkyline(threshold $\tau$)
02 Set resultSet := {};
03 PriorityQueue pq = new PriorityQueue(descending);
04 pq.insert( db.getRoot(), $\infty$ );
05 while (pq.hasUnprocessedElements()) do:
06   Element elem = pq.nextUnprocessedElement();
07   if(elem.isMBR())
08     pq.remove(elem);
09     for (Element child : elem.getChildElements()) do:
10       if(child.isMBR())
11         pq.insert(child, $\hat{P}_{IASP}^{pq}$(child));
12       else //child is an object
13         pq.insert(child, $\hat{P}_{MC}^{pq}$(child));
14     end for;
15     update $\hat{P}_{MC}^{pq}(f(\vec{x})), \check{P}_{MC}^{pq}(f(\vec{x})) : \ \forall f(\vec{x}) \in pq$
16     update $\hat{P}_{IASP}^{pq}(R) : \ \forall R \mid R$ is MBR $\wedge R \in pq$
17   else //elem is an object
18     if($\hat{P}_{MC}$(elem)$< \tau$)
19       pq.markAsProsessed(elem);
20       return resultSet;
21     else if($\check{P}_{MC}$(elem)$\geq \tau$)
22       pq.markAsProsessed(elem);
23       resultSet.add(elem);
24     else // we can not decide yet
25       MBR nextMBR = pq.getNextMBR();
26       pq.update(nextMBR, $\infty$); // is processed next
27 end while;
28 return resultSet;
```

Figure 5: Indexed Algorithm for Skyline Construction.

The indexed approach is backed by a priority queue held in main memory. The priority queue is sorted descending according to the $\hat{P}_{MC}(f(\vec{x}))$ of an object $f(\vec{x})$ and according to $\hat{P}_{IASP}(R)$ of a MBR R. The first element to be inserted in the priority queue is the root of the index. The next element to be processed is the one with the highest current skyline probability. If the next element is a node with MBR R, it is removed from the queue and all its child elements are inserted into the queue. By inserting the child elements, the values of $\hat{P}_{MC}(f(\vec{x}))$ and $\check{P}_{MC}(f(\vec{x}))$ of all objects $f(\vec{x})$ as well as $\hat{P}_{MC}(R)$ of all MBRs R are updated.

If the next element is an object $f(\vec{x})$, it can be returned provided that one of the two following conditions are fulfilled:

- if $\hat{P}_{MC}(f(\vec{x})) < \tau$ we can exclude the object $f(\vec{x})$ from the skyline.

- if $\check{P}_{MC}(f(\vec{x})) \geq \tau$ we know that $f(\vec{x})$ belongs to the skyline and can add it to the result set.

When we exclude an object $f(\vec{x})$ from the skyline we can stop searching for further objects in the queue, since all other objects have smaller skyline probabilities than $f(\vec{x})$. If we cannot decide for the current object if it belongs to the skyline or not, we process the priority queue's next MBR and insert its children. Thus, the skyline probabilities for all objects contained in the queue will be refined.

The pseudocode for the indexed τ-skyline algorithm is depicted in Figure 5. The update of the upper and lower bounds of the skyline probability ($\hat{P}_{MC}^{pq}(f(\vec{x})), \check{P}_{MC}^{pq}(f(\vec{x}))$) of every object $f(\vec{x})$ of the priority queue in line 14 can be done in one iteration step over the priority queue when inserting all the child elements of the former MBR. In this iteration the computation of the upper bound (line 15) of the independence assumption skyline probability

($\hat{P}^{pq}_{IASP}(R)$) for every MBR R of the priority queue can be done as well.

Our experiments presented in the next section demonstrate substantial performance gains by indexing. As with any index structure, the magnitude of improvement depends on data distribution and dimensionality. Thus, an acceptable worst case runtime complexity is an important characteristic. The indexed approach has a worst case runtime complexity of:

$$O\left(s \cdot n^2\right),$$

with s being the sample rate and n the number of objects in the database, since the objects are retrieved from the tree and afterwards in the worst case compared to each other.

6. EXPERIMENTAL EVALUATION

In this section, we present an extensive experimental evaluation on synthetic and real world data. Uncertain objects are provided by single Gaussian distributions or Gaussian Mixture Models. Unless otherwise specified, the following experiments have been performed on 4,000 uniformly distributed 3-dimensional uncertain objects with a sample-rate of 100 and a threshold of $\tau = 0.7$. Uncertainty is simulated by representing each object as Gaussians with mean between 0 and 1,000 and a variance between 0 and $\frac{1000}{4 \cdot \sqrt[d]{n}}$.

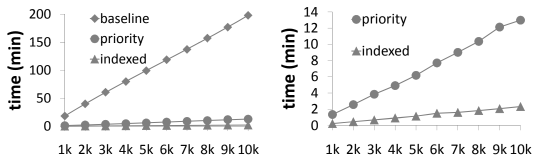
Besides synthetic data we used the NBA game-by-game technical statistics from 1991 to 2005 which is available at the NBA website `www.NBA.com`. The data set contains the performance statistics of 1,313 players. We represented each player as an uncertain object including 3 important statistics on his performance: number of points, number of assists, and number of rebounds (each defined by *mean* and *variance*).

We implemented all of our algorithms single threaded in Java. Unless otherwise noted, all experiments were performed on a SUN Fire X4600 with SunOS which is equipped with four Dual-Core AMD Opteron Processor and has 32 GByte of RAM. But in all experiments our algorithms only used 256 MB of Ram and a single CPU core.

Runtime w.r.t. Sample Rate. Figure 6 displays the effect of varying sample rates on NBA data. It is evident that all algorithms scale linearly with increasing sample rate (cf. Figure 6(a)). However, the linear factor is much larger for the baseline algorithm than for the other two. For 10,000 samples the baseline algorithm needs 198 minutes, whereas the priority algorithm needs 13 minutes. The indexed algorithm taking approximately 2 minutes is the fastest method. The reason for this is that relatively costly Montecarlo integrations are performed for each pair of objects in the baseline approach. Exploiting the lower bounding approximation even the simple priority algorithm saves much time. In the indexed approach, whole sub-trees can be pruned based on this efficient approximation, leading to superior runtime as displayed in detail in Figure 6(b).

Another experiment for the sample rate was evaluated on a synthetic generated Gaussian Mixture Model data set with 1500 objects in the database. As depicted in figure 7(a) the differences in runtime especially for the baseline are tremendous, here the priority and the indexed algorithms show their superiority.

Semantically the sample rate determines the level of abstraction which is used to represent an uncertain object. A higher sample rate may lead to more accurate results. Our experiments on NBA data demonstrate that a sample rate of 100 is sufficient to achieve a stable result. The skyline probabilities of the NBA players are displayed in Table 1. LeBron James is in all our results the most outstanding player. He is always several percentages better than the second best player according to the skyline probability. Players with almost equivalent skyline probability differ only in the 3rd decimal place e.g. Jason Kidd and Charles Barkley or Grant Hill and John Stockton. As depicted in Figure 7(b) the skyline probabilities of the NBA-players show highly stable results and only vary on average about 0.7% when repeating the experiment multiple times with sample rates ranging from 1000 to 4000.

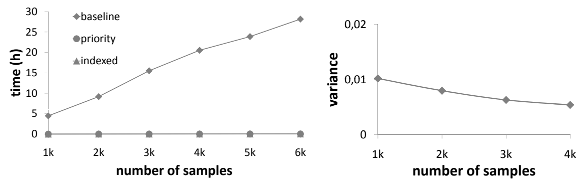


(a) all three algorithms (b) priority and indexed

Figure 6: Runtime for Different Sample Rates.

LeBron James	0.525133	Karl Malone	0.414282
Shaquille O'neal	0.496458	Allen Iverson	0.387400
Jason Kidd	0.484001	Chris Webber	0.383983
Charles Barkley	0.482588	Michael Jordan	0.379210
Dennis Rodman	0.466790	John Stockton	0.318612
Kevin Garnett	0.458139	Grant Hill	0.317689
Tim Duncan	0.420447	Kevin Johnson	0.312901

Table 1: NBA Players with a Skyline Probability of at least 0.3.

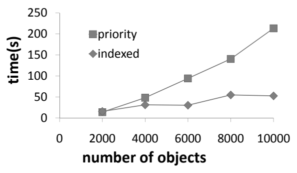


(a) sample rate (b) variance of results

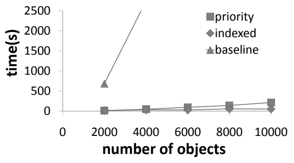
Figure 7: Variance of Results and Runtime for Sample Rate.

Runtime w.r.t. Number of Objects. Figure 8 shows the runtime behavior of the three algorithms on different database size. We generated synthetic data sets with varying number of objects. As it can be seen in Figure 8(b), the runtime of the baseline algorithm heavily depends on the database size n, whereas the priority and the indexed demonstrate their superiority. Even at moderate database sizes of 4,000 objects the baseline approach is outperformed by a factor of 57 compared to the priority algorithm. Figure 8(a) compares the runtime of priority and indexed algorithm. The indexed shows a speedup of factor 4 for 10,000 objects. The response time is only 52 seconds whereas the priority algorithm needs 213 seconds.

I/O Costs of the Indexed Approach. Another experiment demonstrates the I/O cost of the indexed algorithm. We defined a page size of 2 kByte and assumed a cache of 20% of the pages for both algorithms. In Figure 9(b) the resulting page accesses are depicted. It can be seen that the indexed outperforms the priority algorithms by magnitudes especially for large number of objects in the database. Comparing the data set with 10,000 objects, the indexed algorithm only has to access 186 pages whereas the priority has to load 6.8 times more pages, resulting in 1,280 page accesses.



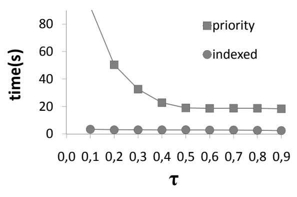
(a) priority and indexed



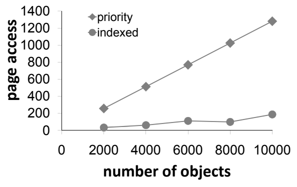
(b) all three algorithms

Figure 8: Runtime Analysis with Increasing Database Size.

Runtime w.r.t. τ. Figure 9(a) depicts the runtime on 1500 synthetic generated Gaussian Mixture Model data with increasing values for the skyline-threshold τ, ranging from 0.1 to 0.9. The runtime for the baseline algorithm is not included in this diagram, since its runtime does not depend on τ. Recall that the baseline approach first computes the skyline for all objects and just scans in a postprocessing step the result for objects having a higher skyline probability than τ. The runtime of the baseline approach is about 24 minutes for all τ values. The performance of the priority algorithm starts to degrade for thresholds lower than approximately 0.4, whereas the indexed approach scales only slightly sub-linear with decreasing τ. This indicates that pruning is effective even for very small threshold values. Even for the extreme case of $\tau = 0.1$ the query is processed in 4 seconds, which is a speedup of a factor of 24 w.r.t. the priority algorithm.



(a) different τ threshold values



(b) number of page accesses

Figure 9: Runtime Analysis on Changing τ Threshold and Page Accesses.

7. CONCLUSIONS

In this paper, we have introduced the probabilistic skyline query which is a powerful building block for data mining and decision support applications on uncertain data. This is the first approach extending the skyline operator to uncertain objects which are modelled by probability density functions. Analogously to the conventional skyline query, we provide definitions for dominance and skyline probability for uncertain objects. We propose an efficient algorithm to compute the skyline probability of each uncertain object and introduce the τ-skyline query, together with an efficient algorithms for query processing. The result set of the τ-skyline query contains all objects with a skyline probability of more than τ%. Our algorithms relies on a very simple data structure based on MBRs and thus can be integrated into many indexing techniques. In an extensive experimental evaluation on synthetic and real data we demonstrate that the proposed algorithms are highly efficient and scalable to large data sets.

8. REFERENCES

[1] R. S. Blum, Y. Zhang, B. M. Sadler, and R. J. Kozick. Approximation of correlated nongaussian noise pdfs using gaussian mixture models, published. In *American University, Washington DC*, 1999.

[2] C. Böhm, A. Pryakhin, and M. Schubert. The gauss-tree: Efficient object identification in databases of probabilistic feature vectors. In *ICDE*, page 9, 2006.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, pages 551–562, 2003.

[5] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB*, pages 876–887, 2004.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting: Theory and optimizations. In *Intelligent Information Systems*, pages 595–604, 2005.

[7] N. N. Dalvi and D. Suciu. Answering queries from statistics and probabilistic views. In *VLDB*, pages 805–816, 2005.

[8] A. Faradjian, J. Gehrke, and P. Bonnet. Gadt: A probability space adt for representing and querying the physical world. In *ICDE*, pages 201–211, 2002.

[9] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, pages 275–286, 2002.

[10] Q. Li, B. Moon, and I. Lopez. Skyline index for time series data. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):669–684, 2004.

[11] X. Lin, Y. Yuan, W. Wang, and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In *ICDE*, pages 502–513, 2005.

[12] K. Lu, Y. Qian, D. Rodríguez, W. Rivera, and M. Rodriguez. Wireless sensor networks for environmental monitoring applications: A design framework. In *GLOBECOM*, pages 1108–1112, 2007.

[13] A. M. Mainwaring, D. E. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA*, pages 88–97, 2002.

[14] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, pages 467–478, 2003.

[15] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.

[16] B. Sarikaya, M. A. Alim, and S. Rezaei. Integrating wireless eegs into medical sensor networks. In *IWCMC*, pages 1369–1374, 2006.

[17] A. D. Sarma, O. Benjelloun, A. Y. Halevy, and J. Widom. Working models for uncertain data. In *ICDE*, page 7, 2006.

[18] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *VLDB*, pages 301–310, 2001.

[19] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB*, pages 922–933, 2005.

[20] A. K. H. Tung, Z. Huang, H. Lu, and B. C. Ooi. Continuous skyline queries for moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1645–1658, 2006.

[21] D.-L. Yu and D.-W. Yu. Detecting sensor faults for a chemical reactor rig via adaptive neural network model. In *ISNN (3)*, pages 544–549, 2005.