A Survey of Uncertain Data Algorithms and Applications

Charu C. Aggarwal, *Senior Member*, *IEEE*, and Philip S. Yu, *Fellow*, *IEEE*

Abstract—In recent years, a number of indirect data collection methodologies have led to the proliferation of uncertain data. Such databases are much more complex because of the additional challenges of representing the probabilistic information. In this paper, we provide a survey of uncertain data mining and management applications. We will explore the various models utilized for uncertain data representation. In the field of uncertain data management, we will examine traditional database management methods such as join processing, query processing, selectivity estimation, OLAP queries, and indexing. In the field of uncertain data mining, we will examine traditional mining problems such as frequent pattern mining, outlier detection, classification, and clustering. We discuss different methodologies to process and mine uncertain data in a variety of forms.

Index Terms—Mining methods and algorithms, database applications, database management, information technology and systems.

1 Introduction

In recent years, many advanced technologies have been developed to store and record large quantities of data continuously. In many cases, the data may contain errors or may only be partially complete. For example, sensor networks typically create large amounts of uncertain data sets. In other cases, the data points may correspond to objects which are only vaguely specified, and are therefore considered uncertain in their representation. Similarly, surveys and imputation techniques create data which is uncertain in nature. This has created a need for *uncertain data management* algorithms and applications [2].

In uncertain data management, data records are typically represented by probability distributions rather than deterministic values. Some examples in which uncertain data management techniques are relevant are as follows:

- The uncertainty may be a result of the limitations of the underlying equipment. For example, the output of sensor networks is uncertain because of the noise in sensor inputs or errors in wireless transmission.
- In many cases such as demographic data sets, only partially aggregated data sets are available because of privacy concerns. Thus, each aggregated record can be represented by a probability distribution. In other privacy-preserving data mining applications, the data is perturbed in order to preserve the sensitivity of attribute values. In some cases, probability density functions of the records may be available. Some recent techniques [8] construct privacy models, such that the output of the transformation approach is friendly to the use of uncertain data mining and management techniques.
- In some cases, data attributes are constructed using statistical methods such as forecasting or imputation. In such cases, the underlying uncertainty in the derived data can be estimated accurately from the underlying methodology. An example is that of *missing data* [56].
- In many mobile applications, the trajectory of the objects may be unknown. In fact, many spatiotemporal applications are inherently uncertain, since the future behavior of the data can be predicted only approximately. The further into the future that the trajectories are extrapolated, the greater the uncertainty.

The field of uncertain data management poses a number of unique challenges on several fronts. The two broad issues are those of *modeling* the uncertain data, and then leveraging it to work with a variety of applications. A number of issues and working models for uncertain data have been discussed in [2] and [34]. The second issue is that of adapting data management and mining applications to work with the uncertain data. The main areas of research in the field are as follows:

- **Modeling of uncertain data.** A key issue is the process of modeling the uncertain data. Therefore, the underlying complexities can be captured while keeping the data useful for database management applications.
- **Uncertain data management.** In this case, one wishes to adapt traditional database management techniques for uncertain data. Examples of such techniques could be join processing, query processing, indexing, or database integration.
- **Uncertain data mining.** The results of data mining applications are affected by the underlying uncertainty in the data. Therefore, it is critical to design data mining techniques that can take such uncertainty into account during the computations.

In the next sections, we will discuss these different aspects of uncertain data representation, management, and mining. We will discuss the different issues with uncertain

- *C.C. Aggarwal is with the IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532. E-mail: charu@us.ibm.com.*
- *P.S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607. E-mail: psyu@cs.uic.edu.*

Manuscript received 12 Oct. 2007; revised 21 May 2008; accepted 2 Sept. 2008; published online 10 Sept. 2008.
Recommended for acceptance by C. Clifton.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-10-0501.
Digital Object Identifier no. 10.1109/TKDE.2008.190.

data representation, and their corresponding effect on database applications. We will also survey a broad variety of database management and mining applications.

This paper is organized as follows: In Section 2, we will examine the issue of uncertain data representation and modeling. In Section 3, we will examine a number of data management algorithms for uncertain data. We specifically examine the problems of query processing, indexing, selectivity estimation, OLAP, and join processing. A number of mining algorithms for uncertain data are discussed in Section 4. We examine the clustering and classification problem as well as a general approach to mining uncertain data. Section 5 contains the conclusions and summary.

2 Uncertain Data Representation and Modeling

The problem of modeling uncertain data has been studied extensively in the literature [1], [46], [45], [49], [72]. A database that provides incomplete information consists of *a set of possible instances of the database*. It is important to distinguish between incomplete databases and probabilistic data, since the latter is a more specific definition which creates database models with crisp probabilistic quantification.

2.1 Probabilistic Database Definitions

A probabilistic database is defined [45] as follows:

Definition 2.1. *A probabilistic-information database is a finite probability space whose outcomes are all possible database instances consistent with a given schema. This can be represented as the pair* $(\mathcal{X}, p)$, *where* $\mathcal{X}$ *is a finite set of possible database instances consistent with a given schema, and* $p(I)$ *is the probability associated with any instance* $I \in \mathcal{X}$. *We note that since* $p(\cdot)$ *represents the probability vector over all instances in* $\mathcal{X}$, *we have* $\sum_{I \in \mathcal{X}} p(I) = 1$.

We note that the above representation is a formalism of the "possible worlds model" [1]. The direct specification of such databases is unrealistic, since an exponential number of instances would be needed to represent the table. Therefore, the natural solution is to use a variety of simplified models which can be easily used for data mining and data management purposes. We will discuss more on this issue slightly later.

Probabilistic ?-tables [41], [54] are a simple way of representing probabilistic data. In this case, one models the probability that a particular tuple is present in the database. Thus, the probability of a particular instantiation of the database can be defined as the product of the probabilities of the corresponding set of tuples to be present in the database with the product of the probabilities of the complementary set of tuples to be absent from the database.

A closely related probabilistic representation is that of **probabilistic or-set tables.** While the probabilistic ?-table is concerned with the presence or absence of a particular tuple, the p-or-set table is concerned with modeling the probabilistic behavior of each attribute for a tuple that is known to be present in the database. In this case, each attribute is represented as an "or" over various possibilities along with corresponding probability values. An instantiation of the database is constructed by picking each outcome for an attribute independently. The ProbView model presented in [54] is a kind of or-set table. The only significant difference is that the ProbView model uses confidence values instead of probabilities. Other interesting representations of probabilistic databases may be found in [45]. A number of interesting properties of such databases are also discussed in [41], [54], and [80].

2.2 Simplifying Assumptions in Practical Applications

The above definitions are fairly general formalisms for probabilistic data. In many practical applications, one may often work with simplifying assumptions on the underlying database. One such simplifying assumption is that the presence and absence of different tuples is probabilistically independent. In such a formalism, all possible probability distributions on possible worlds are not captured with the use of independent tuples. This is referred to as *incompleteness*. Furthermore, one needs to be careful in the application of such a formalism, since it may result in *inconsistency*. For example, in an uncertain spatiotemporal database with tuples representing object locations at different times, the locations of the objects need to be consistent. In any particular instantiation of the database, it is important that the same object not be in multiple localities at the same time. Therefore, if the database is represented in terms of positional tuples, one needs to check for consistency of tuples within a given temporal locality. Such restrictions are often represented by rules that enforce the relationships between the behavior of the different tuples.

Most data mining or query processing applications work with further simplifications. For example, attribute-uncertainty models represent attributes by a discrete or continuous probability distribution, depending upon the data domain. Some applications [9] on continuous data may even work with statistical parameters such as the underlying variance of the corresponding attribute value. Thus, from an application point of view, the uncertainty may be represented in different ways, and it may not always conform to a database-centric view. Furthermore, uncertain databases are often designed with specific application goals in mind. Our discussions above can be summarized in terms of the major classes of uncertainty that most applications work with. Broadly, most applications work on two kinds of uncertainty: 1) **Existential uncertainty:** In this case, a tuple may or may not exist in the database, and the presence and absence of one tuple may affect the probability of the presence or absence of another tuple in the database. In some cases, the tuple independence assumption is used, according to which the probabilities of presence of the different tuples are independent of one another. Furthermore, there may be constraints that correspond to mutual exclusivity of certain tuples in the database. 2) **Attribute level uncertainty:** In this case, a number of tuples and their modeling have already been determined. The uncertainties of the individual attributes are modeled by a probability density function, or other statistical parameters such as the variance.

2.3 Recent Projects

A number of recent projects have designed uncertain databases around specific application requirements. For example, the *Conquer* project [3], [42] introduced query rewriting algorithms to extract clean and consistent answers from unclean data under possible worlds semantics. Methods are also proposed to derive probabilities of uncertain items. One of the key aspects of the Conquer project is that it permits real time and dynamic data cleaning in such a way that clean and consistent answers may be obtained for queries. Another example of such a database is the *Orion* project [25], [28] which presents query processing and indexing techniques in order to manage uncertainty over continuous intervals. Such application-specific databases are designed for their corresponding domain, and are not very effective in extracting information from "possible worlds" semantics.

A recent and interesting line of models for uncertain data is derived from the *Trio* project [16], [62], [34] at Stanford University. This work introduces the concept of Uncertainty-Lineage Database (*ULDB*), which is a database with both uncertainty and *lineage*. We note that the introduction of lineage as a first-class concept within the database is a novel concept which is useful in a variety of applications such as query processing. The basic idea in lineage is that the model keeps track of the *sources* from which the data was acquired and also keeps track of its influence in the database. Thus, database with lineage can link the query results (or the results from any potential application) to the source from which they were derived. The probabilistic influence of the data source on the final result is an important factor which should be accounted for in data management applications. Thus, data (or results) which are found to be unreliable are discarded.

Finally, a recent effort is the MayBMS project [4], [5], [6] at Cornell University. One advantage of this system is that it fits seamlessly into modern database systems. For example, this approach has a powerful query language which was built on top of PostgreSQL. Another unique feature of the system is that it uses the concept of U-relations in order to maximize space-efficiency. Space-efficiency is a critical feature in uncertain database systems, since the uncertainty results in considerable expansion of the underlying database representation. Details of the most recent approach may be found in [6].

2.4 Extensions to Semistructured and XML Data

Recently, uncertain data models have also been extended to semistructured and XML data. Some of the earliest work on probabilistic semistructured data may be found in [66]. XML data poses numerous unique challenges. Since XML is structured, the probabilities need to be assigned to the structural components such as nodes and links. Furthermore, element probabilities could occur at multiple levels and nested probabilities within a subtree must be considered. Furthermore, incomplete data should be handled gracefully since one may not insist on having complete probability distributions. In order to handle the issue that there can be nesting of XML elements, probabilities are associated with the attribute values of elements in an indirect way. The approach is to modify the schema in XML so as to make any attribute into a subelement. Thus, these new elements can be handled by the probabilistic system. Another unique issue in the case of XML data is that the probabilities in an ancestor-descendent chain are related probabilistically.

In the most general case, this can lead to issues of computational intractability. The approach in [66] is to model some classes of dependence (e.g., mutual exclusion) which are useful and efficient to model. The work in [66] also designs techniques for a restricted class of queries on the data. Another interesting approach to probabilistic XML data construction has been discussed in [50]. In this technique, probabilistic XML trees are constructed in order to model the structural behavior of the data. The uncertainty in a probabilistic tree is modeled by introducing two kinds of nodes: 1) probability nodes, which enumerate all possibilities, and 2) possibility nodes, which have an associated probability. The uncertainty in the different kinds of nodes is modeled with the use of the *kind* function, which assigns node kinds. Furthermore, a *prob* function is used, which assigns probabilities to nodes. The query evaluation technique enumerates all possible worlds in a recursive manner. The query is then applied to each such enumerated world. Other related work on XML data representation and modeling may be found in [79].

3 Uncertain Data Management Applications

In this section, we will discuss the design of a number of data management applications with uncertain data. These include applications such as query processing, Online Analytical Processing, selectivity estimation, indexing, and join processing. We will provide an overview of the application models and algorithms in this section.

3.1 Query Processing of Uncertain Data

In traditional database management, queries are typically represented as SQL expressions which are then executed on the database according to a query plan. As we will see, the incorporation of probabilistic information has considerable effects on the correctness and computability of the query plan.

3.1.1 Intensional and Extensional Semantics

A given query over an uncertain database may require computation or aggregation over a large number of possibilities. In some cases, the query may be nested, which greatly increases the complexity of the computation. There are two broad semantic approaches used:

- **Intensional semantics.** This typically models the uncertain database in terms of an event model (which defines the possible worlds), and use tree-like structures of inferences on these event combinations. This tree-like structure enumerates all the possibilities over which the query may be evaluated and subsequently aggregated. The tree-like enumeration results in an exponential complexity in evaluation time, but always yields correct results.
- **Extensional semantics.** Extensional semantics attempts to design a plan which can approximate these queries without having to enumerate the entire tree of inferences. This approach treats uncertainty as a generalized truth value attached to formulas, and attempts to evaluate (or approximate) the uncertainty of a given formula based on that of its subformulas.

For the intensional case, the key is to develop a probabilistic relational algebra with intensional semantics which always yields correct results. It has been shown in [32] that certain queries have #P-complete data complexity under intensional semantics. Note that the extensional semantics approach is mostly useful for simple expressions. When the relations are more complicated or nested, there may be dependencies in the underlying query results, which cannot be evaluated easily. Since intensional semantics uses a comprehensive enumeration-based approach, it always yields correct results, whereas extensional semantics provides an efficient heuristic, which is useful only when it works approximately or correctly. In order to understand this point, consider a possible worlds model of a database drawn from k possible tuples $s_1, s_2, \ldots, s_k$, each of which have probability of presence in the database equal to 0.2. The aim is to compute the probability that both s_1 and s_2 are present in the database. An intensional plan would therefore require us to explicitly create the event variables $e(s_1)$, $e(s_2)$ for s_1 and s_2 and compute the probability $P(e(s_1) \cap e(s_2))$. Note that each of the variables $e(s_1)$ and $e(s_2)$ will depend upon how the underlying database is modeled in terms of events, and will evaluate into a tree-like structure of inferences over possible worlds of events. An accurate and efficient extensional plan may not be possible in this case. On the other hand, if the correlations among different tuples are extremely weak or absent, then an efficient *extensional plan* would simply compute this probability as $P(s_1) * P(s_2) = 0.04$.

Clearly, a general method is required to reduce the query evaluation complexity in relational databases. One of the earliest techniques for adding probabilistic information into query evaluation was discussed in [41]. This model is a generalization of the standard relational model. In this model, probabilistic relations are treated as generalizations of deterministic relations. Thus, even though deterministic models allow binary tuple weights, probabilistic relations allow tuple weights which can vary between 0 and 1. The basic operators of relational algebra are modified in order to take the weights into account during query processing. Thus, while applying an operator of the relational algebra, the weights of the result tuples are computed as a function of the tuple weights in the argument relation. A more recent technique proposed in [32] designs a technique in which a correct extensional plan is available. We note that since the problem is #P-complete, a correct extensional plan is not always available. However, many queries, which occur in practice, do admit a correct extensional plan. According to [32], 8 out of 10 TPC/H queries[1] fall into this category. For queries which do not admit a correct extensional plan, two techniques are proposed to construct results which yield approximately correct answers. A fast heuristic is designed which can avoid large errors, and a sampling-based Monte-Carlo algorithm is designed which is more expensive, but can guarantee arbitrarily small errors. In addition, the technique in [32] also extends the solution to the case of *uncertain predicates* on deterministic data. A different imprecision model is discussed in [33] in which only the data statistics and explicit probabilities at the data sources are used. It is shown in [33] that such imprecisions can be modeled by a certain kind of probabilistic database with complex tuples correlations. The method in [32] is then used in order to rewrite the queries for effective query resolution. We note that the work in [32] assumes tuple independence which is often not the case for a probabilistic database. In the event that "possible worlds" semantics are used, the algorithms for query processing become much more difficult, since one needs to maintain consistency over the query answers. This problem is also related to that of determining consistent query answers in inconsistent databases [12].

1. The TPC-H benchmark is a standard suite of decision support queries developed for benchmarking. More details can be found at http://www.tpc.org/tpch.

3.1.2 Queries with Correlations

While the work in [32] assumes tuple independence, this may not always be the case in many practical applications. For example, data from sensors [35] may be highly correlated both in terms of space and time. Furthermore, even if it is assumed that the tuples are independent, many intermediate results of queries may contain complex correlations. For examples, even the simple join-operator is not closed under tuple independence. In [73], a technique has been proposed on querying correlated tuples with the use of statistical modeling techniques. The method in [73] constructs a uniform framework which expresses uncertainties and dependencies through the use of joint probability distributions. The query evaluation problem on probabilistic databases is cast as an inference problem in probabilistic graphical models [40]. Probabilistic graphical models form a powerful class of approaches which can compactly represent and reason about complex dependency patterns involving large numbers of correlated random variables. The main idea in the use of this approach is the use of factored representations for modeling the correlations. A variety of algorithms may then be used on the probabilistic graphical model, and the exact choice of algorithm depends upon the requirements for accuracy and speed.

3.1.3 Top-k Query

A related query is the top-k query in which the aim is to find the top-k answers for a particular query. The top-k ranking is based on some scoring function in deterministic applications. However, in uncertain applications, such a clean definition does not exist, since the process of reporting a tuple in a top-k answer does not depend only on its score but also on its membership probability. A further challenge is to use possible worlds semantics which can allow complex correlations among tuples in the database. In order to deal with the issue of possible worlds semantics, the technique in [74] uses *generation rules* which are logical formulas that determine valid worlds. The interplay of the possible worlds semantics with top-k queries requires the careful redefinition of the semantics for the query itself. For example, consider the case of a radar-controlled traffic system [74], in which the radar readings may be in error because of multiple sources of uncertainty such as interference from high-voltage lines and human identification mistakes. Some examples of deterministic top-k queries are as follows:

- Determine the top-k speeding cars in the last hour.
- Determine a ranking over the models of the top-k speeding cars.

While these queries are clear in the deterministic case, they need to be reformulated for the case of uncertain and imprecise data. For example, all responses to the queries need to be defined in valid possible worlds in order to avoid answers inconsistent with generation rules and other database constraints. Furthermore, in the second query, one may wish to evaluate the top-k query in a most probable world. The interaction between the "most probable" and the "top-k" results in different possible interpretations of uncertain top-k queries:

- The top-k tuples in the "most probable" world.
- The "most probable top-k" tuples that belong to valid possible worlds.
- The set of "most probable top-$i\text{th}$" tuples across all possible worlds, where $i = 1, \ldots, k$.

We note that the interpretations of the queries above involve both ranking and aggregations across possible worlds. The work in [74] models uncertain top-k queries as a state-space search problem, and introduces several space navigation algorithms with optimality guarantees on the number of accessed tuples in order to find the most probable top-k answers. In order to model the state-space probabilities, a *Rule Engine* is used, which is responsible for computing the state-space probabilities. This *Rule Engine* can be modeled in the form of a Bayesian Network [40]. The work in [74] also creates a framework for integrating space navigation algorithms and data access methods for leveraging existing DBMS technologies. One key result presented in [74] is that among all sequential access methods, the retrieval of tuples in the order of their scores leads to the least possible number of accessed tuples to answer uncertain top-k queries.

3.1.4 The OLAP Model

One interesting data model for query processing is that of the OLAP model. The queries which are most relevant to the OLAP setting are *aggregation queries*, in which one attempts to aggregate a particular function of the data on a part of the data cube. The earliest work in the aggregation setting was discussed in [23], [59], [70], and [71]. Much of this work does not relate directly to OLAP queries in the sense that while they provide aggregation functions, they do not use the domain hierarchies which are inherent in the OLAP environment. The earliest work in the OLAP setting was discussed in [64], which considers the semantics of aggregate queries in an uncertain environment. However, this technique does not consider the implications of an OLAP setting which uses domain hierarchies in order to define the data.

In [20], a crisp set of criteria has been identified in order to handle ambiguity. The criteria which are identified in [20] are as follows:

- **Consistency.** This criterion discusses the concept of consistency from the OLAP perspective. This accounts for the relationship between similar queries which are issued at related nodes in a domain hierarchy in order to meet users' intuitive expectations as they navigate up and down the hierarchy. For example, for the case of a SUM query, the SUM for a query region should be equal to the value obtained by adding the results of SUM for the query subregions that partition the region.
- **Faithfulness.** This captures the notion that more precise data should lead to more accurate results. For example, for a SUM query over nonnegative measures, as the imprecision in the data increases and grows outside the query region, it is expected that the result of the SUM query should be nonincreasing.
- **Correlation-preservation.** This requires that the correlation properties of the data should not be affected by the allocation of ambiguous data records. For example, the computation of the SUM under a tuple-specific constraint will be affected by the correlations among different tuples.

In order to model the uncertainty, the work in [20] relaxes the restriction that the dimension attributes must be assigned leaf-level values from the domain hierarchy. For example, we can denote that a repair took place in Texas without specifying a city explicitly. This has implications for how queries are answered: if a query aggregates repair costs in Austin, should the example repair be included, and how? The second extension is to introduce a new measure attribute which represents uncertainty. This is in the form of a probability distribution function over the base domain. Two broad approaches are proposed in [20] in order to deal with these different kinds of uncertainty:

- **Query allocation.** In this case, data which is assigned to higher levels of the hierarchy needs to be *allocated* to lower level leaf nodes by partial assignment. This partial assignment is captured by the weights on the assignment to nodes of different level. For response consistency, it is reasonable to expect that this assignment should be query independent.
- **Aggregating uncertain measures.** In this case, the query needs to aggregate over different probability density functions. The problem of aggregating pdfs is closely related to a problem studied in the statistics literature, which is that of *opinion pooling* [44]. The opinion-pooling problem is to form a *consensus opinion* from a given set of opinions θ. The set of opinions as well as the consensus opinion are presented as pdfs over a discrete domain O.

The work in [20] also allows a possible worlds interpretation of a database D containing imprecise facts, as a prelude to defining query semantics. If an imprecise fact r maps onto a region R of cells, then each cell in R represents a possible completion of r that eliminates the imprecision in r. More details may be found in [20]. Since imprecise data can often contain domain constraints in order to avoid inconsistency, a key issue is the extension of this model to the constrained case. In [21], the regularities in the constraint space are captured with the use of a constraint hypergraph in order to provide efficient answers to such queries.

3.2 Indexing Uncertain Data

The problem of indexing uncertain data arises frequently in the context of several application domains such as moving trajectories or sensor data. In such cases, the data is updated only periodically in the index, and therefore the current attribute values cannot be known exactly; they can only be estimated. There are many different kinds of queries which can be resolved with the use of index structures:

- **Range queries.** In range queries, the aim is to find all the objects in a given range. Since the objects are uncertain, their exact positions cannot be known, and hence their membership in the range also cannot be known deterministically. Therefore, a probability value is associated for each object to belong to a range. All objects whose probability of membership lies above a certain threshold are retained.

- **Nearest neighbor queries.** In nearest neighbor queries, we attempt to determine the objects with the least expected nearest neighbor distance to the target. An alternative way of formulating the probabilistic nearest neighbor query is in terms of the nonzero probability that a given object is the nearest neighbor to the target.
- **Aggregate queries.** In such queries, the aim is to determine the aggregate statistics from queries such as the *sum* or the *max*. Aggregate queries are inherently more difficult than other kinds of queries such as range or nearest neighbor queries because one has to account for the interplay of different objects.

In [25], a broad classification of the queries has been provided in the context of index structures. Queries can often be classified depending upon the nature of the answers. An *entity-based* query returns a set of objects that satisfy the condition of the query. A *value-based* query returns a single value, examples of which include the querying of the value of a particular dimension, or computing some statistical function of a set of objects satisfying query constraints (e.g., average, max). Another property which can be used to classify queries is whether or not aggregation is involved. In [25], broad classes of query processing techniques have been discussed for each of these different kinds of queries.

3.2.1 Moving Object Environments

An important domain for indexing and querying imprecise data is that of moving object environments [28]. In such environments, it is infeasible for the database tracking the movement of the objects to store the exact locations of the objects at all times. The location of an object is known with certainty only at the time of the update. Between two updates, the uncertainty of the location increases till the next update. The error in answers to queries can be controlled by limiting the level of uncertainty.

Several specific models of uncertainty are possible for the case of moving objects. One popular model for uncertainty is that, at any point in time, the moving object is within a certain distance d of its last reported position. If the object moves further than this distance, it reports its new location, and relocates its anchor point to the new reported position. Other models for uncertainty may assume specific patterns of movement such as that in a straight line. In such cases, the objects are assumed to lie in an interval along a straight line. In the case of [28], the uncertainty of a moving point is characterized in a fairly general way.

Definition 3.1. *An uncertainty region $U_i(t)$ of an object O_i at time t is a closed region such that O_i can be found only in this region.*

Definition 3.2. *The uncertainty density function $f_i(x, y, t)$ is the probability density function of the object O_i at location (x, y) and time t. This uncertainty function has a value of 0 outside $U_i(t)$.*

We note that this is a fairly general model of uncertainty in that it does not assume any specific behavior of the object inside $U_i(t)$.

Aside from the standard range query, the work in [28] also tackles the *probabilistic nearest neighbor query*. In the probabilistic nearest neighbor query, the aim is to determine probabilistic candidates for the nearest neighbor of a given target along with corresponding probability values.

The process of responding to a probabilistic range query is fairly straightforward. In this case, the probability density function is integrated over the entire range of the query. All objects for which this probability value lies above a certain threshold are reported.

The technique for processing a probabilistic nearest neighbor query involves evaluating the probability of each object being closest to the query point. One of the key challenges of the nearest neighbor query is that unlike the probabilistic range query, one cannot determine the probability for an object independent of the other points. The solution basically comprises the steps of *projection*, *pruning*, *bounding*, and *evaluation*. These steps are summarized as follows:

- **Projection.** In this phase, the uncertainty region of each moving object is computed based on the uncertainty model used by the application. The shapes of the uncertainty regions are determined by the uncertainty model used, the last recorded position of the object O_i, the time elapsed since the last update, and the maximum speeds of the objects.
- **Pruning phase.** This allows us to explicitly prune some of the objects without having to go through the expensive process of computing nearest neighbor probabilities. For example, if the shortest distance of the target to one uncertain region is greater than the corresponding longest distance of the target to another region, then it is possible to prune the former. Therefore, the key to the algorithm is to find f, the minimum of the longest distances of the uncertainty regions from the target q. Then, any object for which the shortest distance to the target q is larger than f is eliminated.
- **Bounding phase.** The pruning can be extended to portions of uncertainty regions which cannot be completely pruned. For each element, there is no need to examine all portions of the uncertainty region. It is necessary to only look at the regions that are located no farther than f from the target point q. This can be conceptually achieved by drawing a bounding circle C of radius f centered at q. Any portion of the uncertainty region outside C can be ignored.
- **Evaluation phase.** In this phase, one calculates for each object, the probability that it is indeed the nearest neighbor to the target O. The solution is based on the fact that the probability of an object o being the nearest neighbor with distance r to the target q is given by the probability of o being at a distance r from q times the probability that every other object is at a distance r or larger from q. This value can then be integrated over different values of r.

3.2.2 Probabilistic Threshold Queries

A related work in [24] proposes the concept of *probabilistic threshold queries*. In such queries, the aim is to determine all objects whose behavior satisfies certain conditions with a minimum probability. The formal definition is as follows:

Definition 3.3. *Given a closed interval $[c, d]$, where $c, d \in \mathcal{R}$ and $c \leq d$, a probabilistic threshold query returns a set of tuples T_i, such that the probability p_i that $T_i.a$ is inside $[c, d]$, is greater than or equal to p, where $0 \leq p \leq 1$. We note that $T_i.a$ represents the probability attribute of tuple T_i.*

Thus, a probabilistic threshold query can be treated as a range query, which operates on probabilistic uncertainty information, and returns items whose probabilities of satisfying the query exceed p.

A number of index structures have been proposed in [24] in order to resolve this query. A naive way of evaluating the query is to first find all the tuples whose uncertainty intervals have some overlap with the corresponding range. Once these tuples have been determined, the corresponding probability of intersection can be determined in a straightforward way. In order to find all the tuples which intersect over a given range, it is necessary to build an index structure over different intervals, and apply a range search over the index for the prespecified interval. This can unfortunately be quite inefficient. The second problem is that of the probability of each element in the data needs to be evaluated. If many items overlap with the specified interval, but only a few have probability of inclusion greater than p, then this can be quite inefficient.

A different solution proposed in [24] is referred to as *Probability Threshold Indexing*. This index structure is essentially based on a modification of a 1D R-Tree, where probability information is augmented to its internal nodes in order to facilitate pruning. In a traditional R-Tree, a range query is resolved by examining only those nodes of the tree which intersect with the user-specified range. This idea can be generalized by constructing tighter bounds (called x-bounds) than the Minimum Bounding Rectangle (MBR) of each node. Let M_j denote the MBR/uncertainty interval represented by the jth node of an R-Tree, ordered by preorder traversal. Then, the x-bound of M_j is defined as follows:

Definition 3.4. *An x-bound of an MBR/uncertainty interval M_j is a pair of lines, namely left-x-bound (denoted by $M_j.lb(x)$) and right-x-bound (denoted by $M_j.rb(x)$). Every uncertain object contained in this MBR is guaranteed to have a probability of at most x (where $0 \leq x \leq 1$) of being left of the left-x-bound and also guaranteed to have a probability of at most x of being right of the right-x-bound.*

We note that this kind of bound is a generalization of the concept of the MBR. This is because the MBR of an internal node can be viewed as a 0-bound. This is because it guarantees that all intervals in the node are contained in it with probability 1.

The purpose of storing the information of the x-bound of a node is to avoid investigating the contents of a node. This saves I/O costs during index exploration. The presence of the x-bound allows us to decide whether an internal node contains any qualifying MBRs without further probing into the subtrees of this node. Let p be the threshold probability for the query. The two necessary pruning conditions (both conditions must hold) for node M_j to be pruned with the use of the x-bound are as follows:

- M_j can be pruned if $[a, b]$ does not intersect left-x-bound or right-x-bound of M_j, i.e., either $b < M_j.lb(x)$ or $a > M_j.rb(x)$.
- $p \geq x$.

In the event that the above conditions do not hold, the internal contents of node M_j are examined and further exploration of the tree is resumed. It has been shown in [24] that the probability threshold query (PTQ) index is quite efficient when the threshold p is fixed a priori across all queries. When the threshold p varies, then the index continues to be experimentally efficient on the average, though the actual behavior mat vary quite a bit across different queries.

3.2.3 Uncertain Categorical Data

A method for indexing uncertain categorical data has been discussed in [76]. The definition used in [76] for the categorical data domain is as follows:

Definition 3.5. *Given a discrete categorical domain $D = \{d_1, \ldots, d_N\}$, an uncertain discrete attribute (UDA) u is a probability distribution over D. It can be represented by the probability vector $u.P = \{p_1, \ldots, p_N\}$ such that $Pr(u = d_i) = u.p_i$.*

The probability that two uncertain attribute values are equal can be computed by calculating the corresponding equality probability over all possible uncertain values. Therefore, we have the following.

Observation 3.1. *Given two UDAs u and v, the probability that they are equal is given by $Pr(u = v) = \sum_{i=1}^{N} u.p_i \times v.p_i$.*

Analogous to the notion of equality of value is distributional similarity. The distance function may be defined in terms of the L_1 function, the L_2 function, or the Kullback-Leibler distance function. The kinds of queries resolved by the technique in [76] are as follows:

- **Probabilistic equality query (PEQ).** Given a UDA q, and a relation R with a UDA a, the query returns all tuples t from R along with probability values, such that the probability value $Pr(q = t.a) \geq 0$.
- **Probabilistic equality threshold query (PETQ).** Given a UDA q, a relation R with UDA a, and a threshold τ, $\tau \geq 0$. The answer to the query is all tuples t from R such that $Pr(q = t.a) \geq \tau$.
- **Distributional similarity threshold query (DSTQ).** Given a UDA q, a relation R with UDA a, a threshold τ_d, and a divergence function F, DSTQ returns all tuples t from R such that $F(q, t.a) \leq \tau_d$.
- **Probabilistic equality threshold join (PETJ).** Given two uncertain relations R, S, both with UDAs a, b, respectively; relation $R \bowtie_{R_a = S_b, \tau} S$ consists of all pairs of tuples r, s from R, S, respectively, such that $Pr(r.a = s.b) \geq \tau$.

In [76], two separate index structures are proposed in order to resolve the queries on categorical uncertain data. The first index is the *probabilistic inverted index*. In the probabilistic inverted index, for each value in the categorical domain, a list of the tuple-ids is stored, which have a nonzero probability of taking on that particular value. Along with each tuple-id, this probability value is also stored. The inner lists containing the tuple-ids are often organized as a dynamic structure such as the B-Tree in order to facilitate insertions and deletions. As in any inverted index, the insertion and deletion are extremely straightforward. One only needs to determine the corresponding list(s), and insert or delete the corresponding tuple-id.

The inverted index can be used in conjunction with various pruning techniques in order to answer PETQs. The first step is to determine all the tuples in the different inverted lists which match the target parameters of the query. From these candidate tuples, only those which qualify more than the threshold are retained. A variety of other pruning techniques can be used in order to improve the efficiency of the different queries. The different techniques discussed in [76] include row pruning, column pruning, and approaches which examine the lists in a highest-probability first fashion. The effectiveness of these different techniques for different kinds of queries is discussed in [76].

3.2.4 Probabilistic Distribution R-Tree

Next, we will discuss the probabilistic distribution R-Tree which is an alternative for indexing UDAs. The broad approach is to *index the vector of probability values of the possible attribute values*. Thus, if there are N possible probability values then, data points are created in R^N. One distinction from traditional R-Trees is that the underlying queries have very different semantics. The uncertain queries are hyperplane queries on the N-dimensional cube. The MBRs of this R-Tree are thus defined in terms of the corresponding probability values. This ensures that the essential pruning properties of R-Trees are maintained. For example, for the case of probabilistic threshold query, one can compute the maximum probability of equality for any node in the subtree by taking the maximum dot product of the target object probabilities with the corresponding probability vector from the MBR. When this value is less than the user-specified threshold, the corresponding subtree can be pruned. The two different index structures for categorical data have been tested in [76]. The results suggest that neither of the two techniques emerges as a clear winner, and either of the techniques may perform better depending upon the nature of the query and the underlying data.

3.2.5 Other Work

Most of the above techniques make certain assumptions about the underlying probability distributions. An interesting technique discussed in [77] examines the problem for the case of arbitrary probability density functions. In this case, a general assumption is made about the probability distribution functions, in the sense that they are not all assumed to be even of the same type. For example, the uncertainty function for one object could be uniform, whereas the uncertainty function for another object could be Gaussian. This makes the problem much more difficult from the point of view of indexing, search, and pruning. In [77], an index structure called the U-Tree has been proposed, which can handle such kinds of queries. Other methods for indexing arbitrary probability distributions have been discussed in [19] and [57]. Finally, an interesting method called the Gauss-Tree [18] has been proposed for the case of probabilistic feature vectors. This tree has been shown to retain effectiveness for probabilistic retrieval. A detailed discussion is beyond the scope of this survey.

3.3 Join Processing on Uncertain Data

In the case of join processing, techniques have been developed for probabilistic join queries and similarity joins. In the case of probabilistic join queries, it is assumed that each item is associated with a range of possible values and a probability density function, which quantifies the behavior of the data over that range. The range of values associated with the uncertain variable a are denoted by $a.U = [a.l, a.r]$. Thus, $a.l$ is the lower bound of the range and $a.r$ is the upper bound of the range. By incorporating the notion of uncertainty into data values, imprecise answers are generated. Each join-pair is associated with a probability to indicate the likelihood that the two tuples are matched. A second kind of join [53] is the *similarity join*. Similarity is measured by the distance between the two feature vectors. The join is performed based on this distance.

3.3.1 Probabilistic Join Queries

Since each tuple-pair is probabilistic in nature, the join may contain a number of false positives which are typically those pairs which are associated with probability values. Each tuple-pair is associated with a probability that indicates the likelihood of the join. In order to compute these probability values, the notions of equality and inequality need to be extended to support uncertain data. We note that those join-pairs which have low probability can be discarded. This variant of probabilistic join queries are referred to as *Probabilistic Threshold Join Queries*. We note that the use of thresholds reduces the number of false positives, but it may also result in the introduction of false negatives. Thus, there is a tradeoff between the number of false positives and false negatives depending upon the threshold which is chosen. The reformulation of the join queries with thresholds is also helpful in improving the performance requirements of the method.

A number of pruning techniques are developed in order to improve the effectiveness of join processing. These pruning techniques are as follows: 1) **Item-level pruning:** In this case, two uncertain values are pruned without evaluating the probability. 2) **Page-level pruning:** In this case, two pages are pruned without probing into the data stored in each page. 3) **Index-level pruning:** In this case, the data which is stored in a subtree is pruned.

We note that a key operator in the case of joins is that of *equality*, since a join is performed only when the corresponding attribute values are equal. For the case of continuous data with infinitesimal resolution, this is never the case since any of the pair of attributes can take on an infinite possible number of values. Therefore, a pair of attributes are defined to be equal to one another within acceptable resolution c, if one attribute value is within c of another. Let a and b be the two join attributes. Let $a.f(x)$ and $b.F(x)$ represent the corresponding probability density and cumulative density functions, respectively. Correspondingly, the probability can be calculated as follows:

$$P(a =_c b) = \int_{-\infty}^{+\infty} a.f(x) \cdot (b.F(x+c) - b.F(x-c))dx. \quad (1)$$

For the case of the $>$ and $<$ operators, it is not necessary to use the resolution, and it is possible compute the corresponding probability of inequality $P(a > b)$ and $P(a < b)$ in a straightforward way. In order to evaluate the join, common block-nested-loop and indexed-loop can be used. The advantage of these algorithms is that they have been implemented in most database systems, and therefore only a small amount of modification is required in order to support the joins. The main difference is to use the uncertainty information in order to compute the probability of equality. For the use of probability density functions such as the uniform or the Gaussian function, closed form formulas may be obtained in order to determine the probability of equality. Subsequently, those pairs with probability less than the required threshold can be pruned.

We note that the computations of the probability of a join can sometimes be expensive when the probabilistic computations cannot be expressed in closed form. Therefore, it is often useful to be able to develop quick pruning conditions in order to exclude certain tuple pairs from the join. Suppose a and b are uncertain valued variables and $a.U \cap b.U \neq \phi$. Let $l_{a,b,c,}$ be $\max\{a.l - c, b.l - c\}$, and let $u_{a,b,c}$ be $\min\{a.r + c, b.r + c\}$. For equality and inequality, the following pruning conditions hold true:

- $P(a =_c b)$ is at most $\min\{a \cdot F(u_{a,b,c}) - a \cdot F(l_{a,b,c}), b \cdot F(u_{a,b,c}) - b \cdot F(l_{a,b,c}))\}$.
- Correspondingly, it is easy to see that $P(a \neq_c b)$ is at least equal to the complement of the above expression.

We note that the above expressions can be computed easily as long as the cumulative density function of the expression is available either in closed or numerical form. We note that a tuple pair can be eliminated when the probability of equality is less than the user-defined threshold.

We further note that in some cases, it may not be necessary to report the explicit probabilities of the tuple joins, as long as all tuples whose join probability is above the user-defined threshold are reported. For such cases, it is only necessary to determine whether the required probability lies above a given threshold. For such cases, we can use another pruning condition.

For a pair of uncertain-valued variables a and b, it is possible to compute a bound on the corresponding probability that one is greater than the other. Specifically, the bounds are as follows:

- If $a.l \leq b.r < a.r$, $P(a > b) \geq 1 - a \cdot F(b.r)$.
- If $a.l \leq b.l \leq a.r$, $P(a > b) \leq 1 - a \cdot F(b.l)$.

The detailed proof of these results is described in [27]. The above two inequalities can be used for those join tuples which satisfy the preconditions described above. Depending upon the direction of the inequality, one can immediately include or exclude the corresponding join tuples from the inequality.

We note that in many of these join processing algorithms, the unit of retrieval is a page from an index structure. In such cases, one can prune the entire node of the index tree by constructing bounds on the join behavior of the nodes in the tree. By using this approach, either page-level pruning can be achieved, or index-level pruning can be achieved by using an inner level node in the index tree. A concept called the x-bound is proposed in [26], and is used to augment the nodes of the underlying index structure. For more details, we refer the reader to [26]. Another recent method for spatial joins is discussed in [58].

3.3.2 Similarity Join

The most popular similarity join is the distance-range join. In the distance-range join, we perform the join between two records, if the distance between the two does not exceed a user-defined parameter ϵ. The natural generalization for the case of uncertain data is to compute the expected distance between two relations, and perform the join if this expected distance is less than the parameter ϵ. This may result in considerable inaccuracies in the join computation process. This is because the expected distances are often skewed by the behavior of the tail end behavior of the probability functions of different attributes. Thus, the expected distances may not reflect the true likelihood that a given pair of records may join on a particular attribute. The result is that different joins which have similar probability of lying within the range of ϵ may be treated inconsistently. Therefore, it has been proposed in [53] to assign a probability value to each object pair. This probability value reflects the likelihood that the object pair belongs to the join result set. Only those join pairs which have nonzero probability of belonging to the join-result set are returned. In order to define this probability, one needs to quantify whether the distance between a pair of joining attributes lies within a certain range. To do so, the method in [53] computes the probability that the distance between the pair does not exceed ϵ. We note that in the deterministic case, when the distances are known, this distance function is the dirac-delta function. Thus, the deterministic case is a special case of the uncertain similarity join algorithm.

3.4 Data Integration with Uncertainty

An important application in the context of uncertain data is that of *data integration*. A first approach to this problem has been discussed in [37]. In order to do so, the work in [37] introduces the concept of *probabilistic schema mappings*. These are defined as a set of *possible* (ordinary) mappings between a source schema and a target schema, where each possible mapping has an associated probability. It is suggested that there are two possible interpretations to probabilistic schema mappings. The first (table-specific mapping) assumes that there is a single correct mapping, but we do not know which it is. This single correct mapping applies to all tuples. In the second interpretation (tuple-specific mapping), the mapping depends upon the tuple to which it is applied.

A number of algorithms are described in [37] for answering queries in the presence of probabilistic schema mappings. It has been shown that in the case of table-specific mappings, the data complexity is PTIME, and in the case of tuple-specific mappings, the complexity is #P-complete. Therefore, the second case is much more difficult. Nevertheless, it has been shown in [37] that for large classes of real-world queries, it is possible to obtain all the answers in PTIME. More details on the specific algorithms may be found in [37].

3.5 Probabilistic Skylines on Uncertain Data

A problem which is quite relevant to the case of uncertain data is that of probabilistic skyline computation. The work in [65] provides a first approach to this problem. The problem of skyline computation is used in multicriteria decision-making applications. For example, consider the case when statistics of different NBA players are computed, such as the number of assists, rebounds, baskets, etc. It is unlikely that a single player will achieve the best performance in all respects. Therefore, the concepts of dominance and skyline are defined [65] as follows:

Definition 3.6. *For two d-dimensional points $u = (u_1, \ldots, u_d)$ and $v = (v_1, \ldots, v_d)$, u is said to* dominate *v, if for each $i \in \{1, \ldots, d\}$, we have $u_i \leq v_i$, and for some $i_0 \in \{1, \ldots, d\}$, we have $u_{i_0} < v_{i_0}$.*

The above definition assumes that smaller values are more preferable, though it is easy enough to create a definition in which larger values may be preferable for one or more of the dimensions. The concept of dominance can be used in order to formally define the concept of a skyline.

Definition 3.7. *Given a set of points S, a point u is a skyline point if there exists no other point $v \in S$ such that v dominates u. The* skyline *on S is the set of all skyline points.*

Clearly, all players that lie on the skyline may be considered outstanding players. Most skyline analyses only use certain data in the form of the mean performance of the different players. In practice, the performance of a player on different criteria may vary substantially from game to game. For example, it is known that most players are far more effective, when playing on their home court. Therefore, it is possible to improve the quality of the analysis by using uncertainty information.

The key challenge in skyline computation is to capture the dominance relationship between uncertain objects. Therefore, the concept of *probabilistic skyline* was proposed in [65]. In this case, the probability of an object being in the skyline is the probability that the object is not dominated by any other objects.

Definition 3.8. *Given a probability threshold p $(0 \leq p \leq 1)$, the p-skyline is the set of uncertain objects, such that each of them has probability of at least p to be in the skyline.*

Constructing a probabilistic skyline is much more complicated, because in many applications, the probability density function of uncertain data objects is not available explicitly. Only a set of instances are collected in order to approximate the probability density function. For example, in the case of the NBA example, the instances correspond to the game-by-game performance of a particular player, whereas the uncertain object corresponds to the distribution of a particular player's performance. One possible solution is to apply the skyline approach on the entire collected set of instances. However, this can be inefficient in practice, when the set of collected instances are very large compared to the underlying objects on which the skylines are computed.

In [65], two algorithms are proposed. The first is a bottom-up algorithm which computes the skyline probabilities of some selected instances of objects, and uses those instances to prune other instances and uncertain objects effectively. The second is a top-down algorithm which recursively partitions the instances of uncertain objects into subsets, and prunes subsets and objects aggressively. Both the top-down and bottom-up algorithms use the bounding-pruning-refining iteration. In the case of the bottom-up algorithm, the steps are as follows:

- **Bounding.** For an instance of an uncertain data object, we compute an upper bound and lower bound of its skyline probability. Then, we can convert this bound to the skyline probability of an uncertain object.
- **Pruning.** If the lower bound of an uncertain object U is larger than the threshold p, then it lies in the p-skyline. If the upper bound is less than p, then U is not in the p-skyline.
- **Refining.** For all objects which cannot be conclusively determined to be either excluded or included in the skyline, we need to get tighter bounds for the next iteration of bounding, pruning, and refining.

An important observation here is that in this method, we compute and refine the bounds of instances of uncertain objects by selectively computing the skyline probabilities on a small subset of instances. This technique is called bottom-up, since the bound computation and refinement start from instances in the bottom, and go up to skyline probabilities of objects. We refer the reader to the details of how the bounding and refinement are performed to [65].

4 Mining Applications for Uncertain Data

Recently, a number of mining applications have been devised for the case of uncertain data. Such applications include clustering and classification. We note that the presence of uncertainty can affect the results of data mining applications significantly. For example, in the case of a classification application, an attribute which has lower uncertainty is more useful than an attribute which has a higher level of uncertainty. Similarly, in a clustering application, the attributes which have a higher level of uncertainty need to be treated differently from those which have a lower level of uncertainty.

4.1 Clustering Uncertain Data

The presence of uncertainty changes the nature of the underlying clusters, since it affects the distance function computations between different data points. A technique has been proposed in [51] in order to find density-based clusters from uncertain data. The key idea in this approach is to compute uncertain distances effectively between objects which are probabilistically specified. The fuzzy distance is defined in terms of the distance distribution function. This distance distribution function encodes the probability that the distances between two uncertain objects lie within a certain user-defined range. Let $d(\overline{X}, \overline{Y})$ be the random variable representing the distance between $\overline{X}$ and $\overline{Y}$. The distance distribution function is formally defined as follows:

Definition 4.1. *Let $\overline{X}$ and $\overline{Y}$ be two uncertain records, and let $p(\overline{X}, \overline{Y})$ represent the distance density function between these*

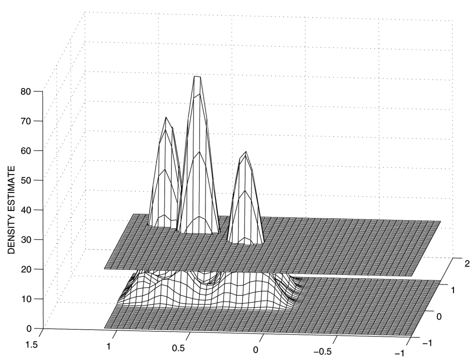

Fig. 1. Density-based profile with lower density threshold.

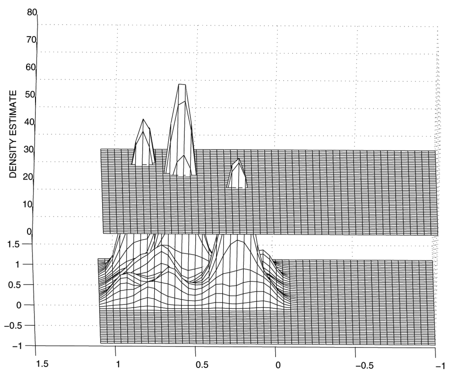

Fig. 2. Density-based profile with higher density threshold.

objects. Then, the probability that the distance lies within the range (a, b) is given by the following relationship:

$$P\big(a \leq d(\overline{X}, \overline{Y}) \leq b\big) = \int_a^b p(\overline{X}, \overline{Y})(z)dz. \tag{2}$$

Based on this technique and the distance density function, the method in [51] defines a *reachability probability* between two data points. This defines the probability that one data point is directly reachable from another with the use of a path, such that each point on it has density greater than a particular threshold. We note that this is a direct probabilistic extension of the deterministic reachability concept which is defined in the DBSCAN algorithm [38]. In the deterministic version of the algorithm [38], data points are grouped into clusters when they are reachable from one another by a path which is such that every point on this path has a minimum threshold data density. To this effect, the algorithm uses the condition that the ϵ-neighborhood of a data point should contain at least $MinPts$ data points. The algorithm starts off at a given data point and checks if the ϵ neighborhood contains $MinPts$ data points. If this is the case, the algorithm repeats the process for each point in this cluster and keeps adding points until no more points can be added. One can plot the density profile of a data set by plotting the number of data points in the ϵ-neighborhood of various regions, and plotting a smoothed version of the curve. This is similar to the concept of probabilistic density estimation. Intuitively, this approach corresponds to the continuous contours of intersection between the density thresholds in Figs. 1 and 2 with the corresponding density profiles. The density threshold depends upon the value of $MinPts$. Note that the data points in any contiguous region will have density greater than the threshold. Note that the use of a higher density threshold (Fig. 2) results in three clusters, whereas the use of a lower density threshold results in two clusters. The fuzzy version of the DBSCAN algorithm (referred to as FDBSCAN) works in a similar way to the DBSCAN algorithm, except that the density at a given point is uncertain because of the underling uncertainty of the data points. This corresponds to the fact that the number of data points within the ϵ-neighborhood of a given data point can be estimated only probabilistically, and is essentially an uncertain variable. Correspondingly, the reachability from one point to another is no longer deterministic, since other data points may lie within the ϵ-neighborhood of a given point with a certain probability, which may be less than 1. Therefore, the additional constraint that the computed reachability probability must be greater than 0.5 is added. Thus, this is a generalization of the deterministic version of the algorithm in which the reachability probability is always set to 1.

Another related technique discussed in [52] is that of hierarchical density based clustering. An effective (deterministic) density based hierarchical clustering algorithm is OPTICS [13]. We note that the core idea in OPTICS is quite similar to DBSCAN and is based on the concept of *reachability distance* between data points. While the method in DBSCAN defines a *global density parameter* which is used as a threshold in order to define reachability, the work in [52] points out that different regions in the data may have different data density, as a result of which it may not be possible to define the clusters effectively with a single density parameter. Rather, many different values of the density parameter define different (hierarchical) insights about the underlying clusters. The goal is to define an implicit output in terms of ordering data points, so that when the DBSCAN is applied with this ordering, one can obtain the hierarchical clustering at any level for different values of the density parameter. The key is to ensure that the clusters at different levels of the hierarchy are consistent with one another. One observation is that clusters defined over a lower value of ϵ are completely contained in clusters defined over a higher value of ϵ, if the value of $MinPts$ is not varied. Therefore, the data points are ordered based on the value of ϵ required in order to obtain $MinPts$ in the ϵ-neighborhood. If the data points with smaller values of ϵ are processed first, then it is assured that higher density regions are always processed before lower density regions. This ensures that if the DBSCAN algorithm is used for different values of ϵ with this ordering, then a consistent result is obtained. Thus, the output of the OPTICS algorithm is not the cluster membership, but it is the order in which the data points are processed. We note that that since the OPTICS algorithm shares so many characteristics with the DBSCAN algorithm, it is fairly easy to extend the OPTICS algorithm to the uncertain case using the same approach as that was used for extending the DBSCAN algorithm. This is referred to as the FOPTICS algorithm. Note that one of the core-concepts needed to order to data points is to determine the value of ϵ which is needed in order to obtain $MinPts$ in the corresponding neighborhood. In the uncertain case, this value is defined probabilistically, and the corresponding expected values are used to order the data points.

Finally, a technique in [63] uses an extension of the K-means algorithm in order to cluster the data. This technique is referred to as the UK-means algorithm. In UK-means, an object is assigned to the cluster whose representative has the smallest expected distance to the object. We note that expected distance computation is an expensive task. Therefore, the technique in [63] uses a number of pruning operations in order to reduce the computational load. The idea here is to use branch-and-bound techniques in order to minimize the number of expected distance computations between data points and cluster representatives. The broad idea is that once an upper bound on the minimum distance of a particular data point to some cluster representative has been quantified, it is necessary to perform the computation between this point and another cluster representative, if it can be proved that the corresponding distance is greater than this bound. This approach is used to design an efficient algorithm for clustering uncertain location data.

The techniques in [51] and [63] were developed for the case of static data. Recently, the problem of clustering uncertain data has also been extended to the case of data streams [10]. In order to do so, the microclustering concept developed in [11] is extended to the uncertain case. In order to incorporate uncertainty into the clustering process, additional information about error statistics are incorporated into microclusters. It has been shown in [10] that it is possible to efficiently cluster uncertain data streams with the use of such an approach. More recently, approximation algorithms [31] have been proposed for clustering uncertain data.

4.2 Classification of Uncertain Data

A closely related problem is that of classification of uncertain data in which the aim is to classify a test instance into one particular label from a set of class labels. In [17], a method was proposed for support vector machine classification of uncertain data. This technique is based on a discriminative modeling approach which relies on a total least squares method. This is because the total least squares method assumes a model in which we have additive noise. However, instead of using Gaussian noise, the technique in [17] uses a simple bounded uncertainty model. Such a model has a natural and intuitive geometric interpretation. Note that the support vector machine technique functions by constructing boundaries between groups of data records. Then, the margin created by the support vector machine can be modified by using the uncertainty of the points which lie on the boundary. For example, if points on one side of the boundary have greater uncertainty, this influences the way in which the margins are adjusted by the classifier. This is because the uncertainty in the data may result in some probability that the uncertain data point is on either side of the SVM boundary. The key idea in [17] is to provide a geometric algorithm which optimizes the *probabilistic separation* between the two classes on both sides of the boundary. Thus, the main difference from a standard SVM approach is to use the *probability* that a given data point lies on either side of the boundary while computing the degree of separation between the two classes.

4.3 Frequent Pattern Mining

The problem of frequent pattern mining has also been explored in the context of uncertain data. In this model, it is assumed that each item has an existential uncertainty in belonging to a transaction. This means that the probability of an item belonging to a particular transaction is modeled in this approach. In this case, an item set is defined to be frequent, if its *expected support* is at least equal to a user-specified threshold.

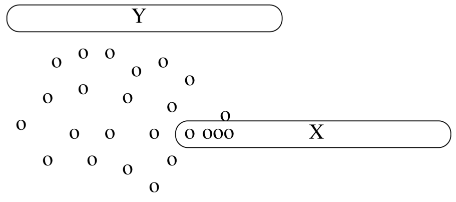

Fig. 3. Effect of uncertainty on outlier detection.

In order to solve this version of the frequent pattern mining problem, the *U-Apriori* algorithm is proposed which essentially mimics the *Apriori* algorithm, except that it performs the counting by computing the expected support of the different item sets. The expected support of a set of items in a transaction is obtained by simply multiplying the probabilities of the different items in the transaction. The approach can be made further scalable by using the concept of *data trimming*. In the data trimming approach, those items with very low existential probability are pruned from the data. The algorithm is then applied to the trimmed data. In [29], it has been shown that this approach can accurately mine the frequent patterns while maintaining efficiency. Further pruning tricks for improving the efficiency of frequent pattern mining algorithms may be found in [30]. Methods for finding frequent items in very large uncertain data sets or data streams may be found in [78].

4.4 Outlier Detection with Uncertain Data

The problem of outlier detection has also been extended to the case of uncertain data. In the case of the outlier detection problem, differing levels of uncertainty across different dimensions may affect the determination of the outliers in the underlying data. For example, consider the case in Fig. 3, in which the contours of uncertainty for two data points X and Y are illustrated in the form of elliptical shapes. The data point X seems to be further away from the overall data distribution as compared to the data point Y. However, the contours of uncertainty are such that the data point X has a greater *probability* of being drawn from the overall data distribution. Correspondingly, it is possible to define the concept of an outlier in terms of the probability that a given data point is drawn from a dense region of the overall data distribution.

In order to quantify the probability that a given uncertain data point is drawn from a dense region, we define the concept of η-probability. The η-probability of a data point $\overline{X_i}$ is defined as the probability that the uncertain data point lies in a region with (overall data) density at least η. Since the data point is uncertain, the η-probability may be computed by integrating the density of the data point along the contour of the intersection of the overall density function with threshold η. However, this can be computationally challenging from a numerical point of view. Therefore, the η-probability may be estimated with the use of sampling. The idea is to draw multiple samples from the data and compute the fraction of the samples over which the density threshold is specified. This can be used to define the concept of a (δ, η)-outlier.

Definition 4.2. *An uncertain data point $\overline{X_i}$ is defined to be a (δ, η)-outlier, if the η-probability of $\overline{X_i}$ in some subspace is less than δ.*

In order to determine the (δ, η)-outliers, the algorithm of [7] explores subspaces in the data in a bottom-up fashion and determines all those data points for which the condition of Definition 4.2 is satisfied. A variety of techniques for speeding up the algorithm using microclustering is also discussed in [7]. It has been shown in [7] that the approach is much more effective than deterministic algorithms for outlier detection.

4.5 General Approaches to Mining Uncertain Data

The techniques discussed in [17], [51], [52], and [63] are useful for working with a specific application such as clustering or classification. A different approach is to design an *intermediate representation* which can be used with a variety of data mining applications. A method of this nature has been proposed in [9]. In this case, a relaxed assumption is used that only the errors (in terms of standard deviation) of the records are known rather than the entire probability density function. This is a more realistic assumption of many scenarios, since it may often be possible to measure the standard deviation of an uncertain record, whereas the probability density function may be obtained only by more extensive theoretical modeling. In any case, if the pdf is available, one can still apply the method by using the derived standard deviation of the density function. It is assumed that the mean value of the ith record is denoted by $\overline{X_i}$ and the standard deviation by $\psi_i(\cdot)$.

In [9], the broad idea is to design an intermediate representation of the data which can then be leveraged in order to effectively perform the mining process. This intermediate representation is in the form of an *adjusted density estimate*. We refer to the density estimate as "adjusted," since the uncertainty is taken into account while creating the estimate. The density estimation $\overline{f}(x)$ based on N data points and is defined as follows:

$$\overline{f}(x) = (1/N) \cdot \sum_{i=1}^{N} \Big(1/\sqrt{2\pi} \cdot \big(h + \psi(\overline{X_i})\big)\Big) \cdot e^{\frac{-(x-\overline{X_i})^2}{\left(2\cdot\left(h^2+\psi(\overline{X_i})^2\right)\right)}}. \quad (3)$$

The above result assumes a Gaussian-kernel for each data point. This density estimate incorporates the error information, and can be utilized for a variety of data mining tasks as follows:

- Many density-based clustering algorithms [38] can be used in conjunction with this method. These clustering algorithms simply use a lower threshold on $\overline{f}(x)$ in order to isolate the dense (clustered) regions of the data. The technique can be used in order to isolate arbitrarily shaped clusters.
- A related approach [7] uses upper thresholds on the density estimate $\overline{f}(x)$ in order to isolate the sparse regions in the data. This can be used for outlier detection, by reporting those data points which lie in such sparse regions.
- In [9], it has been shown how to use this technique for classification. For a given test-instance, one determines the class-specific density at that point, and reports the class with the highest density. The density estimate can also be computed over different subspaces in order to further improve the accuracy.

In general, since density estimation encodes the summary behavior of the data, it is expected that such an approach can be used for any data mining problem which uses the aggregate data behavior in different spatial localities.

5 Summary

The field of uncertain data management has seen a revival in recent years because of new ways of collecting data which have resulted in the need for uncertain representations. This paper surveys the broad areas of work in this rapidly expanding field. We presented the important data mining and management techniques in this field along with the key representational issues in uncertain data management. While the field will continue to expand over time, it is hoped that this survey will provide an understanding of the foundational issues and a good starting point to practitioners and researchers in focussing on the important and emerging issues in this field.

Acknowledgments

The research of Charu C. Aggarwal was sponsored in part by the US Army Research laboratory and the UK Ministry of Defense under Agreement W911NF-06-3-0001. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies of the US Government, the US Army Research Laboratory, the UK Ministry of Defense, or the UK Government. The US and UK governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice hereon.

References

[1] S. Abiteboul, P. Kanellakis, and G. Grahne, "On the Representation and Querying of Sets of Possible Worlds," *Proc. ACM SIGMOD,* 1987.

[2] *Managing and Mining Uncertain Data,* C. Aggarwal, ed. Springer, 2009.

[3] P. Andritsos, A. Fuxman, and R.J. Miller, "Clean Answers over Dirty Databases: A Probabilistic Approach," *Proc. 22nd IEEE Int'l Conf. Data Eng. (ICDE),* 2006.

[4] L. Antova, C. Koch, and D. Olteanu, "From Complete to Incomplete Information and Back," *Proc. ACM SIGMOD,* 2007.

[5] L. Antova, C. Koch, and D. Olteanu, "$10^{(10^6)}$ Worlds and Beyond: Efficient Representation and Processing of Incomplete Information," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.

[6] L. Antova, T. Jansen, C. Koch, and D. Olteanu, "Fast and Simple Relational Processing of Uncertain Data," *Proc. 24th IEEE Int'l Conf. Data Eng. (ICDE),* 2008.

[7] C.C. Aggarwal and P.S. Yu, "Outlier Detection with Uncertain Data," *Proc. SIAM Int'l Conf. Data Mining (SDM),* 2008.

[8] C.C. Aggarwal, "On Unifying Privacy and Uncertain Data Models," *Proc. 24th IEEE Int'l Conf. Data Eng. (ICDE),* 2008.

[9] C.C. Aggarwal, "On Density Based Transformations for Uncertain Data Mining," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.

[10] C.C. Aggarwal and P.S. Yu, "A Framework for Clustering Uncertain Data Streams," *Proc. 24th IEEE Int'l Conf. Data Eng. (ICDE),* 2008.

[11] C.C. Aggarwal, J. Han, J. Wang, and P. Yu, "A Framework for Clustering Evolving Data Streams," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB),* 2003.

[12] M. Arenas, L. Bertossi, and J. Chomicki, "Consistent Query Answers in Inconsistent Databases," *Proc. 18th ACM Symp. Principles of Database Systems (PODS),* 1999.

[13] M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," *Proc. ACM SIGMOD,* 1999.

[14] D. Barbara, H. Garcia-Molina, and D. Porter, "The Management of Probabilistic Data," *IEEE Trans. Knowledge and Data Eng.,* vol. 4, no. 5, pp. 487-502, Oct. 1992.

[15] D. Bell, J. Guan, and S. Lee, "Generalized Union and Project Operations for Pooling Uncertain and Imprecise Information," *Data and Knowledge Eng.,* vol. 18, no. 2, 1996.

[16] O. Benjelloun, A. Das Sarma, A. Halevy, and J. Widom, "ULDBs: Databases with Uncertainty and Lineage," *Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB),* 2006.

[17] J. Bi and T. Zhang, "Support Vector Machines with Input Data Uncertainty," *Proc. Advances in Neural Information Processing Systems (NIPS),* 2004.

[18] C. Bohm, A. Pryakhin, and M. Schubert, "The Gauss-Tree: Efficient Object Identification of Probabilistic Feature Vectors," *Proc. 22nd IEEE Int'l Conf. Data Eng. (ICDE),* 2006.

[19] C. Bohm, P. Kunath, A. Pryakhin, and M. Schubert, "Querying Objects Modeled by Arbitrary Probability Distributions," *Proc. 10th Int'l Symp. Spatial and Temporal Databases (SSTD),* 2007.

[20] D. Burdick, P. Deshpande, T.S. Jayram, R. Ramakrishnan, and S. Vaithyanathan, "OLAP over Uncertain and Imprecise Data," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB '05),* pp. 970-981, 2005.

[21] D. Burdick, A. Doan, R. Ramakrishnan, and S. Vaithyanathan, "OLAP over Imprecise Data with Domain Constraints," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* 2007.

[22] R. Cavello and M. Pittarelli, "The Theory of Probabilistic Databases," *Proc. 13th Int'l Conf. Very Large Data Bases (VLDB),* 1987.

[23] A.L.P. Chen, J.-S. Chiu, and F.S.-C. Tseng, "Evaluating Aggregate Operations over Imprecise Data," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 2, pp. 273-284, Apr. 1996.

[24] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter, "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.

[25] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," *Proc. ACM SIGMOD,* 2003.

[26] R. Cheng, S. Singh, S. Prabhakar, R. Shah, J. Vitter, and Y. Xia, "Efficient Join Processing over Uncertain-Valued Attributes," *Proc. 15th ACM Int'l Conf. Information and Knowledge Management (CIKM),* 2006.

[27] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, J. Vitter, and Y. Xia, "Efficient Join Processing over Uncertain Data," Technical Report CSD TR# 05-004, Dept. of Computer Science, Purdue Univ., 2005.

[28] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Querying Imprecise Data in Moving Object Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 9, pp. 1112-1127, Sept. 2004.

[29] C.-K. Chui, B. Kao, and E. Hung, "Mining Frequent Itemsets from Uncertain Data," *Proc. 11th Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD),* 2007.

[30] C.-K. Chui and B. Kao, "A Decremental Approach for Mining Frequent Itemsets from Uncertain Data," *Proc. 12th Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD),* 2008.

[31] G. Cormode and A. McGregor, "Approximation Algorithms for Clustering Uncertain Data," *Proc. 27th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS),* 2008.

[32] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.

[33] N. Dalvi and D. Suciu, "Query Answering Using Statistics and Probabilistic Views," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB),* 2005.

[34] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working Models for Uncertain Data," *Proc. 22nd IEEE Int'l Conf. Data Eng. (ICDE),* 2006.

[35] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.

[36] D. Dey and S. Sarkar, "PSQL: A Query Language for Probabilistic Relational Data," *Knowledge and Data Eng.,* vol. 28, no. 1, pp. 107-120, 1998.

[37] X. Dong, A. Halevy, and C. Yu, "Data Integration with Uncertainty," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* 2007.

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD),* 1996.

[39] D. Florescu, D. Koller, and A. Levy, "Using Probabilistic Information in Data Integration," *Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB),* 1997.

[40] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning Probabilistic Relational Models," *Proc. 16th Int'l Joint Conf. Artificial Intelligence (IJCAI),* 1999.

[41] N. Fuhr and T. Rolleke, "A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems," *ACM Trans. Information Systems,* 1997.

[42] A. Fuxman, E. Fazli, and R.J. Miller, "Conquer: Efficient Management of Inconsistent Databases," *Proc. ACM SIGMOD,* 2005.

[43] *IEEE Data Eng. Bull.,* M. Garofalakis, and D. Suciu, eds., special issue on probabilistic data management, Mar. 2006.

[44] C. Genest and J. Zidek, "Combining Probability Distributions: A Critique and an Annotated Bibliography," *Statistical Science,* vol. 1, pp. 114-148, 1986.

[45] T. Green and V. Tannen, "Models for Incomplete and Probabilistic Information," *Data Eng. Bull.,* vol. 29, no. 1, 2006.

[46] G. Grahne, *The Problem of Incomplete Information in Relational Databases,* vol. 554, Lecture Notes in Computer Science, Springer-Verlag, 1991.

[47] E. Hung, "ProbSem: A Probabilistic Semistructured Databases Model," technical report, Univ. of Maryland, 2002.

[48] E. Hung, L. Getoor, and V. Subrahmanian, "PXML: A Probabilistic Semistructured Data Model and Algebra," *Proc. 19th IEEE Int'l Conf. Data Eng. (ICDE),* 2003.

[49] T. Imielinski and W. Lipski Jr, "Incomplete Information in Relational Databases," *J. ACM,* 1984.

[50] M. Keulen, A. Keijzer, and W. Alink, "A Probabilistic XML Approach to Data Integration," *Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE),* 2005.

[51] H.-P. Kriegel and M. Pfeifle, "Density-Based Clustering of Uncertain Data," *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery in Data Mining (KDD),* 2005.

[52] H.-P. Kriegel and M. Pfeifle, "Hierarchical Density Based Clustering of Uncertain Data," *Proc. Fifth IEEE Int'l Conf. Data Mining (ICDM),* 2005.

[53] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz, "Probabilistic Similarity Join over Uncertain Data," *Proc. 11th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* 2006.

[54] L.V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian, "ProbView: A Flexible Database System," *ACM Trans. Database Systems,* vol. 22, no. 3, pp. 419-469, 1997.

[55] S.K. Lee, "An Extended Relational Database Model for Uncertain and Imprecise Information," *Proc. 18th Int'l Conf. Very Large Data Bases (VLDB),* 1992.

[56] R. Little and D. Rubin, *Statistical Analysis with Missing Data.* John Wiley & Sons, 1987.

[57] V. Ljosa and A.K. Singh, "APLA: Indexing Arbitrary Probability Distributions," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.

[58] V. Ljosa and A.K. Singh, "Top-k Spatial Joins of Probabilistic Objects," *Proc. 24th IEEE Int'l Conf. Data Eng. (ICDE),* 2008.

[59] S.I. McClean, B.W. Scotney, and M. Shapcott, "Aggregation of Imprecise and Uncertain Information," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 6, pp. 902-912, Nov./Dec. 2001.

[60] A. Motro, "Accommodating Imprecision in Database Systems: Issues and Solutions," *SIGMOD Record,* vol. 19, no. 4, 1990.

[61] A. Motro, "Sources of Uncertainty, Inconsistency, and Imprecision in Database Systems," *Uncertainty Management in Information Systems,* pp. 9-34, 1996.

[62] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A.D. Sarma, R. Murthy, and T. Sugihara, "Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS," *Proc. Third Biennial Conf. Innovative Data Systems Research (CIDR),* 2007.

[63] W. Ngai, B. Kao, C. Chui, R. Cheng, M. Chau, and K.Y. Yip, "Efficient Clustering of Uncertain Data," *Proc. Sixth IEEE Int'l Conf. Data Mining (ICDM),* 2006.

[64] T. Pedersen, C. Jensen, and C. Dyreson, "Supporting Imprecision in Multidimensional Databases Using Granularities," *Proc. 11th Int'l Conf. Scientific and Statistical Database Management (SSDBM),* 1999.

[65] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic Skylines on Uncertain Data," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* 2007.
[66] A. Nierman and H. Jagadish, "ProTDB: Probabilistic Data in XML," *Proc. 28th Int'l Conf. Very Large Data Bases (VLDB),* 2002.
[67] D. Pfozer and C. Jensen, "Capturing the Uncertainty of Moving Object Representations," *Proc. Int'l Conf. Solid State Devices and Materials (SSDM),* 1999.
[68] H. Prade and C. Testemale, "Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries," *Information Sciences,* 1984.
[69] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch, "Indexing Uncertain Categorical Data," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.
[70] R. Ross, V. Subrahmanian, and J. Grant, "Aggregate Operators in Probabilistic Databases," *J. ACM,* vol. 52, no. 1, 2005.
[71] E. Rundensteiner and L. Bic, "Evaluating Aggregates in Probabilistic Databases," *Data and Knowledge Eng.,* vol. 7, no. 3, pp. 239-267, 1992.
[72] F. Sadri, "Modeling Uncertainty in Databases," *Proc. Seventh IEEE Int'l Conf. Data Eng. (ICDE),* 1991.
[73] P. Sen and A. Deshpande, "Representing and Querying Correlated Tuples in Probabilistic Databases," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.
[74] M. Soliman, I. Ilyas, and K.C.-C. Chang, "Top *k*-Query Processing in Uncertain Databases," *Proc. 23rd IEEE Int'l Conf. Data Eng. (ICDE),* 2007.
[75] B.W. Silverman, *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.
[76] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch, "Indexing Uncertain Categorical Data," *Proc. 22nd IEEE Int'l Conf. Data Eng. (ICDE),* 2006.
[77] Y. Tao, R. Cheng, X. Xiao, W. Ngai, B. Kao, and S. Prabhakar, "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB),* 2005.
[78] Q. Zhang, F. Li, and K. Yi, "Finding Frequent Items in Probabilistic Data," *Proc. ACM SIGMOD,* 2008.
[79] W. Zhao, A. Dekhtyar, and J. Goldsmith, "A Framework for Management of Semistructured Probabilistic Data," *J. Intelligent Information Systems,* pp. 1-39, 2004.
[80] E. Zimanyi, "Query Evaluation in Probabilistic Databases," *Theoretical Computer Science,* vol. 171, nos. 1/2, 1997.

Charu C. Aggarwal received the BTech degree in computer science from the Indian Institute of Technology Kanpur in 1993 and the PhD degree from Massachusetts Institute of Technology in 1996. He has been a research staff member at IBM T.J. Watson Research Center, Hawthorne, New York since then. He has published more than 100 papers in major conferences and journals in the database and data mining field. He has applied for or been granted more than 65 US and International patents, and has thrice been designated as a master inventor at IBM for the commercial value of his patents. He has been granted 17 invention achievement awards by IBM for his patents. His work on real-time bioterrorist threat detection in data streams won the IBM Corporate award for Environmental Excellence in 2003. He is a recipient of the IBM Outstanding Innovation Award in 2008 for his scientific contributions to privacy technology and a recipient of the IBM Research Division award for his scientific contributions to stream mining for the System S project. He has served on the program committee of most major database conferences, and was a program chair for the Data Mining and Knowledge Discovery Workshop 2003 and a program vice-chair for the SIAM Conference on Data Mining 2007, ICDM Conference 2007, and the WWW Conference, 2009. He served as an associate editor of the *IEEE Transaction on Data Engineering* from 2004 to 2008. He is an associate editor of the *ACM SIGKDD Explorations* and an action editor of the *Data Mining and Knowledge Discovery Journal*. He is a senior member of the IEEE and a life-member of the ACM.

Philip S. Yu received the BS degree in electrical engineering from National Taiwan University, the MS and PhD degrees in electrical engineering from Stanford University, and the MBA degree from New York University. He is a professor in the Department of Computer Science, University of Illinois, Chicago, and also holds the Wexler Chair in Information and Technology. He was manager of the Software Tools and Techniques group at the IBM T.J. Watson Research Center. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. He has published more than 520 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. He is an associate editor of the *ACM Transactions on the Internet Technology* and the *ACM Transactions on Knowledge Discovery from Data*. He is on the steering committee of the IEEE Conference on Data Mining and was a member of the IEEE Data Engineering steering committee. He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* (2001-2004), an editor, an advisory board member, and also a guest coeditor of the special issue on mining of databases. He had also served as an associate editor of *Knowledge and Information Systems*. In addition to serving as a program committee member on various conferences, he was the program chair or a cochair of the IEEE Workshop of Scalable Stream Processing Systems (SSPS '07), the IEEE Workshop on Mining Evolving and Streaming Data (2006), the 2006 joint conferences of the Eighth IEEE Conference on E-Commerce Technology (CEC '06) and the Third IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE '06), the 11th IEEE International Conference on Data Engineering, the Sixth Pacific Area Conference on Knowledge Discovery and Data Mining, the Ninth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, the Second IEEE International Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the Second IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the general chair or a cochair of the 2006 ACM Conference on Information and Knowledge Management, the 14th IEEE International Conference on Data Engineering, and the Second IEEE International Conference on Data Mining. He had received several IBM honors including two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 94th plateau of Invention Achievement Awards. He was an IBM Master Inventor. He received a Research Contributions Award from the IEEE International Conference on Data Mining in 2003 and also an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. He is a fellow of the ACM and the IEEE.