RC24394 (W0710-179) October 31, 2007
Computer Science

IBM Research Report

A Survey of Uncertain Data Algorithms and Applications

Charu C. Aggarwal, Philip S. Yu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A Survey of Uncertain Data Algorithms and Applications

Charu C. Aggarwal, Philip S. Yu
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{ charu, psyu}@us.ibm.com

Abstract

In recent years, a number of indirect data collection methodologies have lead to the proliferation of uncertain data. Such data points are often represented in the form of a probabilistic function, since the corresponding deterministic value is not known. This increases the challenge of mining and managing uncertain data, since the precise behavior of the underlying data is no longer known. In this paper, we provide a survey of uncertain data mining and management applications. In the field of uncertain data management, we will examine traditional 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 classification and clustering. We will also examine a general transform based technique for mining uncertain data. We discuss the models for uncertain data, and how they can be leveraged in a variety of applications. We discuss different methodologies to process and mine uncertain data in a variety of forms.

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 be only 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.

In uncertain data management, data records are represented by probability distributions rather than deterministic values. Therefore, a data record is represented by the corresponding parameters of a multi-dimensional probability distribution. 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 often uncertain. This is 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. Thus, each aggregated record is actually a probability distribution.
- In 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.
- 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.
- A very large class of uncertain data mining applications is that of *missing data* [43]. This particular subclass has been studied extensively in the literature. A missing data entry may either be modeled as a probabilistic distribution after imputation, or it may be modeled as a data entry with a pre-defined error.

- In many mobile applications, the trajectory of the objects may be unknown. In fact, many spatio-temporal applications are inherently uncertain, since the future behavior of the data can be predicted only approximately.

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 [22]. The second issue is that of adapting data management and mining applications to work with the uncertain data. Some examples of applications in which uncertain data mining techniques are relevant are as follows:

- Modeling of Uncertain Data: A key issue is the process of modeling the uncertain data for query processing.
- Query Processing: In this case, we wish to determine the probabilistic responses to a variety of queries. Such queries can be range queries, similarity queries, or trajectory queries in mobile objects.
- Uncertain Data Mining: The results of data mining applications are affected by the underlying uncertainty in the data. For example, the results of applications such as clustering and classification are skewed by the uncertainty in the underlying data. Therefore, it is critical to design data mining techniques which can take such uncertainty into account during the computations.

In the next sections, we will discuss the various issues involved in mining and managing uncertain data. We will discuss traditional database management problems such as OLAP, query processing, join processing, indexing and selectivity estimation. We will also discuss techniques for a number of data mining problems such as clustering and classification. We will also discuss a general transform-based technique for mining and managing uncertain data applications.

This paper is organized as follows. In the next section, 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, 33, 32, 36, 57]. We would like to distinguish between incomplete databases and probabilistic data, since the latter is a more specific definition which creates database models with crisp probabilistic quantifications. A database that provides incomplete information consists of *a set of possible instances*. A general model is the "possible worlds model" discussed in [1]. The "possible worlds" semantics assumes that the probability of presence of a tuple in the databases affects the probability of presence or absence of another tuple and vice-versa. Thus, we often refer to each possible state of the entire database as a possible world. Such restrictions are often represented by rules which enforce the relationships between the behavior of the different tuples. Since there are often an exponential number of possibilities in representing the database, such an representational approach is often difficult to use from a practical or application point of view. Therefore, a key problem in uncertain databases is to construct a formalism which can be interpreted as a "possible worlds model" but is also friendly to use of a variety of data management and mining applications. One such complete formalism is the intensional database in which we work with the underlying *events* which create the uncertainty. Thus, the underlying events are uncertain in nature, and the uncertainty in the attribute values are derived from these events. From an application point of view, such an approach can sometimes be challenging, since it requires us to work with combinations of events which can be very large. In a practical sense, we often work with incomplete formalisms. One such commonly used formalism is one in which each tuple is independent, and we focus on the uncertainty only at the attribute level. Such a formalism is incomplete, since we are not capturing all possible worlds with the use of independent

tuples. Furthermore, one needs to be careful in the application of such a formalism, since it may result in inconsistency. For example, in a spatio-temporal database, the same object cannot 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.

In many cases, the information about the set of possible instances is quantified with the use of probability distributions. Thus, the set of possible values taken on by a variable can be quantified with the use of a probability distribution. Such databases are referred to as probabilistic databases. Therefore, a probabilistic databases is defined [32] as follows:

Definition 2.1 *A probabilistic-information database is a finite probability space whose outcomes are all the set of possible pairs* $(\mathcal{X}, p)$*, such that* $\sum_{I \in \mathcal{X}} p(I) = 1$.

We note that the above representation uses a discrete representation of the uncertainty. The direct specification of such databases is unrealistic, since we would need an exponential number of instances in order to represent the table. Therefore, the natural solution is to construct simplified probabilistic tables which can be easily used for data mining and data management purposes.

Probabilistic ?-tables [28, 41] are a simple way of representing probabilistic data. In this case, we model 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 not be present in 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 we know 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 [41] is a kind of or-set table. The only significant difference is that the ProbView model uses confidence values instead of probabilities.

The **probabilistic c-table** is closely related to the probabilistic or-set tables. In this case, the probabilistic c-table consists of a c table T together with a finite probability space $\text{dom}(x)$ for each variable x that occurs in T.

A number of interesting properties of such databases are discussed in [28, 41, 64]. We further note that the above discussion is useful for the point of database formalism. Most data mining or query processing applications work with attribute-uncertainty models in each attribute value is represented by a discrete or continuous probability distribution, depending upon the data domain. Some applications [3] 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, which may not always conform to a database-centric view of things. 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:** This is the most general case, and reflects the behavior of the "possible worlds" model. 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 which correspond to mutual exclusivity of certain tuples in the database. **(2) Attribute Level Uncertainty:** This is a more restricted model in which the number of tuples and their modeling have already been determined. The uncertainty of the individual attributes are modeled by a probability density function, or other statistical parameters such as the variance. We note that attribute level uncertainty is inherently easier to handle than existential uncertainty and may be sufficient for many practical problems.

Recently, uncertain data models have also been extended to semi-structured and XML data. Some of the earliest work on probabilistic semi-structured data may be found in [51]. XML

data poses numerous unique challenges. Since XML is structured, it is important to assign and interpret probabilities naturally for structurally related elements. 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, we associate probabilities 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. We note that in the most general case, this can lead to issues of computational intractability. The approach in [51] is to model some classes of dependence (eg. mutual exclusion) which are useful and efficient to model. The work in [51] also designs techniques for a restricted class of queries on the data. Another interesting approach to probabilistic XML data construction has been discussed in [37]. In this technique, we construct probabilistic XML trees 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. (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, that assigns node kinds. Furthermore, we use a *prob* function which assigns probabilities to nodes. The query evaluation technique enumerates all possible worlds in a recursive manner. These enumerated worlds are then used in order to respond to the query. Other related work on XML data representation and modeling may be found in [26, 63].

A number of recent projects have designed uncertain databases around specific application requirements. For example, the *Conquer* project [2, 29] 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 [16, 19] 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 [10, 47, 22] at Stanford University. This work introduces the concept of *ULDB* (Uncertainty-Lineage Database), 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 we keep track of the *sources* from which the data was acquired and also keep 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.

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. One of the earliest techniques for adding probabilistic information into query evaluation was discussed in [28]. This model is a generalization of the standard relational model. In this model, probabilistic relations are treated as generaliza-

tions 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. This approach is referred to as that of *extensional semantics*. Another approach which is based on Dempster-Shafer theory was presented in [42]. 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. An example discussed in [28] considers a query in which we are asking for records which belong to either the database DB or IR. This query can be expressed as $\mu \in (DB - IR)) \cup (\mu \in (IR - DB))$. Thus, a query plan which computes the probabilities of the two subexpressions $\mu \in (DB - IR)$ and $\mu \in (IR - DB)$ would be incorrect if it assumes independence of the expressions corresponding to $\mu \in (DB - IR)$ and $\mu \in (IR - DB)$.

Probabilistic reasoning which is based on intensional semantics overcomes this problem with the use of tree-like structures of inferences. In the example discussed above, it is known that the expressions are disjoint, and are therefore the corresponding probabilities of $(DB - IR)$ and $(IR - DB)$ can be added. In general, extensional semantics yields results which are consistent with probability theory for tree-like (disjoint) structures of inferences. Note that if we had simply represented the expression as $(\mu \in DB) \cup (\mu \in IR)$, there would be no way to combine the probabilities from the resulting subexpressions effectively. Therefore, the key is to develop a probabilistic relational algebra with intensional semantics which always yields correct results. An immediate consequence of the use of intensional semantics in query planning is that there is exponential increase in the computational requirements with complexity of queries. It has been shown in [20] that certain queries have $\#P$-complete data complexity under probabilistic semantics.

Clearly, a method is required to reduce the query evaluation complexity in relational databases. Therefore, a technique proposed in [20] 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. Howeber many queries which occur in practice do admit a correct extensional plan. According to [20], 8 out of 10 TPC/H queries 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 more expensive sampling based Monte-Carlo algorithm is designed which is more expensive, but can guarantee arbitrarily small errors. In addition, the technique in [20] also extends the solution to the case of *uncertain predicates* on deterministic data. A different imprecision model is discussed in [21] in which we use only the statistics on the data and explicit probabilities at the data sources. It is shown in [21] that such imprecisions can be modeled by a certain kind of probabilistic databases with complex tuples correlations. The method in [20] is then used in order to rewrite the queries for effective query resolution. We note that the work in [20] assumes tuple independence which is often not the case for 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 [6].

While the work in [20] assumes tuple independence, this may not always be the case in many practical applications. For example, data from sensors [23] may be highly correlated both in terms of space and time. Furthermore, even if we assume that the tuples are independent many intermediate results of queries may contain complex correlations. For examples, even the simple operator of performing a join is not closed under tuple independence. In [58], a technique has been proposed on querying correlated tuples with the use of statistical modeling techniques. The method in [58] constructs a uniform framework which expresses and uncertainties and dependencies through the use of joint probability distributions. The query evaluation problem on probabilistic database is cast as an inference problem in probabilistic graphical models [27]. 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.

A related query is the top-k query in which we wish 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 [59] 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 re-definition of the semantics for the query itself. For example, consider the case of a radar-controlled traffic system [59], 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, we may be after the top-k queries in a most probable worlds. 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^{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 [59] 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 [27]. The work in [59] also creates a framework for integrating space navigation algorithms and data access methods leveraging existing DBMS technologies. One key result presented in [59] 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.

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 we attempt 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 [14, 44, 55, 56]. 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 [49], which considers the semantics of aggregate queries in an uncertain environment. However this technique does not consider uncertainty or investigate criteria which shed light on appropriate query semantics.

In [12], a crisp set of criteria has been identified in order to handle ambiguity. This criteria are then used in a principled manner in order to arrive at appropriate semantics for queries. The criteria which are identified in [12] are as follows:

- **Consistency:** This criteria 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.
- **Faithfulness:** This captures the notion that more precise data should lead to more accurate results.

- **Correlation-Preservation:** This requires that the statistical properties of the data should not be affected by the allocation of ambiguous data records.

In order to model the uncertainty, the work in [12] 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 [12] 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* [31]. 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 [12] 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 [12].

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, we attempt 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. We retain all the objects whose probability of membership lies above a certain threshold.

- **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 non-zero probability that a given object is the nearest neighbor to the target.

- **Aggregate Queries:** In such queries, we attempt 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 we also have to account for the interplay of different objects.

In [16], a broad classification of the queries has been provided. 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 (eg. average, max). Another property which can be used to classify queries is whether or not aggregation is involved. In [16], broad classes of query processing techniques have been discussed for each of these different kinds of queries.

A related application for indexing and querying imprecise data is that of moving object environments [19]. 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 [19], 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 [19] also tackles the *probabilistic nearest neighbor query*. In the probabilistic nearest neighbor query, we 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, we simply integrate the probability density function 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, we 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 we can easily 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, we eliminate any object for which the shortest distance to the target q is larger than f.

- **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. We only have to 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, we calculate 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.

A related work in [15] proposes the concept of *probabilistic threshold queries*. In such queries, we attempt 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* $[a, b]$ *where* $a, b \in \mathcal{R}$ *and* $a \leq b$*, a probabilistic threshold query returns a set of tuples* T_i*, such that the probability* $T_i \cdot a$ *is inside* $[a, b]$*, denoted by* p_i*, is greater than or equal to* p*, where* $0 \leq p \leq 1$.

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 [15] 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, we will need 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 [15] is referred to as *Probability Threshold Indexing*. This index structure is essentially based on a modification of a 1-dimensional 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 pre-order 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 $M_j\dot{l}b(x)$) and right-x-bound (denoted by $M_j\dot{r}b(x)$). Every interval $[L_i, R_i]$ 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 of being right of the right-x-bound.*

We note that this kind of bound is a generalization of the concept of the Minimum Bounding Rectangle. 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 $b < M_j\dot{l}b(x)$ or $a > M_j\dot{r}b(x)$.
- $p \geq x$

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 [15] 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.

A method for indexing uncertain categorical data has been discussed in [61]. The definition used in [61] 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:

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

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 [61] are as follows:

- **Probabilistic Equality Query (PEQ):** Given an Uncertain Discrete Attribute (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 [61] 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, we store a list of the tuple-ids which

have a non-zero probability of taking on that particular value. Along with each tuple-id, we also store this probability value. The inner lists containing the tuple-ds 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. We only need 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 probabilistic equality threshold queries. 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, we retain only those which qualify more than the threshold. A variety of other pruning techniques can be used in order to improve the efficiency of the different queries. The different techniques discussed in [61] 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 [61].

In the next section, we will discuss the probabilistic distribution R-Tree which is an alternative for indexing uncertain discrete attributes. The broad approach is to *index the vector of probability values of the possible attribute values*. Thus, if we have N possible probability values then, we create data points 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 we 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 [61]. 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.

An interesting technique discussed in [62] 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 [62], an index structure called the U-Tree has been proposed, which can handle such kinds of queries.

3.3 Join Processing on Uncertain Data

The problem of join queries over uncertain data has been studied in [17]. In this case, 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. We use the term *Probabilistic Join Queries* to describe these types of joins over uncertain data. 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, we define a parameter called resolution, and define a pair of attributes to be equal to one another, if one attribute value is within c of another. Correspondingly, the probability can be calculated as follows:

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

For the case of the $>$ and $<$ operators, we do not need to use the resolution, can we can 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 we report all tuples whose join probability is above the user-defined threshold. 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, we can 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 [18]. The above two inequalities can be used for those join tuples which satisfy the pre-conditions described above. Depending upon the direction of the inequality, we 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, we 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 [17], and is used to augment the nodes of the underlying index structure. For more details, we refer the reader to [17].

A second kind of join which is studied in the literature [40] is the *similarity join*. This

similarity is measured by the distance between the two feature vectors. The most popular distance range join which is often studied is the distance-range join. In the distance range join, we perform the join between two records, is the distance between the two does not exceed a user-defined parameter ϵ. One possibility is to compute the expected distance between two relations, and perform the join, if this expected distance is less than the parameter ϵ. This is because the expected distances are often skewed by the behavior of the tail end behavior of the probability functions, and different joins which have similar probability of lying within the range of ϵ may be treated inconsistently. Therefore, it has been proposed in [40] to assign to each object pair, a probability values which reflects the likelihood that the object pair belongs to the join result set. Only those join pairs which have non-zero probability of belonging to the join result set are returned.

3.4 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 [50] provides a first approach to this problem. The problem of skyline computation is used in multi-criteria 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 skyline concept is defined as follows:

Definition 3.6 *Consider a set of objects with multiple criteria to be optimized. An object U is said to be in the skyline, if there is no other object V which is better than U in more than one aspect.*

Clearly all players which 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 [50]. 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 3.7 *Given a probability threshold p ($0 \leq p \leq 1$), the p-skyline is the st of uncertain objects, such that each of them takes of probability of 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 [50], 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 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 in [50].

The top-down method starts with the whole set of instances of an uncertain data object. The skyline probability is bounded by using the maximum and minimum corners of the minimum bounding box of the object. This is used for the purpose of pruning. In order to improve the bounds, the instances are recursively partitioned into subsets, and the skyline probability of each subset can can be bounded using its minimum bounding box. The process of continued until the skyline membership of each object is conclusively determined.

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 [38] 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. We formally define the distance distribution function as follows:

Definition 4.1 *Let $\overline{X}$ and $\overline{Y}$ be two uncertain record, and let $p(\overline{X}, \overline{Y})$ represent the distance density function between these objects. Then, the probability that the distance lies within the range (a, b) is given by the following relationship:*

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

Based on this technique and the distance density function, the method in [38] 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 which lies in a region on 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 [25]. Therefore, the fuzzy version of the DBSCAN algorithm (referred to as FDBSCAN) works in a similar way as the DBSCAN algorithm, except that we add the additional constraint that the computed reachability probability must be greater than 0.5.

Another related technique discussed in [39] is that of hierarchical density based clustering. An effective (deterministic) density based hierarchical clustering algorithm is OPTICS [7]. We note that the core idea in OPTICS is defined based on the concept of *reachability distance* between data points. Correspondingly, the work in [38] defines the concept of fuzzy reachability distance with the help of the distance distribution function discussed above. This is then used in order to construct the FOPTICS algorithm which works in an exactly analogous way to the OPTICS algorithm, except that we use the fuzzy reachability distance.

Finally, a technique in [48] 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 [48] 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 we have quantified an upper bound on the minimum distance of a particular data point to some cluster representative, we do not need 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 [38, 48] were developed for the case of static data. Recently, the technique has also been extended to the case of data streams. In [4], it has been shown how to extend the method to the case of data streams. In order to do so, we use the micro-clustering concept which was developed in [5]. In order to incorporate the uncertainty into the clustering process, we need a method to incorporate and leverage the error information into the micro-clustering statistics and algorithms. As discussed earlier, it is assumed that the data stream consists of a set of multi-dimensional records $\overline{X}_1 \ldots \overline{X}_k \ldots$ arriving at time stamps $T_1 \ldots T_k \ldots$. Each $\overline{X_i}$ is a multi-dimensional record containing d dimensions which are denoted by $\overline{X_i} = (x_i^1 \ldots x_i^d)$. In order to apply the micro-clustering method to the uncertain data mining problem, we need to also define the concept of error-based micro-clusters. We define such micro-clusters as follows:

Definition 4.2 *An uncertain micro-cluster for a set of d-dimensional points $X_{i_1} \ldots X_{i_n}$ with time stamps $T_{i_1} \ldots T_{i_n}$ and error vectors $\overline{\psi(\overline{X_{i_1}})} \ldots \overline{\psi(\overline{X_{i_n}})}$ is defined as the $(3 \cdot d + 2)$ tuple $(\overline{CF2^x}(\mathcal{C}), \overline{EF2^x}(\mathcal{C}), \overline{CF1^x}(\mathcal{C}), t(\mathcal{C}), n(\mathcal{C}))$, wherein $\overline{CF2^x}(\mathcal{C})$, $\overline{EF2^x}(\mathcal{C})$, and $\overline{CF1^x}(\mathcal{C})$ each correspond to a vector of d entries. The entries in $\overline{EF2^x}(\mathcal{C})$ correspond to the error-based entries. The definition of each of these entries is as follows:*

• *For each dimension, the sum of the squares of the data values is maintained in $\overline{CF2^x}(\mathcal{C})$. Thus, $\overline{CF2^x}(\mathcal{C})$ contains d values. The p-th entry of $\overline{CF2^x}(\mathcal{C})$ is equal to $\sum_{j=1}^{n} (x_{i_j}^p)^2$. This corresponds to the second moment of the data values along the p-th dimension.*

• *For each dimension, the sum of the squares of the errors in the data values is maintained in $\overline{EF2^x}(\mathcal{C})$. Thus, $\overline{EF2^x}(\mathcal{C})$ contains d values. The p-th entry of $\overline{EF2^x}(\mathcal{C})$ is equal to*

$\sum_{j=1}^{n} \psi_p(X_{i_j})^2$. *This corresponds to the sum of squares of the errors in the records along the p-th dimension.*

• *For each dimension, the sum of the data values is maintained in* $\overline{CF1^x}(\mathcal{C})$. *Thus,* $\overline{CF1^x}(\mathcal{C})$ *contains d values. The p-th entry of* $\overline{CF1^x}(\mathcal{C})$ *is equal to* $\sum_{j=1}^{n} x_{i_j}^p$. *This corresponds to the first moment of the values along the p-th dimension.*

• *The number of points in the data is maintained in* $n(\mathcal{C})$.

• *The time stamp of the last update to the micro-cluster is maintained in* $t(\mathcal{C})$.

We note that the uncertain definition of micro-clusters differs from the deterministic definition, since we have added an additional d values corresponding to the error information in the records. We will refer to the uncertain micro-cluster for a set of points $\mathcal{C}$ by $\overline{ECF}(\mathcal{C})$. We note that error based micro-clusters maintain the important *additive property* [5] which is critical to its use in the clustering process. We restate the additive property as follows:

Property 4.1 *Let* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *be two sets of points. Then all non-temporal components of the error-based cluster feature vector* $\overline{ECF(\mathcal{C}_1 \cup \mathcal{C}_2)}$ *are given by the sum of* $\overline{ECF(\mathcal{C}_1)}$ *and* $\overline{ECF(\mathcal{C}_2)}$.

The additive property follows from the fact that the statistics in the individual micro-clusters are expressed as a separable additive sum of the statistics over individual data points. We note that the single temporal component $t(\mathcal{C}_1 \cup \mathcal{C}_2)$ is given by $\max\{t(\mathcal{C}_1), t(\mathcal{C}_2)\}$. We note that the additive property is an important one, since it ensures that it is easy to keep track of the cluster statistics as new data points arrive. Next we will discuss the process of uncertain micro-clustering.

The $UMicro$ algorithm works using an iterative approach which maintains a number of micro-cluster centroids around which the clusters are built. It is assumed that one of the inputs to the algorithm is n_{micro}, which is the number of micro-clusters to be constructed. The algorithm starts off with a number of null clusters and initially creates new singleton clusters, to which new points are added subsequently. For any incoming data point, the closest cluster centroid is determined. The closest cluster centroid is determined by using

the *expected distance* of the uncertain data point to the *uncertain micro-clusters*. The process of expected distance computation for the closest centroid is tricky, and will be subsequently discussed. Furthermore, for the incoming data point, it is determined whether it lies within a *critical uncertainty boundary* of the micro-cluster. If it lies within this critical uncertainty boundary, then the data point is added to the micro-cluster, otherwise a new micro-cluster needs to be created containing the singleton data point. In order to create a new micro-cluster, it must either be added to the current set of micro-clusters, or it needs to replace one of the older micro-clusters. In the initial stages of the algorithm, the current number of micro-clusters is less than n_{micro}. If this is the case, then the new data point is added to the current set of micro-clusters as a separate micro-cluster with a singleton point in it. Otherwise, the new data point needs to replace one of the older micro-clusters. For this purpose, we always replace the least recently updated micro-cluster from the data set. This information is available from the temporal time stamp in the different micro-clusters. A key step in this is the computation of the expected similarity between data points and micro-cluster centroids. It has been shown in [4] that this expected computation can be performed by using the information encoded in the summary statistics. More details of this computation process may be found in [4].

4.2 Classification of Uncertain Data

A closely related problem is that of classification of uncertain data in which we attempt to classify a test instance into one particular label from a set of class labels. In [11], 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 [11] 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.

4.3 General Approaches to Mining Uncertain Data

The techniques discussed in [11, 38, 39, 48] 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 [3]. In this case, a relaxed assumption is used that we know only the errors (in terms of standard deviation) of the records rather than the entire probability density function. This is a more realistic assumption 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, we 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 [3], 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 a *adjusted density estimate*. We refer to the density estimate as "adjusted", since we are taking the uncertainty into account while creating the estimate. The approach of kernel density estimation is used in order to construct the adjusted density estimates of the data. The idea in kernel density estimation [60] is to provide a continuous estimate of the density of the data at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K'_h(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h which determines the level of smoothing created by the function. The kernel estimation $\overline{f}(x)$ based on N data points and kernel function $K'_h(\cdot)$ is defined as follows:

$$\overline{f}(x) = (1/N) \cdot \sum_{i=1}^{N} K'_h(x - \overline{X_i}) \tag{3}$$

Thus, each discrete point $\overline{X_i}$ in the data set is replaced by a continuous function $K'_h(\cdot)$ which peaks at X_i and has a variance which is determined by the smoothing parameter h. An

example of such a distribution would be a gaussian kernel with width h.

$$K'_h(x - \overline{X_i}) = (1/\sqrt{2\pi} \cdot h) \cdot e^{-(x-\overline{X_i})^2/(2h^2)} \tag{4}$$

The overall effect of kernel density estimation is to convert the (discrete) data set into a continuous density estimate by replacing each data point with a smoothed "bump", whose width is determined by h. The density distribution at a given coordinate is equal to the sum of the contributions of all the "bumps" represented by the data points. The result is a continuous distribution in which the random artifacts are suppressed and the density behavior provides a global overview of the dense as well as sparsely populated regions of the data. The estimation error depends upon the kernel width h which is chosen in a data driven manner. A widely used rule for approximating the bandwidth is the Silverman approximation rule [60] for which h may be chosen to be $1.06 \cdot \sigma \cdot N^{-1/5}$, where σ^2 is the variance of the N data points. It has been shown [60] that for most smooth functions $K'_h(\cdot)$, when the number of data points goes to infinity, the estimator $\overline{f}(x)$ asymptotically converges to the true density function $f(x)$, provided that the width h is chosen using the above relationship. For the d-dimensional case, the kernel function is chosen to be the product of d identical kernels $K_i(\cdot)$, each with its own smoothing parameter h_i.

The presence of errors can change the density estimates because of the different levels of error in different entries or fields. For example a data point or field with very large error should affect the density estimation of its locality to a smaller extent than one which contains small errors. When estimations of such errors are available, it is desirable to incorporate them into the estimation process. A direct way of doing so is to adapt the kernel function so that the measurement errors are taken into account during the calculation of the density distribution. Correspondingly, we define the following error-based kernel $Q'_h(x - X_i, \psi(\overline{X_i})$ function, which depends both upon the error as well as the values of the underlying data points.

$$Q'_h(x - X_i, \psi(\overline{X_i}) = (1/\sqrt{2\pi} \cdot (h + \psi(\overline{X_i}))) \cdot e^{\frac{-(x-X_i)^2}{(2\cdot(h^2+\psi(\overline{X_i})^2))}} \tag{5}$$

The overall effect of changing the kernel function is that the width of the bandwidth along the corresponding dimension is increased by $\psi(\overline{X_i})$. The intuition behind this choice of modifying the kernel is to adjust the contributions of the kernel function for a point depending upon

its (error-based) probability density. Note that in the limiting case, when there are a large number of data points N, the value of the bandwidth h goes to zero, and this kernel function has a gaussian distribution with standard error exactly equal to the standard error of the data point. Conversely, the error-based kernel function converges to the standard kernel function when the value of the error $\psi(\overline{X_i})$ is 0. Therefore, in these boundary cases, the direct error-based generalization of the kernel function has a probability distribution with the same standard error as the data point. It is also clear that in the limiting case of a large number of data points, (when the bandwidth h tends to zero by the Silverman rule) the kernel function reflects the errors in each data point accurately. As in the previous case, the error-based density at a given data point is defined as the sum of the error-based kernels over different data points. Therefore, we define the error based density $\overline{f^Q}$ at point x as follows:

$$\overline{f^Q}(x, \psi(\overline{X_i})) = (1/N) \cdot \sum_{i=1}^{N} Q'_h(x - \overline{X_i}, \psi(\overline{X_i})) \tag{6}$$

As in the previous case, we can easily generalize the definition to the multi-dimensional case. Specifically, the error for the jth dimension is denoted by $\psi_j(\overline{X_i})$. The overall kernel function is defined as the product of the kernel function for the different dimensions. In [3], it has been shown how to generalize this approach to the case of data streams. The key idea is to use micro-clusters instead of individual data points in order to perform the kernel function computation. The kernel function is then expressed in terms of micro-cluster statistics rather than individual data points. We further note that an improved micro-clustering algorithm was proposed in [4]. It is possible to substitute this improved algorithm over the approach presented in [3].

5 Summary

The problem of uncertain data management is a new area of research which is rapidly expanding in a variety of directions. A variety of methods have been proposed for indexing and query processing of uncertain data. The problem is also applicable to the case of OLAP based aggregation queries, in which we wish to find aggregate responses to a variety of projection queries. We discussed a number of query processing and indexing techniques in

the data management domain. In recent years, a number of data mining techniques have also been developed for the case of uncertain data. A number of techniques have been developed for the problems of clustering and classification. In [3], a broad template has been designed for constructing intermediate density based representations, which can then be leveraged for mining uncertain data.

References

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

[2] P. Andritsos, A. Fuxman, R. J. Miller. Clean Answers over Dirty Databases: A probabilistic Approach. *ICDE Conference*, 2006.

[3] C. C. Aggarwal. On Density Based Transformations for Uncertain Data Mining. *ICDE Conference*, 2007.

[4] C. C. Aggarwal, P. S. Yu. A Framework for Clustering Uncertain Data Streams. *ICDE Conference*, 2008.

[5] C. C. Aggarwal, J. Han, J. Wang, P. Yu. A Framework for Clustering Evolving Data Streams. *VLDB Conference*, 2003.

[6] M. Arenas, L. Bertossi, J. Chomicki. Consistent Query Answers in Inconsistent Databases. *ACM PODS Conference*, 1999.

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

[8] D. Barbara, H. Garcia-Molina, D. Porter: The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5), pp. 487–502, 1992.

[9] D. Bell, J. Guan, S. Lee: Generalized Union and Project Operations for Pooling Uncertain and Imprecise Information. *Data and Knowledge Engineering*, 18(2), 1996.

[10] O. Benjelloun, A. Das Sarma, A. Halevy, J. Widom: ULDBs: Databases with uncertainty and lineage. *VLDB Conference*, 2006.

[11] J. Bi, T. Zhang: Support Vector Machines with Input Data Uncertainty. *NIPS Conference*, 2004.

[12] D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, S. Vaithyanathan. OLAP Over Uncertain and Imprecise Data, *VLDB Conference* pp. 970–981, 2005.

[13] R. Cavello, M. Pittarelli. The Theory of Probabilistic Databases. *VLDB Conference*, 1987.

[14] A. L. P. Chen, J.-S. Chiu, F. S.-C. Tseng. Evaluating Aggregate Operations over Imprecise Data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2), 273–284, 1996.

[15] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, J. Vitter: Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data. *VLDB Conference*, 2004.

[16] R. Cheng, D. Kalashnikov, S. Prabhakar: Evaluating Probabilistic Queries over Imprecise Data. *SIGMOD Conference*, 2003.

[17] R. Cheng, S. Singh, S. Prabhakar, R. Shah, J. Vitter, Y. Xia: Efficient Join Processing over Uncertain-valued Attributes. *CIKM Conference*, 2006.

[18] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, J. Vitter, Y. Xia: Efficient Join Processing over Uncertain Data. *Technical Report CSD TR# 05-004, Dept of CS, Purdue University*, 2005.

[19] R. Cheng, D. Kalashnikov, S. Prabhakar. Querying imprecise data in moving object environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1112–1127, 2004.

[20] N. Dalvi, D. Suciu: Efficient Query Evaluation on Probabilistic Databases. *VLDB Conference*, 2004.

[21] N. Dalvi, D. Suciu: Query Answering using Statistics and Probabilistic Views. *VLDB Conference*, 2005.

[22] A. Das Sarma, O. Benjelloun, A. Halevy, J. Widom: Working Models for Uncertain Data. *ICDE Conference*, 2006.

[23] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong: Model-driven data acquisition in sensor networks. *VLDB Conference*, 2004.

[24] D. Dey, S. Sarkar: PSQL: A Query Language for Probabilistic Relational Data. *Knowledge and Data Engineering*, 28(1), 107–120, 1998.

[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu: A Density Based Algorithm for Discovcering Clusters in Large Spatial Databases with Noise. *KDD Conference*, 1996.

[26] D. Florescu, D. Koller, A. Levy. Using probabilistic information in data integration. *VLDB Conference*, 1997.

[27] N. Friedman, L. Getoor, D. Koller, A. Pfeffer: Learning Probabilistic Relational Models, *IJCAI*, 1999.

[28] N. Fuhr, T. Rolleke: A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 1997.

[29] A. Fuxman, E. Fazli, R. J. Miller. Conquer: Efficient Management of Inconsistent Databases. *SIGMOD Conference*, 2005.

[30] M. Garofalakis, and D. Suciu (editors): *IEEE Data Engineering Bulletin Special Issue on Probabilistic Data Management*, March 2006.

[31] C. Genest, J. Zidek: Combining Probability Distributions: A Critique and an Annotated Bibliography, *Statistical Science*, 1:114–148, 1986.

[32] T. Green, V. Tannen: Models for Incomplete and Probabilistic Information. *Data Engineering Bulletin*, 29(1), 2006.

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

[34] E. Hung. ProbSem: A probabilistic semistructured databases Model. *Technical report, University of Maryland*, 2002.

[35] E. Hung, L. Getoor, V. Subrahmanian: PXML: A probabilistic semistructured data model and algebra. *ICDE Conference*, 2003.

[36] T. Imielinski, W. Lipski Jr.: Incomplete Information in Relational Databases. *Journal of the ACM*, 1984.

[37] M. Keulen, A. Keijzer, W. Alink: A Probabilistic XML Approach to Data Integration. *ICDE Conference*, 2005.

[38] H.-P. Kriegel, M. Pfeifle: Density-based clustering of uncertain data. *KDD Conference*, 2005.

[39] H.-P. Kriegel, M. Pfeifle: Hierarchical Density Based Clustering of Uncertain Data. *ICDM Conference*, 2005.

[40] H.-P. Kriegel, P. Kunath, M Pfeifle, M. Renz: Probabilistic Similarity Join over Uncertain Data. *DASFAA Conference*, 2006.

[41] L. V. S Lakshmanan, N. Leone, R. Ross, V. S. Subrahmanian. ProbView: A Flexible Database System. *ACM TODS*, 22(3):419–469, 1997.

[42] S. K. Lee: An extended An extended relational database model for uncertain and imprecise information. *VLDB Conference*, 1992.

[43] R. Little, D. Rubin. Statistical Analysis with Mssing Data, *John Wiley and Sons*, 1987.

[44] S. I. McClean, B. W. Scotney, M. Shapcott. Aggregation of Imprecise and Uncertain Information. *IEEE TKDE*, 13(6):902–912, 2001.

[45] A. Motro. Accomodating Imprecision in Database Systems: Issues and Solutions, *SIGMOD Record*, 19(4), 1990.

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

[47] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, T. Sugihara: Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS. *CIDR Conference*, 2007.

[48] W. Ngai, B. Kao, C. Chui, R. Cheng, M. Chau, K. Y. Yip. Efficient Clustering of Uncertain Data, *ICDM Conference*, 2006.

[49] T. Pedersen, C. Jensen, C. Dyreson. Supporting Imprecision in Multidimensional Databases using Granularities. *SSDBM Conference*, 1999.

[50] J. Pei, B. Jiang, X. Lin, Y. Yuan. Probabilistic Skylines on Uncertain Data. *VLDB Conference*, 2007.

[51] A. Nierman, H. Jagadish: ProTDB: Probabilistic Data in XML. *VLDB Conference*, 2002.

[52] D. Pfozer, C. Jensen. Capturing the uncertainty of moving object representations. *SSDM Conference*, 1999.

[53] H. Prade, C. Testemale: Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and vague Queries. *Information Sciences*, 1984.

[54] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, S. Hambrusch: Indexing Uncertain Categorical Data. *ICDE Conference*, 2007.

[55] R. Ross, V. Subrahmanian, J. Grant: Aggregate Operators in Probabilistic Databases, *Journal of the ACM*, 52(1), 2005.

[56] E. Rundensteiner, L. Bic: Evaluating Aggregates in Probabilistic Databases, *Data and Knowledge Engineering*, 7(3), 239–267, 1992.

[57] F. Sadri: Modeling Uncertainty in Databases. *ICDE Conference*, 1991.

[58] P. Sen, A. Deshpande: Representing and Querying Correlated Tuples in Probabilistic Databases. *ICDE Conference*, 2007.

[59] M. Soliman, I. Ilyas, K. C.-C. Chang: Top k-Query Processing in Uncertain Databases. *ICDE Conference*, 2006.

[60] B. W. Silverman. Density Estimation for Statistics and Data Analysis. *Chapman and Hall*, 1986.

[61] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, S. Hambrusch. Indexing Uncertain Categorical Data. *ICDE Conference*, 2006.

[62] Y. Tao, R. Cheng, X. Xiao, W. Ngai, B. Kao, S. Prabhakar. Indexing Multi-dimensional Uncertain Data with Arbitrary Probabality Density Functions. *VLDB Conference*, 2005.

[63] W. Zhao, A. Dekhtyar, J. Goldsmith. A Framework for Management of Semistructured Probabilistic Data. *Journal of Intelligent Information Systems*, 1–39, 2004.

[64] E. Zimanyi: Query Evaluation in Probabilistic Databases. *Theoretical Computer Science*, 171(1–2), 1997.