Chapter 18

A Survey of Uncertain Data Clustering Algorithms

Charu C. Aggarwal
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
charu@us.ibm.com

18.1 Introduction

Many data sets which are collected often have uncertainty built into them. In many cases, the underlying uncertainty can be easily measured and collected. When this is the case, it is possible to use the uncertainty in order to improve the results of data mining algorithms. This is because the uncertainty provides a probabilistic measure of the relative importance of different attributes in data mining algorithms. The use of such information can enhance the effectiveness of data mining algorithms, because the uncertainty provides a guidance in the use of different attributes during the mining process. Some examples of real applications in which uncertainty may be used are as follows:

- Imprecise instruments and hardware are sometimes used in order to collect the data. In such

cases, the level of uncertainty can be measured by prior experimentation. A classic example of such hardware are sensors, in which the measurements are often imprecise.

- The data may be imputed by statistical methods, such as forecasting. In such cases, the uncertainty may be inferred from the methodology used in order to perform the imputation.

- Many privacy-preserving data mining techniques use probabilistic perturbations [11] in order to reduce the fidelity of the underlying data. In such cases, the uncertainty may be available as an end result of the privacy-preservation process. Recent work [8] has explicitly connected the problem of privacy-preservation with that of uncertain data mining, and has proposed a method which generates data, which is friendly to the use of uncertain data mining methods.

The problem of uncertain data has been studied in the traditional database literature [14, 44], though the issue has seen a revival in recent years [4, 8, 15, 19, 21, 22, 42, 49, 51, 52]. The driving force behind this revival has been the evolution of new hardware technologies such as sensors which cannot collect the data in a completely accurate way. In many cases, it has become increasingly possible to collect the uncertainty along with the underlying data values. Many data mining and management techniques need to be carefully re-designed in order to work effectively with uncertain data. This is because the uncertainty in the data can change the results in a subtle way, so that deterministic algorithms may often create misleading results [4]. While the raw values of the data can always be used in conjunction with data mining algorithms, the uncertainty provides additional insights which are not otherwise available. A survey of recent techniques for uncertain data mining may be found in [2].

The problem of clustering is a well known and important one in the data mining and management communities. The problem has been widely explored in the context of deterministic data. Details of a variety of clustering algorithms may be found in [37, 34]. The clustering problem has been widely studied in the traditional database literature [27, 38, 56] because of its applications to a variety of customer segmentation and data mining problems.

Uncertainty modeling is very relevant in the context of a number of different clustering applications. An example is illustrated in [41] in which uncertainty was incorporated into the clustering process in the context of a sales merchandising application. Since the problem of data clustering is closely related to that of classification, the methods for uncertain data clustering can also be used to enable algorithms for other closely related data mining problems such as outlier detection [6] and classification [4]. This is because clustering serves as a general-purpose summarization tool, which can be used in the context of a wide variety of problems.

The presence of uncertainty significantly affects the behavior of the underlying clusters. This is because the presence of uncertainty along a particular attribute may affect the expected distance between the data point and that particular attribute. In most real applications, there is considerable skew in the uncertainty behavior across different attributes. The incorporation of uncertainty into the clustering behavior can significantly affect the quality of the underlying results.

The problem of uncertain data clustering is often confused with that of *fuzzy clustering* [50]. In the case of uncertain data clustering, the uncertainty belongs to the *representation of the source objects which are being clusters*, and the actual clustering model may be either probabilistic or deterministic. In the case of fuzzy clustering [50], the source objects are typically deterministic, and the membership of objects to clusters is probabilistic. In other words, each object has a degree of belongingness to the different clusters, which is "fuzzy" or probabilistic in nature.

In this chapter, we will provide a survey of clustering algorithms for uncertain data. The main classes of clustering algorithms for uncertain data are as follows:

- **Mixture-Modeling Algorithms:** Mixture modeling techniques use probabilistic models for clustering uncertain data. A classic example of such an approach is given in [33], which uses an EM-approach [23] for the clustering process.

- **Density-based Methods:** A density-based method for uncertain data was proposed in [42]. This is referred to as the FDBSCAN algorithm. This approach modifies the DBSCAN algorithm to the case of uncertain data. An alternative method modifies the OPTICS algorithm to the case of uncertain data [43]. This is referred to as the FOPTICS algorithm.

- **Partitional Methods:** The K-means algorithm has been modified for the case of uncertain data [16, 48, 45, 20, 28]. Typically, the main challenge in these methods is that the uncertain distance computations for the *k*-means algorithms are too slow. Therefore, the focus is on improving efficiency by using pruning methods [48], speeding up distance computations [45], or by using fast approximation algorithms, which provide worst-case bounds [20, 28].

- **Streaming Algorithms:** The problem of clustering uncertain data has been extended to the case of data streams [5]. For this purpose, we extend the micro-clustering approach [10] to the case of data streams.

- **High Dimensional Algorithms:** High dimensional data poses a special challenge in the uncertain data, because the data is distributed in a very sparse way to begin with. The addition of uncertainty and noise further adds to the sparsity. Therefore, effective methods need to be designed for approximately determining clusters in such applications.

In this chapter, we will provide a detailed discussion of each of the above algorithms for uncertain data. This chapter is organized as follows. In the next section, we will discuss mixture model clustering of uncertain data. In section 18.3, we will describe density-based clustering algorithms for uncertain data. These include extensions of popular deterministic algorithms such as the DBSCAN and OPTICS algorithms. In section 18.4, we will discuss partitional algorithms for clustering uncertain data. Most of these methods are extensions of the *k*-means and *k*-median algorithms. This includes methods such as the UK-means, CK-means, and a number of approximation algorithms for clustering uncertain data. Section 18.5 discusses streaming algorithms for clustering uncertain data. Section 18.6 discusses high dimensional algorithms for clustering uncertain data. The uncertain data clustering problem has also been explored in the context of the possible worlds model in section 18.7. Section 18.9 contains the conclusions and summary.

18.2 Mixture Model Clustering of Uncertain Data

Mixture model clustering [23] is a popular method for clustering of deterministic data, and it models the clusters in the underlying data in terms of a number of probabilistic parameters. For example, the data can be modeled as a mixture of gaussian clusters, and then the parameters of this mixture can be learned from the underlying data. The core idea [23] is to determine model parameters, which ensure a maximum likelihood fit of the *observed instantiations* of the data with the proposed model. A popular method in order to determine these model parameters is the EM algorithm, which uses an expectation maximization approach to iteratively update the parameters with the observed data instances.

The work in [33] generalizes this approach to the case of uncertain data, where each data value may be drawn from an interval. The main difference between the uncertain version of the algorithm and the deterministic version is that each instantiation is now an uncertain value of the record, rather than a deterministic value. Correspondingly, the EM algorithm is also changed in order to evaluate the expressions in the E-step and M-step as an expectation over the uncertain range of the data value. We note that the approach can be used fairly easily for any uncertain distribution, which is represented in the form of a probability histogram of values.

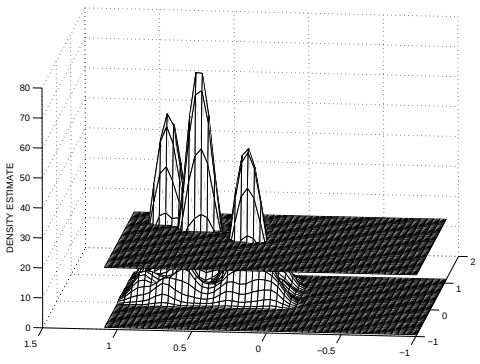

FIGURE 18.1: Density Based Profile with Lower Density Threshold

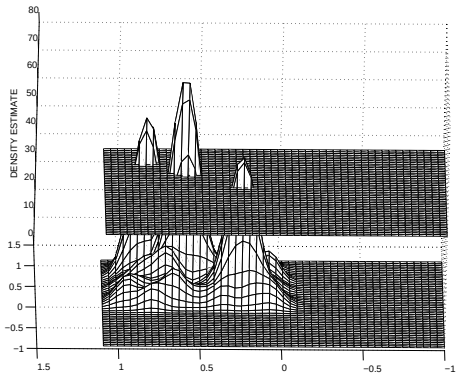

FIGURE 18.2: Density Based Profile with Higher Density Threshold

Another algorithm known as the *MMVar* algorithm has been proposed in [29], in which the centroid of a cluster C is defined as an uncertain object CMM, that represents the mixture model of C. The cluster compactness criterion used by the *MMVar* algorithm is the minimization of the variance of the cluster centroid.

18.3 Density Based Clustering Algorithms

Density-based methods are very popular in the deterministic clustering literature, because of their ability to determine clusters of arbitrary shapes in the underlying data. The core-idea in these methods is to create a density profile of the data set with the use of kernel density estimation methods. This density profile is then used in order to characterize the underlying clusters. In this section, we will discuss two variations of such density-based methods, which are the FDBSCAN and FOPTICS methods.

18.3.1 FDBSCAN Algorithm

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 [42] 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 18.3.1 *Let $\overline{X}$ and $\overline{Y}$ be two uncertain records, and let $p(\overline{X},\overline{Y})$ represent the distance density function between these objects. Then, the probability that the distance lies within the range (a,b) is given by the following relationship:*

$$P(a \leq d(\overline{X},\overline{Y}) \leq b) = \int_a^b p(\overline{X},\overline{Y})(z)dz \tag{18.1}$$

Based on this technique and the distance density function, the method in [42] 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 [24]. In the deterministic version of the algorithm [24], 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 of Figures 18.1 and 18.2 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 (Figure 18.2) results in 3 clusters, whereas the use of a lower density threshold results in 2 clusters. The fuzzy version of the DBSCAN algorithm (referred to as FDBSCAN) works in a similar way as 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.

18.3.2 FOPTICS Algorithm

Another related technique discussed in [43] is that of hierarchical density based clustering. An effective (deterministic) density based hierarchical clustering algorithm is OPTICS [12]. 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 [43] 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, once 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. A different hierarachical clustering algorithm with the use of an information-theoretic approach was proposed in [31].

18.4 Partitional Clustering Algorithms

Partitional clustering methods are algorithms which extend the *k*-means and *k*-medoid principles to the case of uncertain data. In this section, we will discuss these methods. The advantage of using partitional clustering methods is their relative simplicity and quick execution.

18.4.1 The UK-means Algorithm

A common approach to clustering is the *k*-means algorithm. In the *k*-means algorithm, we construct clusters around a pre-defined number of cluster centers. A variety of distance functions may be used in order to map the points to the different clusters. A *k*-means approach to clustering uncertain data was studied in the context of moving object data [16, 48]. In the case of moving objects, the actual locations of the objects may change over time as the data is reported intermittently. Thus, the position of a vehicle could be an arbitrary or circle region which uses the reported location as its center and has a size which is dependent upon the speed and direction of the vehicle. A probability density function could be used to model the probability of presence of the vehicle at a given location at a particular time.

One possibility is to simply replace each uncertain data point by a representative point such as its centroid, and apply the (deterministic) *k*-means clustering method directly to it. The UK-means clustering approach is very similar to the *K*-means clustering approach, except that we use the *expected distance* from the data's uncertainty region to the representative of the candidate cluster to which it is assigned. It was shown in [16], that the use of expected distances has clear advantages over an approach which uses deterministic clustering algorithms over representative data points. This approach is referred to as the UK-means algorithm.

A key challenge is the computation of the expected distances between the data points and the centroids for the *k*-means algorithm. A natural technique for computing these expected distances is to use Monte-carlo sampling, in which samples for the data points are used in order to compute the uncertain distances. Another technique is to create discrete buckets from both the distributions and compute the expected distances by a pairwise weighted average from different pairs of buckets. Thus, if one pdf is discretized into m_1 buckets, and another pdf is discretized into m_2 buckets, such an approach would require $m_1 \cdot m_2$ distance computations. The Monte-Carlo approach can be very expensive because a large number of samples may be required in order to compute the distances accurately. Similarly, a large number of discrete buckets may be required in order to compute the pairwise distances accurately. The work in [16] uses a purely brute-force version of the UK-means algorithm in which no optimization or pruning of the distance computations is performed. This version can be impractical, especially if a high level of accuracy is required in the clustering process. Clearly, some kind of pruning is required in order to improve the efficiency of the approach.

The work in [48] improves on the work of [16], and designs a pruned version of the UK-means algorithm. 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. In order to compute the bounds, the minimum bounding rectangle for the representative point for a cluster region is computed. The uncertain data point also represents a region over which the object may be distributed. For each representative cluster, its minimum bounding rectangle is used to compute the following two quantities with respect to the uncertain data point:

- The minimum limit on the expected distance between the MBR of the representative point and the uncertain region for the data point itself.
- The maximum limit on the expected distance between the MBR of the representative point and the uncertain region for the data point itself.

These upper and lower bound computations are facilitated by the use of the Minimum Bounding Rectangles in conjunction with the triangle inequality. We note that a cluster representative can be pruned, if its maximum limit is less than the minimum limit for some other representative. The approach is [48] constructs a *k*-d tree on the cluster representatives in order to promote an orderly pruning strategy and minimize the number of representatives which need to be accessed. It was shown in [48] that such an approach significantly improves the pruning efficiency over the brute force algorithm.

18.4.2 The CK-Means Algorithm

While the work in [16] claims that UK-means provides qualitatively superior results to deterministic clustering, the work in [45] shows that the model utilized by the UK-means is actually equivalent to deterministic clustering, by replacing each uncertain data point by its expected value. Thus, the UK-means approach actually turns out to be equivalent to deterministic clustering. This contradicts the claim in [16] that the UK-means algorithm provides superior results to a deterministic clustering method which replaces uncertain data points with their centroids. We further note that most of the computational complexity is created by the running time required for expected distance calculations. On the other hand, deterministic distance computations are extremely efficient, and are almost always superior to any method which is based on expected distance computations, whether or not pruning is used.

The UK-means algorithm aims to optimize the mean square expected distance about each cluster centroid. A key step is the computation of the expected square distance of an uncertain data point $\overline{X_i}$ with a cluster centroid $\overline{Y}$, where the latter is approximated as a deterministic entity. Let $\overline{Y}$ be the centroid of a cluster, and $\overline{X_1} \ldots \overline{X_r}$ be the set of data points in the cluster. Then, the expected mean square distance of data point $\overline{X_i}$ about $\overline{Y}$ is given by $E[||\overline{X_i} - \overline{Y}||^2]$. Then, if $\overline{Y}$ is approximated as a deterministic entity, then we can show the following:

Lemma 18.4.1 *Let $\overline{X_i}$ be an uncertain data point, and $\overline{Y}$ be a deterministic point. Let $\overline{c_i} = E[\overline{X_i}]$ and $var(\overline{X_i})$ represent the sum of the variances of the pdfs in $\overline{X_i}$ over all dimensions. Then, we have:*

$$\begin{aligned} E[||\overline{X_i} - \overline{Y}||^2] &= E[||\overline{X_i}||^2] - ||\overline{c_i}||^2 + ||\overline{c_i} - \overline{Y}||^2 \\ &= var(\overline{X_i}) + ||\overline{c_i} - \overline{Y}||^2 \end{aligned}$$

We will provide a proof of a generalized version of this lemma slightly later (Lemma 18.4.2). We further note that the value of $E[||\overline{X_i}||^2] - ||\overline{c_i}||^2$ is equal to the variance of the uncertain data point $\overline{X_i}$ (summed over all dimensions). The term $||\overline{c_i} - \overline{Y}||^2$ is equal to the *deterministic* distance of the $\overline{Y}$ to the centroid of the uncertain data point $\overline{X_i}$. Therefore, the expected square distance of an uncertain data point $\overline{X_i}$ to the centroid $\overline{Y}$ is given by the square sum of its deterministic distance and

the variance of the data point $\overline{X_i}$. The variance of the data point is not dependent on the value of $\overline{Y}$. Therefore, while computing the expected square distance to the different centroids for the UK-means algorithm, it suffices to compute the deterministic distance to the centroid $\overline{c_i}$ of $\overline{X_i}$ instead of computing the expected square distance. This means that by replacing each uncertain data point $\overline{X_i}$ with its centroid, the UK-means can be replicated exactly with an efficient deterministic algorithm.

It is important to note that the equivalence of the UK-means method to a deterministic algorithm is based on the approximation of treating each cluster centroid (in intermediate steps) as a deterministic entity. In practice, some of the dimensions may be much more uncertain than others in the clustering process over most of the data points. This is especially the case when different dimensions are collected using collection techniques with different fidelity. In such cases, the cluster centroids should not be treated as deterministic entities. Some of the streaming methods for uncertain data clustering such as those discussed in [5] also treat the cluster centroids as uncertain entities in order to enable more accurate computations. In those case, such deterministic approximations are not possible. Another method, which treats cluster centroids as uncertain entities was later proposed independently in [30]. The work on clustering streams, while treating centroids as uncertain entities will be discussed in a later section of this chapter.

18.4.3 Clustering Uncertain Data with Voronoi Diagrams

The work in [48] uses minimum bounding boxes of the uncertain objects in order to compute distance bounds for effective pruning. However, the use of minimax pruning can sometimes be quite restrictive in efficiently characterizing the uncertain object, which may have arbitrary shape. An approach which is based on voronoi diagrams, also improves the UK-means algorithms by computing the voronoi diagrams of the current set of cluster representatives [39]. Each cell in this voronoi diagram is associated with a cluster representative. We note that each cell in this voronoi diagram has the property that any point in this cell is closer to the cluster representative for that cell, than any other representative. Therefore, if the MBR of an uncertain object lies completely inside a cell, then it is not necessary to compute its distance to any other cluster representatives. Similarly, for any pair of cluster representatives, the perpendicular bisector between the two is a hyperplane which is equidistant from the two representatives and is easily derivable from the voronoi diagram. In the event that the MBR of an uncertain object lies completely on one side of the bisector, we can deduce that one of the cluster representatives of closer to the uncertain object than the other. This allows us to prune of the representatives.

As in [48], this work is focused on pruning the number of expected distance computations. It has been shown in [39] that the pruning power of the voronoi method is greater than the minmax method proposed in [48]. However, the work in [39] does not compare its efficiency results to those in [45], which is based on the equivalence of UK-means to a deterministic algorithm, and does not require any expected distance computations at all. It would seem to us, that any deterministic method for k-means clustering (as proposed in the reduction of [45]) should be much more efficient than a method based on pruning the number of expected distance computations, no matter how effective the pruning methodology might be.

18.4.4 Approximation Algorithms for Clustering Uncertain Data

Recently, techniques have been designed for approximation algorithms for uncertain clustering in [20]. The work in [20] discusses extensions of the k-mean and k-median version of the problems. Bi-criteria algorithms are designed for each of these cases. One algorithm achieves a $(1+\varepsilon)$-approximation to the best uncertain k-centers with the use of $O(k \cdot \varepsilon^{-1} \cdot \log^2(n))$ centers. The second algorithm picks $2k$ centers and achieves a constant-factor approximation.

A key approach proposed in the paper [20] is the use of a transformation from the uncertain case to a weighted version of the deterministic case. We note that solutions to the weighted version

of the deterministic clustering problem are well known, and require only a polynomial blow-up in the problem size. The key assumption in solving the weighted deterministic case is that the ratio of the largest to smallest weights is polynomial. This assumption is assumed to be maintained in the transformation. This approach can be used in order to solve both the uncertain *k*-means and *k*-median version of the problem with the afore-mentioned approximation guarantees. We refer the reader to [20, 28] for details of these algorithms.

18.4.5 Speeding up Distance Computations

We note that there are two main ways in which the complexity of distance computations in a *k*-means algorithm can be reduced. The first is by using a variety of pruning tricks, which cuts down on the *number* of distance computations between data points and cluster representatives. The second is by speeding up the expected distance computation itself. This kind of approach can especially be useful where the pruning effectiveness of a technique such as that proposed in [48] is not guaranteed. Therefore, a natural question arises as whether one can speed up the uncertain distance computations, which cause the performance bottleneck in these methods.

The work in [54] designs methods for speeding up distance computations for the clustering process. We note that such fast distance computations can be very effective not only for the UK-means algorithm, but for any clustering technique which is dependent on expected distance computations. The work in [54] proposes a number of methods for performing distance computations between uncertain objects, which provide different tradeoffs between effectiveness and efficiency. Specifically, for a pair of uncertain objects $\overline{X}$ and $\overline{Y}$, the following methods can be used in order to compute the distances between them:

- **Certain Representation:** Each uncertain object can be replaced by a certain object, corresponding to the expected values of its attributes. The distances between these objects can be computed in a straightforward way. While this approach is very efficient, it provides very poor accuracy.

- **Sampling:** It is possible to repeatedly sample both the objects for pairs of instantiations and compute the distances between them. The average of these computed distances can be reported as the expected value. However, such an approach may require a large number of samples in order to provide a high quality approximation.

- **Probability Histograms:** Each uncertain object can be approximated by a set of bins, which corresponds to its probability histogram. Then, for every pair of bins between the two objects, the probability of that instantiation and the distance between the average values of those bins is computed. The weighted average over all pairs of bins is reported. Such an approach can still be quite inefficient in many scenarios, where a large number of bins is required to represent the probability histogram effectively.

- **Gaussian Mixture Modeling with Sample Clustering:** Each uncertain object can be approximated with a mixture of gaussians. Specifically, we sample each uncertain object with the use of its pdf, and then cluster these samples with deterministic k-means clustering. Each of these clusters can be fit into a gaussian model. Then, the pairwise weighted average distances between each of the components of the mixture can be computed.

- **Single Gaussian Modeling:** It turns out that it is not necessary to use multiple components in the mixture model for the approximation process. In fact, it suffices to use a single component for the mixture.

The last result is actually not very surprising in light of Lemma 18.4.1. In fact, the Gaussian assumption is not required at all, and it can be shown that the distance between a pair of uncertain

objects (for which the pdfs are independent of one another), can be expressed purely as a function of their means and variances. Therefore, we propose the following (slight) generalization of Lemma 18.4.1.

Lemma 18.4.2 *Let $\overline{X_i}$ and $\overline{Y_i}$ be two uncertain data points, with means $\overline{c_i}$ and $\overline{d_i}$ respectively. Let the sum of the variances across all dimensions of these points be $var(\overline{X_i})$ and $var(\overline{Y_i})$ respectively. Then, we have:*

$$E[||\overline{X_i} - \overline{Y_i}||^2] = ||\overline{c_i} - \overline{d_i}||^2 + var(\overline{X_i}) + var(\overline{Y_i}) \tag{18.2}$$

Proof: We can expand the term within the expectation on the left hand side as follows:

$$E[||\overline{X_i} - \overline{Y_i}||^2] = E[||(\overline{X_i} - \overline{c_i}) + (\overline{c_i} - \overline{d_i}) + (\overline{d_i} - \overline{Y_i})||^2] \tag{18.3}$$

We further note that each of the three expressions within the round brackets on the right hand side are statistically independent of one another. This means that their covariances are zero. Furthermore, the expected values of $(\overline{X_i} - \overline{c_i})$ and $(\overline{d_i} - \overline{Y_i})$ are both 0. This can be used to show that the expectation of the product of any pair of terms within the round brackets on the right hand side of Equation 18.3 is 0. This implies that we can re-write the right hand side as follows:

$$E[||\overline{X_i} - \overline{Y_i}||^2] = E[||\overline{X_i} - \overline{c_i}||^2] + (\overline{c_i} - \overline{d_i})^2 + E[||\overline{d_i} - \overline{Y_i}||^2] \tag{18.4}$$

The first term in the RHS of the above expression is $var(\overline{X_i})$ and the last term is $var(\overline{Y_i})$. The result follows.

The afore-mentioned results suggest that it is possible to compute the distances between pairs of uncertain objects very efficiently, as long as the uncertainty in different objects are statistically independent. Another observation is that these computations *do not require knowledge of the full probability density function of the probabilistic records*, but can be made to work with the more modest assumption about the standard error $var(\cdot)$ of the underlying uncertainty. This is a more reasonable assumption for many applications. Such standard errors are included as a natural part of the measurement process, though the full probability density functions are rarely available. This also suggests that a lot of work on pruning the number of expected distance computations may not be quite as critical to efficient clustering, as has been suggested in the literature.

18.5 Clustering Uncertain Data Streams

In many applications such as sensor data, the data may have uncertainty, because of errors in the readings of the underlying sensors. This may result in uncertain *streams* of data. Uncertain streams pose of special challenge because of the dual complexity of high volume and data uncertainty. As we have seen in earlier sections, efficiency is a primary concern in the computation of expected distances, when working with probability density functions of data points. Therefore, it is desirable to work with simpler descriptions of the underlying uncertainty. This will reduce both the underlying data volume and complexity of stream computations. In recent years, a number of methods have specifically been proposed for clustering uncertain data streams.

18.5.1 The UMicro Algorithm

In this section, we will introduce *UMicro*, the Uncertain MICROclustering algorithm for data streams. We assume that we have a data stream which contains d dimensions. The actual records in

the data are denoted by $\overline{X_1}$, $\overline{X_2}$, ... $\overline{X_N}$.... We assume that the estimated error associated with the jth dimension for data point $\overline{X_i}$ is denoted by $\psi_j(\overline{X_i})$. This error is defined in terms of the standard deviation of the error associated with the value of the jth dimension of $\overline{X_i}$. The corresponding d-dimensional error vector is denoted by $\overline{\psi(\overline{X_i})}$. Thus, the input to the algorithm is a data stream in which the ith pair is denoted by $(\overline{X_i}, \overline{\psi(\overline{X_i})})$.

We note that most of the uncertain clustering techniques work with the assumption that the entire probability density function is available. In many real applications, a more realistic assumption is that only the standard deviations of the errors are available. This is because complete probability distributions are rarely available, and are usually inserted only as a modeling assumption. An overly ambitious modeling assumption can also introduce modeling errors. It is also often quite natural to be able to estimate the standard error in many modeling scenarios. For example, in a scientific application in which the measurements can vary from one observation to another, the error value is the standard deviation of the observations over a large number of measurements. In a k-anonymity based data (or incomplete data) mining application, this is the standard deviation of the partially specified (or imputed) fields in the data. This is also more practical from a stream perspective, because it reduces the volume of the incoming stream, and reduces the complexity of stream computations.

The micro-clustering model was first proposed in [56] for large data sets, and subsequently adapted in [10] for the case of deterministic data streams. The *UMicro* algorithm extends the micro-clustering approach of [10] to the case of uncertain data. 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 18.5.1 *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 $(\overline{CF2^x}(C), \overline{EF2^x}(C), \overline{CF1^x}(C), t(C), n(C))$, wherein $\overline{CF2^x}(C)$, $\overline{EF2^x}(C)$, and $\overline{CF1^x}(C)$ each correspond to a vector of d entries. The entries in $\overline{EF2^x}(C)$ 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 $\overline{CF2^x}(C)$. Thus, $\overline{CF2^x}(C)$ contains d values. The p-th entry of $\overline{CF2^x}(C)$ 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 $\overline{EF2^x}(C)$. Thus, $\overline{EF2^x}(C)$ contains d values. The p-th entry of $\overline{EF2^x}(C)$ 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 $\overline{CF1^x}(C)$. Thus, $\overline{CF1^x}(C)$ contains d values. The p-th entry of $\overline{CF1^x}(C)$ 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 $n(C)$.*
- *The time stamp of the last update to the micro-cluster is maintained in $t(C)$.*

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 C by $\overline{ECF}(C)$. We note that error based micro-clusters maintain the important *additive property* [10] which is critical to its use in the clustering process. We restate the additive property as follows:

Property 18.5.1 *Let C_1 and C_2 be two sets of points. Then all non-temporal components of the error-based cluster feature vector $\overline{ECF}(C_1 \cup C_2)$ are given by the sum of $\overline{ECF}(C_1)$ and $\overline{ECF}(C_2)$.*

```
Algorithm UMicro(Number of Clusters: n_micro)
begin
S = {};
{ Set of micro-clusters }
repeat
   Receive the next stream point X̄;
   { Initially, when S is null, the computations below
    cannot be performed, and X̄ is simply
    added as a singleton micro-cluster to S }
   Compute the expected similarity of X̄ to the closest
    micro-cluster M in S;
   Compute critical uncertainty boundary of M;
   if X̄ lies inside uncertainty boundary
   add X̄ to statistics of M
   else
   add a new micro-cluster to S containing singleton
      point X̄;
   if |S| = n_micro + 1 remove the least recently
       updated micro-cluster from S;
until data stream ends;
end
```

FIGURE 18.3: The UMicro Algorithm

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 $t(C_1 \cup C_2)$ is given by $\max\{t(C_1), t(C_2)\}$. 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. The overall framework for the uncertain stream clustering algorithm is illustrated in Figure 18.3. Next, we will discuss the process of computation of individual subroutines such as the expected distance or the uncertain boundary.

In order to compute the expected similarity of the data point $\overline{X}$ to the centroid of the cluster C, we need to determine a closed form expression which is expressed only in terms of $\overline{X}$ and $ECF(C)$. We note that just as the individual data points are essential random variables with a given error, the centroid $\overline{Z}$ of a cluster C is also a random variable. We make the following observation about the centroid of a cluster:

Lemma 18.5.1 *Let $\overline{Z}$ be the random variable representing the centroid of cluster C. Then, the following result holds true:*

$$E[||Z||^2] = \sum_{j=1}^{d} CF1(C)_j^2/n(C)^2 + \sum_{j=1}^{d} EF2(C)_j/n(C)^2 \tag{18.5}$$

Proof: We note that the random variable Z_j is given by the current instantiation of the centroid and the mean of $n(C)$ different error terms for the points in cluster C. Therefore, we have:

$$Z_j = CF1(C)_j/n(C) + \sum_{\overline{X}\in C} e_j(\overline{X})/n(C) \tag{18.6}$$

Then, by squaring $\overline{Z_j}$ and taking the expected value, we obtain the following:

$$E[Z_j^2] = CF1(C)_j^2/n(C)^2 + 2\cdot \sum_{\overline{X}\in C} E[e_j(\overline{X})]\cdot CF1(C)_j/n(C)^2 + E[(\sum_{\overline{X}\in C} e_j(\overline{X}))^2]/n(C)^2 \tag{18.7}$$

Now, we note that the error term is a random variable with standard deviation $\psi_j(\cdot)$ and zero mean. Therefore $E[e_j] = 0$. Further, since it is assumed that the random variables corresponding to the errors of different records are independent of one another, we have $E[e_j(\overline{X})\cdot e_j(\overline{Y})] = E[e_j(\overline{X})]\cdot E[e_j(\overline{Y})] = 0$. By using these relationships in the expansion of the above equation we get:

$$\begin{aligned} E[Z_j^2] = CF1(C)_j^2/n(C)^2 + \sum_{\overline{X}\in C} E[e_j(\overline{X})^2]/n(C)^2 &= CF1(C)_j^2/n(C)^2 + \sum_{\overline{X}\in C} \psi_j(\overline{X})^2/n(C)^2 \\ &= CF1(C)_j^2/n(C)^2 + EF2(C)_j/n(C)^2 \end{aligned}$$

By adding the value of $E[Z_j^2]$ over different values of j, we get:

$$E[||Z||^2] = \sum_{j=1}^{d} CF1(C)_j^2/n(C)^2 + \sum_{j=1}^{d} EF2(C)_j/n(C)^2 \tag{18.8}$$

This proves the desired result.

Next, we will use the above result to directly estimate the expected distance between the centroid of cluster C and the data point $\overline{X}$. We will prove the following result:

Lemma 18.5.2 *Let v denote the expected value of the square of the distance between the uncertain data point $\overline{X} = (x_1 \ldots x_d)$ (with instantiation $(x_1 \ldots x_d)$ and error vector $(\psi_1(\overline{X}) \ldots \psi_d(\overline{X}))$ and the centroid of cluster C. Then, v is given by the following expression:*

$$v = \sum_{j=1}^{d} CF1(C)_j^2/n(C)^2 + \sum_{j=1}^{d} EF2(C)_j/n(C)^2 + \sum_{j=1}^{d} x_j^2 + \sum_{j=1}^{d} (\psi_j(\overline{X}))^2 - 2\sum_{j=1}^{d} x_j\cdot CF1(C)_j/n(C) \tag{18.9}$$

Proof: Let $\overline{Z}$ represent the centroid of cluster C. Then, we have:

$$v = E[||\overline{X} - \overline{Z}||^2] = E[||\overline{X}||^2] + E[||\overline{Z}||^2] - 2E[\overline{X} \cdot \overline{Z}] = E[||\overline{X}||^2] + E[||\overline{Z}||^2] - 2E[\overline{X}] \cdot E[\overline{Z}]$$

Next, we will analyze the individual terms in the above expression. We note that the value of X is a random variable, whose expected value is equal to its current instantiation, and it has an error along the jth dimension which is equal to $\psi_j(\overline{X})$. Therefore, the expected value of $E[||\overline{X}||^2]$ is given by:

$$E[||\overline{X}||^2] = (E[X])^2 + \sum_{j=1}^{d} (\psi_j(\overline{X}))^2 = \sum_{j=1}^{d} x_j^2 + \sum_{j=1}^{d} (\psi_j(\overline{X}))^2$$

Now, we note that the jth term of $E[Z]$ is equal to the jth dimension of the centroid of cluster C. This is given by the expression $CF1(C)_j/n(C)$, where $CF1_j(C)$ is the jth term of the first order cluster component $CF1(C)$. Therefore, the value of $E[X] \cdot E[Z]$ is given by the following expression:

$$E[X] \cdot E[Z] = \sum_{j=1}^{d} x_j \cdot CF1(C)_j/n(C) \tag{18.10}$$

The results above and Lemma 18.5.1 define the values of $E[||X||^2]$, $E[||Z||^2]$, and $E[X \cdot Z]$. Note that all of these values occur in the right hand side of the following relationship:

$$v = E[||\overline{X}||^2] + E[||\overline{Z}||^2] - 2E[\overline{X}] \cdot E[\overline{Z}] \tag{18.11}$$

By substituting the corresponding values in the right hand side of the above relationship, we get:

$$v = \sum_{j=1}^{d} CF1(C)_j^2/n(C)^2 + \sum_{j=1}^{d} EF2(C)_j/n(C)^2 + \sum_{j=1}^{d} x_j^2 + \sum_{j=1}^{d} (\psi_j(\overline{X}))^2 - 2\sum_{j=1}^{d} x_j \cdot CF1(C)_j/n(C) \tag{18.12}$$

The result follows.

The result of Lemma 18.5.2 establishes how the square of the distance may be computed (in expected value) using the error information in the data point $\overline{X}$ and the micro-cluster statistics of C. Note that this is an efficient computation which requires $O(d)$ operations, which is asymptotically the same as the deterministic case. This is important since distance function computation is the most repetitive of all operations in the clustering algorithm, and we would want it to be as efficient as possible.

While the expected distances can be directly used as a distance function, the uncertainty adds a lot of noise to the computation. We would like to remove as much noise as possible in order to determine the most accurate clusters. Therefore, we design a dimension counting similarity function which prunes the uncertain dimensions during the similarity calculations. This is done by computing the variance σ_j^2 along each dimension j. The computation of the variance can be done by using the cluster feature statistics of the different micro-clusters. The cluster feature statistics of all micro-clusters are added to create one global cluster feature vector. The variance of the data points along each dimension can then be computed from this vector by using the method discussed in [56]. For each dimension j and threshold value *thresh*, we add the *similarity value* $\max\{0, 1 - E[||X - Z||_j^2]/(thresh * \sigma_j^2)\}$ to the computation. We note that this is a similarity value rather than a distance value, since larger values imply greater similarity. Furthermore, dimensions which have a large amount of uncertainty are also likely to have greater values of $E[||X - Z||_j^2]$, and are often pruned from the computation. This improves the quality of the similarity computation.

In this section, we will describe the process of computing the uncertain boundary of a micro-cluster. Once the closest micro-cluster for an incoming point has been determined, we need to decide whether it should be added to the corresponding micro-clustering statistics, or whether a

new micro-cluster containing a singleton point should be created. We create a new micro-cluster, if the incoming point lies outside the uncertainty boundary of the micro-cluster. The uncertainty boundary of a micro-cluster is defined in terms of the standard deviation of the distances of the data points about the centroid of the micro-cluster. Specifically, we use t standard deviations from the centroid of the cluster as a boundary for the decision of whether to include that particular point in the micro-cluster. A choice of $t = 3$ ensures a high level of certainty that the point does not belong to that cluster with the use of the normal distribution assumption. Let $\overline{W}$ be the centroid of the cluster C, and let the set of points in it be denoted by $\overline{Y_1} \ldots \overline{Y_r}$. Then, the uncertain radius U is denoted as follows:

$$U = \sum_{i=1}^{r} \sum_{j=1}^{d} E[||Y_i - W||_j^2] \tag{18.13}$$

The expression on the right hand side of the above Equation can be evaluated by using the relationship of Lemma 18.5.2.

18.5.2 The LuMicro Algorithm

A variation of the *UMicro* algorithm has been discussed in [55], which incorporates the concept of tuple uncertainty into the clustering process. The primary idea in this approach is that the *instance uncertainty* of a cluster is quite important, in addition to the expected distances of assignment. If T be the set of possible probabilistic instances of a tuple, then the instance uncertainty $U(T)$ is defined as follows:

$$U(T) = -\sum_{x_i \in T} p(x_i) \cdot \log(p(x_i)) \tag{18.14}$$

We note that the value of $U(T)$ is somewhat akin to the concept of entropy, is always at least 0, and takes on the least value of 0 for deterministic data. This concept can also be generalized to a cluster (rather than a single tuple) by integrating all possible probabilistic instances into the computation. As more data points are added to the cluster, the tuple uncertainty decreases, because the data in the cluster tends to be biased towards a few common tuple values. Intuitively, this is also equivalent to a reduction in entropy. The *LuMicro* algorithm, implements a very similar approach as the *UMicro* method in terms of assigning data points to their closest clusters (based on expected distance), except that the distance computation is only used to narrow down to a smaller set of candidate centroids. The final decision on centroid assignment is performed by determining the cluster to which the addition of the data point would result in the greatest reduction in uncertainty (or entropy). Intuitively, this can be considered an algorithm which incorporates distance-based and probabilistic entropy-based concepts into the clustering process. Unlike the *UMicro* algorithm, the *LuMicro* method works with the full probability distribution functions of the underlying records, rather than only the error values. This is because the computation of the uncertainty values requires knowledge of the full probability distribution of the tuples.

18.5.3 Enhancements to Stream Clustering

The method for clustering uncertain data streams can be further enhanced in several ways:

- In many applications, it is desirable to examine the clusters over a specific time horizon rather than the entire history of the data stream. In order to achieve this goal, a pyramidal time frame [10] can be used for stream classification. In this time-frame, snapshots are stored in different orders depending upon the level of recency. This can be used in order to retrieve clusters over a particular horizon with very high accuracy.
- In some cases, the behavior of the data stream may evolve over time. In such cases, it is useful to apply a *decay-weighted* approach. In the decay-weighted approach, each point in the stream

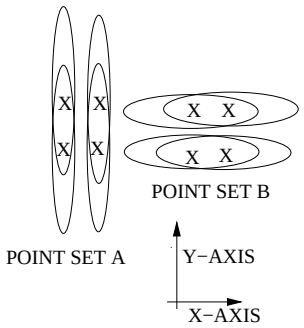

FIGURE 18.4: Effect of Uncertainty in picking projections

is a weighted by a factor which decays over time. Such an approach can be useful in a number of scenarios in which the behavior of the data stream changes considerably over time. In order to use the decay-weighted approach, the key modification is to define the micro-clusters with a weighted sum of the data points, as opposed to the explicit sums. It can be shown that such an approach can be combined with a lazy-update method in order to effectively maintain the micro-clusters.

18.6 Clustering Uncertain Data in High Dimensionality

Recently, this method has also been extended to the case of projected and subspace clustering of high dimensional uncertain data [32, 7]. The high dimensional scenario suffers from data sparsity, which makes it particularly susceptible to noise. The addition of uncertainty typically increases the noise, and reduces the correlations among different dimensions. This tends to magnify the high dimensional sparsity issue and makes the problem even more challenging.

In the case of the standard clustering problem, the main effect of uncertainty is the impact on the distance computations. However, in the uncertain case, the uncertainty also affects the choice of dimensions to be picked. The reason for this is that different dimensions in the data can have very different levels of uncertainty. Clearly, the level of uncertainty in a given dimension is critical information in characterizing the clustering behavior along a particular dimension. This is particularly important for the high dimensional case in which a very large number of dimensions may be available with varying clustering behavior and uncertainty. The interplay between the clustering of the values and the level of uncertainty may affect the subspaces which are most optimal for the clustering process. In some cases, if the uncertainty data is not used in the mining process, this may result in a clustering which does not truly reflect the underlying behavior.

For example, consider the case illustrated in Figure 18.4. In this case, we have illustrated two clusters which are denoted by “Point Set A” and “Point Set B” respectively. In each case, we have also illustrated the uncertainty behavior with elliptical contours. The two data sets are identical, except that the uncertainty contours are very different. In the case of point set A, it is better to pick the projection along the X-axis, because of lower uncertainty along that axis. On the other hand, in the case of point set B, it is better to pick the projection along the Y-axis because of lower uncertainty in that direction. This problem is further magnified when the dimensionality increases, and the different dimensions have different patterns of data and uncertainty distributions. We will examine the interplay between data uncertainty and projections along different dimensionalities for

the clustering process. We will show that the incorporation of uncertainty information into critical algorithmic decisions leads to much better quality of the clustering.

In this section, we will discuss two different algorithms, one of which allows overlap among the different clusters, and the other designs a method for strict partitioning of the data in the uncertain streaming scenario. The first case creates a *soft partitioning* of the data, in which data points belong to clusters with a probability. This is also referred to as *membership degree*, a concept which we will discuss in the next subsection.

18.6.1 Subspace Clustering of Uncertain Data

A subspace clustering algorithm for uncertain data was proposed in [32]. The algorithm uses a grid based method, which attempts to search on the space of medoids and subspaces for the clustering process. In the grid-based approach, the support is counted with a width *w* on the relevant subset of dimensions. Thus, for a given medoid *m*, we examine a distance *w* from the medoid along each of the relevant dimensions. The support of the hypercubes of this grid provide us with an idea of the dense subspaces in the data. For the other dimensions, unlimited width is considered. A Monte-Carlo sampling approach is used in order to search on the space of possible medoids. The core unit of the algorithm is a Monte-Carlo sampling approach, which generates a single good subspace cluster from the database.

In order to achieve this goal, a total of *numMedoids* are sampled from the underlying data. For each such medoid, its best possible local subspace is constructed, in order to generate the grid-based subspace cluster. The quality of this local subspace is identified, and the best medoid (and associated subspace cluster) among all the *numMedoids* different possibilities is identified. In order to generate the local subspaces around the medoid, a support parameter called *minSup* is used. For all local subspaces (corresponding to grid width *w*), which have support of at least *minSup*, the *quality* of the corresponding subspace is determined. The quality of a local subspace cluster is different from the support in order to account for the different number of dimensions in the different subspaces. If this quality is the best encountered so far, then we update the best medoid (and corresponding subspace) encountered so far. The quality function for a medoid *m* and subspace *S* is related to the support as follows:

$$quality(m,S) = support(m,S) * 1/\beta^{|S|} \tag{18.15}$$

Here $\beta \in (0,1)$ normalizes for the different number of dimensions in the different subspaces *S*. The idea is that a subspace with a larger number of dimensions, but with the same support, is considered to be of better quality. A number of different methods can be used in order to compute the support of the subset *S* of dimensions:

- **Expectation-based Support:** In this case, the support is defined as the number of data points, whose centroids lie within a given distance *w* of the medoid along the relevant dimensions. Essentially, this method for support computation is similar to the deterministic case by replacing uncertain objects with their centroids.
- **Minimal Probability-based Support:** In this case, the support is defined as the number of data points, which have a minimum probability of being within a distance of *w* from the medoid along each of the relevant dimensions.
- **Exact Probability-based Support:** This computes the sum of the probabilities that the different objects lie within a distance of *w* along each of the relevant dimensions. This value is actually equal to expected number of objects which lie within a width of *w* along the relevant dimensions.

We note that the last two measurements require the computation of a probability that an uncertain

object lies within a specific width w of a medoid. This probability also reflects the *membership degree* of the data point to the cluster.

We note that the afore-mentioned technique only generates a single subspace cluster with the use of sampling. A question arises as to how we can generalize this in order to generate the *overall* clustering. We note that repeated samplings may generate the same set of clusters, a scenario which we wish to avoid. In order to reduce repeated clusters, two approaches can be used:

- Objects which have a minimal probability of belonging to any of the previously generated clusters are excluded from consideration for being medoids.
- The probability of an object being selected as a medoid depends upon its membership degree to the previously generated clusters. Objects which have very low membership degrees to previously generated clusters have a higher probability of being selected as medoids.

The work in [32] explores the different variations of the subspace clustering algorithms, and shows that the methods are superior to methods such as UK-means and deterministic projected clustering algorithms such as PROCLUS [3].

18.6.2 UPStream: Projected Clustering of Uncertain Data Streams

The *UPStream* algorithm is designed for the high dimensional uncertain *stream scenario*. This algorithm can be considered an extension of the *UMicro* algorithm. The error model of the *UPStream* algorithm is quite different from the algorithm of [32], and uses a model of error standard deviations rather than the entire probability distribution. This model is more similar to the *UMicro* algorithm.

The data stream consists of a set of incoming records which are denoted by $\overline{X_1} \ldots \overline{X_i} \ldots$. It is assumed that the data point $\overline{X_i}$ is received at the time stamp T_i. It is assumed that the dimensionality of the data set is d. The d dimensions of the record $\overline{X_i}$ are denoted by $(x_i^1 \ldots x_i^d)$. In addition, each data point has an error associated with the different dimensions. The error (standard-deviation) associated with the jth dimension for data point $\overline{X_i}$ is denoted by $\psi_j(\overline{X_i})$.

In order to incorporate the greater importance of recent data points in an evolving stream, we use the concept of a *fading function* $f(t)$, which quantifies the relative importance of the different data points over time. The fading function is drawn from the range $(0,1)$, and serves as a multiplicative factor for the relative importance of a given data point. This function is a monotonically decreasing function, and represents the gradual fading of importance of a data point over time. A commonly used decay function is the exponential decay function. This function is defined as follows. The exponential decay-function $f(t)$ with parameter λ is defined as follows as a function of the time t:

$$f(t) = 2^{-\lambda \cdot t} \tag{18.16}$$

We note that the value of $f(t)$ reduces by a factor of 2 every $1/\lambda$ time units. This corresponds to the half-life of the function $f(t)$. We define the half-life as follows:

Definition 18.6.1 *The half-life of the function $f(\cdot)$ is defined as the time t at which $f(t) = (1/2) \cdot f(0)$. For the exponential decay function, the half-life is $1/\lambda$.*

In order to keep track of the statistics for cluster creation, two sets of statistics are maintained:

- Global data statistics which keep track of the variance along different dimensions of the data. This data is necessary in order to maintain information about the scaling behavior of the underlying data.
- Fading micro-cluster statistics which keep track of the cluster behavior, the projection dimensions as well as the underlying uncertainty.

Let us assume that the data points that have arrived so far are $\overline{X_1} \ldots \overline{X_N} \ldots$. Let t_c be the current time.

- For each dimension, the weighted sum of the squares of the individual dimensions of $\overline{X_1} \ldots \overline{X_N} \ldots$ over the entire data stream are maintained. There are a total of d such entries. The ith component of the global second-order statistics is denoted by $gs(i)$ and is equal to $\sum_{j=1}^{N} f(t_c - T_j) \cdot (x_j^i)^2$.
- For each dimension, the sum of the individual dimensions of $\overline{X_1} \ldots \overline{X_N} \ldots$ over the entire data stream are maintained. There are a total of d such entries. The ith component of the global first-order statistics is denoted by $gf(i)$ and is equal to $\sum_{j=1}^{N} f(t_c - T_j) \cdot (x_j^i)$.
- The sum of the weights of the different values of $f(T_j)$ are maintained. This value is equal to $\sum_{j=1}^{N} f(t_c - T_j)$. This value is denoted by gW.

The above statistics can be easily maintained over a data stream since the values are computed additively over arriving data points. At first sight, it would seem that the statistics need to be updated at each clock tick. In reality, because of the multiplicative nature of the exponential distribution, we only need to update the statistics on the arrival of each new data point. Whenever a new data point arrives at time T_i, we multiply each of the statistics by $e^{-\lambda \cdot T_i - T_{i-1}}$ and then add the statistics for the incoming data point $\overline{X_i}$. We note that the global variance along a given dimension can be computed from the above values. Therefore, the global variance can be maintained continuously over the entire data stream.

Observation 18.6.1 *The variance along the ith dimension is given by* $\frac{gs(i)}{gW} - \frac{gf(i)^2}{gW^2}$.

The above fact can be easily proved by using the fact that for any random variable Y the variance $var(Y)$ is given by $E[Y^2] - E[Y]^2$. We will denote the global standard deviation along dimension i at time t_c by $\sigma(i, t_c)$. As suggested by the observation above, the value of $\sigma(i, t_c)$ is easy to maintain by using the global statistics discussed above.

An uncertain micro-cluster $C = \{X_{i_1} \ldots X_{i_N}\}$ is represented as follows.

Definition 18.6.2 *The uncertain micro-cluster for a set of d-dimensional points* $\overline{X_{i_1}} \ldots \overline{X_{i_n}}$ *with time stamps given by* $T_{i_1} \ldots T_{i_n}$, *and error vectors* $\overline{\psi(X_{i_1})} \ldots \overline{\psi(X_{i_n})}$ *is defined as the* $(3 \cdot d + 3)$ *tuple* $\overline{ECF(C)} = (\overline{CF2}(C), \overline{EF2}(C), \overline{CF1}(C), t(C), W(C), n(C))$, *and a d-dimensional bit vector* $B(C)$, *wherein the corresponding entries are defined as follows:*

- *For each of the d dimensions, we maintain the weighted sum of the squares of the data values in* $\overline{CF2(C)}$. *The p-th entry is given by* $\sum_{j=1}^{n} f(t - T_{i_j}) \cdot (x_{i_j}^p)^2$.
- *For each of the d dimensions, we maintain the weighted sum of the squares of the errors (along the corresponding dimension) in* $\overline{EF2(C)}$. *The p-th entry is given by* $\sum_{j=1}^{n} f(t - T_{i_j}) \cdot \psi_p(X_{i_j})^2$.
- *For each of the d dimensions, we maintain the weighted sum of the data values in* $\overline{CF1(C)}$. *The p-th entry is given by* $\sum_{j=1}^{n} f(t - T_{i_j}) \cdot x_{i_j}^p$.
- *The sum of the weights is maintained in* $W(C)$. *This value is equal to* $\sum_{j=1}^{n} f(t - T_{i_j})$.
- *The number of data points is maintained in* $n(C)$.
- *The last time at which a data point was added to the cluster is maintained in* $t(C)$.
- *We also maintain a d-dimensional bit-vector* $B(C)$. *Each bit in this vector corresponds to a dimension. A bit in this vector takes on the value of 1, if that dimension is included in the projected cluster. Otherwise, the value of the bit is zero.*

This definition is quite similar to the case of the *UMicro* algorithm, except that there is also a focus on maintaining dimension-specific information and the time-decay information. We note that the micro-cluster definition discussed above satisfies two properties: the *additive property*, and the *multiplicative property*. The additive property is common to all micro-clustering techniques:

Observation 18.6.2 Additive Property *Let C_1 and C_2 be two sets of points. Then the components of the error-based cluster feature vector (other than the time stamp) $\overline{ECF(C_1 \cup C_2)}$ are given by the sum of $\overline{ECF(C_1)}$ and $\overline{ECF(C_2)}$.*

The additive property is helpful in streaming applications, since the statistics for the micro-clusters can be modified by simply adding the statistics for the incoming data points to the micro-cluster statistics. However, the micro-cluster statistics also include time-decay information of the underlying data points, which can potentially change at each time-stamp. Therefore, we need an effective way to update the micro-cluster statistics without having to explicitly do so at each time stamp. For this purpose, the *multiplicative-property* is useful.

Observation 18.6.3 Multiplicative Property *The decaying components of $\overline{ECF(C)}$ at time t_c can be obtained from the component values at time $t_s < t_c$ by multiplying each component by $2^{-\lambda \cdot (t_c - t_s)}$ provided that no new points have been added to a micro-cluster.*

The multiplicative property follows from the fact the statistics decay at the multiplicative rate of $2^{-\lambda}$ at each tick. We note that the multiplicative-property is important in ensuring that a *lazy update process* can be used for updating the decaying micro-clusters, rather than at each clock-tick. In the lazy-update process, we update a micro-cluster only when a new data point is added to it. In order to do so, we first use the multiplicative property to adjust for time-decay, and then we use the additive property to add the incoming point to the micro-cluster statistics.

The *UPStream* algorithm uses a continuous partitioning and projection strategy in which the different micro-clusters in the stream are associated with a particular projection, and this projection is used in order to define the assignment of data points to clusters. The input to the algorithm is the number of micro-clusters k which are to be determined by the algorithm. The algorithm starts off with a empty set of clusters. The initial set of k data points are assigned to singleton clusters in order to create the initial set of seed micro-clusters. This initial set of micro-cluster statistics provides a starting point which is rapidly modified by further updates to the micro-clusters. For each incoming data point, probabilistic measures are computed over the projected dimensions in order to determine the assignment of data points to clusters. These assignments are used to update the statistics of the underlying clusters. These updates are combined with a probabilistic approach for determining the expected distances and spreads along the projected dimensions. In each update iteration, the details of the steps performed are as follows:

- We compute the global moment statistics associated with the data stream by using the multiplicative and additive property. If t_s be the last time of arrival of a data stream point, and t_c be the current time of arrival, then we multiply the moment statistics by $2^{-\lambda \cdot (t_c - t_s)}$, and add the current data point.

- For each micro-cluster, we compute and update the set of dimensions associated with it. This computation process uses both the uncertainty information of data points within the different micro-clusters. A critical point here is that the original data points which have already been received from the stream are not available, but only the summary micro-cluster information is available. The results in [7] show that the summary information encoded in the micro-clusters is sufficient to determine the projected dimensions effectively.

- We use the projected dimensions in order to compute the expected distances of the data points

to the various micro-clusters. The closest micro-cluster is picked based on the expected projected distance. As in the previous case, the original data points which have already been received from the stream are not available. The information encoded in the micro-clusters is sufficient to compute the expected distances.

- We update the statistics of the micro-clusters based on the incoming data points. The additive and the multiplicative property are useful for updating the micro-clusters effectively for each incoming data point. Since the micro-cluster statistics contains information about the last time the micro-cluster was updated, the multiplicative property can be used in order to update the decay behavior of that micro-cluster. Subsequently, the data point can be added to the corresponding micro=cluster statistics with the use of the additive property.

The steps discussed above are repeated for each incoming data point. The entire clustering algorithm is executed by repeating this process over different data stream points.

18.7 Clustering with the Possible Worlds Model

The "possible worlds model" is the most generic representation of uncertain databases in which no assumptions are made about the independence of different tuples in the database or across different dimensions [1]. All the algorithms discussed so far in this paper make the assumption of independence between tuples and also among different dimensions. In practice, many uncertain databases, in which the records are generated by mutually exclusive or correlated events may be highly dependent in nature. Such databases are drawn from the possible worlds model, and a particular instantiation of the database may have a high level of dependence among the different tuples. Such a method for possible worlds-based clustering has been proposed in [53].

A general-purpose method for performing data analytics in such scenarios is to use Monte-Carlo sampling to generate different instantiations of the database, and then apply the algorithms to each sample [36]. Subsequently, the output of the algorithms on the different samples is merged in order to provide a single global result. We note that the key to the success of this method is the design of an effective sample generator for the uncertain data. In this case, an effective methodology is the use of the value generator functions [36] for VG+ function.

For the case of the clustering application, a total of M possible worlds are generated with the use of the VG+ function. Each of these local samples is then clustered with the use of the deterministic DBSCAN algorithm [24]. In practice, any clustering methodology can be used, bit we work with DBSCAN, because it was used in the case of the possible world clustering proposed in [53]. Since the different samples are completely independent of one another, it is possible to use a high level of parallelism in the clustering process. This results in a total of M possible clusterings of the different samples.

The final step is to merge these M different clusterings into a single clustering. For this purpose, a *clustering aggregation* method which is similar to that proposed in [26] is leveraged. A similarity graph is generated for each of the clusterings. Each uncertain tuple in the database is treated as a node, and an edge is placed between two tuples, if they appear in the same cluster in that particular sample. Thus, a total of M possible similarity graphs can be generated. These M different similarity graphs are merged into a single global similarity graph with the use of techniques discussed in [26]. The final set of clusters are determined by determining the clustered regions of the global similarity graph.

18.8 Clustering Uncertain Graphs

In recent years, uncertain graphs have been studied extensively, because of numerous applications in which uncertainty is present on the edges. Many forms of graphs in biological networks are derived through statistical analysis. Therefore, the links are uncertain in nature. Thus, an uncertain graph is defined as a network $G = (N, A, P)$, where N is the set of nodes, A is the set of edges, and P is a set of probabilities such that each edge in A is associated with a probability in P.

Many techniques can be used in order to perform the clustering:

- It is possible to use the probabilities as the weights on the edges. However, such an approach does not explicitly account for the connectivity of the underlying network and its interaction with the combinatorial nature of the underlying graph. Intuitively, a good cluster in the network is one which is hard to disconnect.
- The possible worlds model has been used in [40] in order to perform the clustering. The edit distance is used on the underlying network in order to perform the clustering. A connection is established with the problem of correlation clustering [13] in order to provide an approximation algorithm for the problem.
- The problem of graph clustering is explicitly connected to the problem of subgraph reliability in [35, 46]. The work in [35] determines methods for finding "reliable" subgraphs in uncertain graphs. These subgraphs are those which are hard to disconnect, based *on a combination of* the combinatorial structure of the graph, and the edge uncertainty probabilities. Thus, such an approach is analogous to the deterministic problem of finding dense subgraphs in deterministic graphs. However, it is not specifically focussed on the problem of *partitioning* the graph. A solution which finds reliable partitions from uncertain graphs is proposed in [46].

18.9 Conclusions and Summary

In this chapter, we discussed recent techniques for clustering uncertain data. The uncertainty in the data may either be specified in the form of a probability density function or in the form of variances of the attributes. The specification of the variance requires less modeling effort, but is more challenging from a clustering point of view. The problem of clustering is significantly affected by the uncertainty, because different attributes may have different levels of uncertainty embedded in them. Therefore, treating all attributes evenly may not provide the best clustering results. This chapter provides a survey of the different algorithms for clustering uncertain data. Most of the conventional classes of deterministic algorithms such as mixture modeling, density based algorithms, partitioning algorithms, streaming algorithms and high dimensional algorithms have been extended to the case of uncertain data. For the streaming and high dimensional scenarios, uncertain data also creates additional challenges because of the following reasons:

- In the streaming scenario, the uncertain data has additional volume. The distance calculations are also much slower in such cases.
- In the high dimensional scenario, the sparsity problem is exacerbated by uncertainty. This is because the uncertainty and noise reduces the correlations among the dimensions. Reduction of correlation between dimensions also results in an increase in sparsity.

We discussed several algorithms for the high dimensional and streaming case, which can be used for effective clustering of uncertain data.

Bibliography

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

[2] C.C. Aggarwal, and P. S. Yu. A Survey of Uncertain Data Algorithms and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 21(5), pp. 609–623, 2009.

[3] C. C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J.-S. Park. Fast Algorithms for Projected Clustering. In *ACM SIGMOD Conference*, 1999.

[4] C. C. Aggarwal. On Density Based Transforms for Uncertain Data Mining. In *ICDE Conference Proceedings*, 2007.

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

[6] C. C. Aggarwal, and P. S. Yu. Outlier Detection with Uncertain Data. In *SDM Conference*, 2008.

[7] C. C. Aggarwal. On High-Dimensional Projected Clustering of Uncertain Data Streams. In *ICDE Conference*, 2009.

[8] C. C. Aggarwal. On Unifying Privacy and Uncertain Data Models. In *ICDE Conference Proceedings*, 2008.

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

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

[11] R. Agrawal, and R. Srikant. Privacy-Preserving Data Mining. In *ACM SIGMOD Conference*, 2000.

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

[13] N. Bansal, A. Blum, and S. Chawla. Correlation clustering, *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, 2004.

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

[15] D. Burdick, P. Deshpande, T. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP Over Uncertain and Imprecise Data. In *VLDB Conference Proceedings*, 2005.

[16] M. Chau, R. Cheng, B. Kao, and J. Ng. Uncertain data mining: An example in clustering location data. In *PAKDD Conference*, pp. 199–204, 2006.

[17] A. L. P. Chen, J.-S. Chiu, and F. S.-C. Tseng. Evaluating Aggregate Operations over Imprecise Data. In *IEEE Transactions on Knowledge and Data Engineering*, 8(2), pp. 273–294, 1996.

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

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

[20] G. Cormode, and A. McGregor. Approximation algorithms for clustering uncertain data. In *PODS Conference*, pp. 191–200, 2008.

[21] N. Dalvi, and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *VLDB Conference Proceedings*, 2004.

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

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood for incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39: pp. 1–38, 1977.

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

[25] H. Garcia-Molina, and D. Porter. The Management of Probabilistic Data. *IEEE Transactions on Knowledge and Data Engineering*, vol. 4(5), pp. 487–501, 1992.

[26] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM TKDD Journal*, 1(1): 4, 2007.

[27] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. *ACM SIGMOD Conference*, 1998.

[28] S. Guha, and K. Munagala. Exceeding Expectations and Clustering Uncertain Data. In *ACM PODS Conference*, 2009.

[29] F. Gullo, G. Ponti, and A. Tagarelli. Minimizing the Variance of Cluster Mixture Models for Clustering Uncertain Objects. *IEEE ICDM Conference*, 2010.

[30] F. Gullo, and A. Tagarelli. Uncertain Centroid-based Partitional Clustering of Uncertain Data, *VLDB Conference*, 2012.

[31] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. A Hierarchical Algorithm for Clustering Uncertain Data via an Information-Theoretic Approach. *IEEE ICDM Conference*, 2008.

[32] S. Gunnemann, H. Kremer, and T. Seidl. Subspace Clustering for Uncertain Data. In *SIAM Conference on Data Mining*, 2010.

[33] H. Hamdan, and G. Govaert. Mixture model clustering of uncertain data. In *Proceedings of IEEE ICFS Conference*, pp. 879–884, 2005.

[34] A. Jain, and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1998.

[35] R. Jin, L. Liu, and C. Aggarwal. Finding Highly Reliable Subgraphs in Uncertain Graphs, *ACM KDD Conference*, 2011.

[36] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. M. Jermaine, and P. J. Haas. Mcdb: a monte carlo approach to managing uncertain data. In *ACM SIGMOD Conference*, 2008.

[37] L. Kaufman, and P. Rousseeuw. *Finding Groups in Data: An Intrduction to Cluster Analysis*. Wiley Interscience, 1990.

[38] R. Ng and J. Han. Efficient and Effective Clustering Algorithms for Spatial Data Mining. In *VLDB Conference*, 1994.

[39] B. Kao, S. D. Lee, D. W. Cheung, W. S. Ho, K. F. Chan. Clustering Uncertain Data using Voronoi Diagrams. In *IEEE ICDM Conference*, 2008.

[40] G. Kollios, M. Potamias, and E. Terzi. Clustering Large Probabilistic Graphs, *IEEE TKDE Journal*, to appear.

[41] M. Kumar, N. Patel, and J. Woo. Clustering seasonality patterns in the presence of errors. In *ACM KDD Conference Proceedings*, pp. 557–563, 2002.

[42] H.-P. Kriegel, and M. Pfeifle. Density-Based Clustering of Uncertain Data. In *ACM KDD Conference Proceedings*, 2005.

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

[44] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian. ProbView: A Flexible Probabilistic Database System. *ACM Transactions on Database Systems*, 22(3), pp. 419–469, 1997.

[45] S. D. Lee, B. Kao, and R. Cheng. Reducing UK-means to K-means. In *ICDM Workshops*, 2006.

[46] L. Liu, R. Jin, C. Aggarwal, and Y. Shen. Reliable Clustering on Uncertain Graphs, *ICDM Conference*, 2012.

[47] S. I. McClean, B. W. Scotney, and M. Shapcott. Aggregation of Imprecise and Uncertain Information in Databases. *IEEE Transactions on Knowledge and Data Engineering*, 13(6), pp. 902–912, 2001.

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

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

[50] M. Sato, Y. Sato, and L. Jain. *Fuzzy Clustering Models and Applications*. Physica–Verlag, Heidelberg, 1997.

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

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

[53] P. Volk, F. Rosenthal, M. Hahmann, D. Habich, and W. Lehner. Clustering Uncertain Data with Possible Worlds. *ICDE Conference*, 2009.

[54] L. Xiao, and E. Hung. An Efficient Distance Calculation Method between Uncertain Objects, *CIDM Conference*, 2007.

[55] C. Zhang, M. Gao, and A. Zhou. Tracking High Quality Clusters over Uncertain Data Streams, *ICDE Conference*, 2009.

[56] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *ACM SIGMOD Conference Proceedings*, 1996.