A Framework for Clustering Uncertain Data Streams

Charu C. Aggarwal, Philip S. Yu

IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532, USA
{ charu, psyu }@us.ibm.com

***Abstract*— In recent years, uncertain data management applications have grown in importance because of the large number of hardware applications which measure data approximately. For example, sensors are typically expected to have considerable noise in their readings because of inaccuracies in data retrieval, transmission, and power failures. In many cases, the estimated error of the underlying data stream is available. This information is very useful for the mining process, since it can be used in order to improve the quality of the underlying results. In this paper we will propose a method for clustering uncertain data streams. We use a very general model of the uncertainty in which we assume that only a few statistical measures of the uncertainty are available. We will show that the use of even modest uncertainty information during the mining process is sufficient to greatly improve the quality of the underlying results. We show that our approach is more effective than a purely deterministic method such as the *CluStream* approach. We will test the approach on a variety of real and synthetic data sets and illustrate the advantages of the method in terms of effectiveness and efficiency.**

I. Introduction

In recent years, the problem of mining uncertain data sets has gained importance because of its numerous applications to a wide variety of problems [1], [8], [9], [10], [5], [16]. This is because data collection methodologies are often inaccurate and are based on incomplete or inaccurate information. For example, sensor data sets are usually noisy and lead to a number of challenges in processing, cleaning and mining the data. Some techniques for adaptive cleaning of such data streams may be found in [15]. In many cases, estimations of the uncertainty in the data are available from the methodology used to measure or reconstruct the data. Such estimates may either be specified in the form of error variances [1] or in the form of probability density functions [8]. In general, a variety of online methods exist [18] to estimate missing data values along with the corresponding errors. In such cases, this results in streams of uncertain data. Some examples in which the uncertainty of the data are available are as follows:

- In cases such as sensor streams, the errors arise out of inaccuracies in the underlying data collection equipment. In many cases, the values may be missing and statistical methods [18] may need to be used to impute these values. In such cases, the error of imputation of the entries may be known a-priori.
- In many temporal and stream applications, quick statistical forecasts of the data may be generated in order to perform the mining. In [2], a technique has been discussed to construct forecasted pseudo-data streams for mining and querying purposes. In such cases, the statistical uncertainty in the forecasts is available.
- In many applications, the data is available only on a partially aggregated basis. In privacy-preserving data mining, uncertainty may be added to the data in order to preserve the privacy of the results. For example, in some perturbation based methods [4], the data points are perturbed with the use of a probability distribution. In such cases, the exact level of uncertainty in the data is available.

The results of data mining algorithms can often be sensitive to the errors in the underlying data. For example, an attribute with a large amount of error is less reliable for data mining purposes than one with smaller error. This can affect the quality of the underlying results for data mining applications, since different attributes and records may need to be treated differently for mining purposes. We will use a flexible model of uncertainty in which we only assume that the standard error of the underlying data point is known. This is less restrictive than models in which it is assumed that the entire probability distribution function of the data is known. In many real applications, only limited uncertainty information (such as the standard error) may be available. This is especially true of stream based applications in which online techniques are used in order to perform data estimation and collection. Furthermore, measures such as the standard error are much more compact from a representational point of view. This is important from the efficiency perspective in data stream applications.

In this paper, we will explore the uncertain clustering problem for the case of data streams. Data streams pose a special challenge because the data records can be processed at most once during the entire computation. Furthermore, the uncertainty information needs to be processed simultaneously with the attribute information. Uncertain data sets are also prevalent in stream applications, because typical data recording methods such as sensors often have noisy or incomplete recording mechanisms [15].

The problem of clustering [12], [21], [23] has been studied extensively in the database literature because of its numerous applications to problems in customer segmentation, target marketing, and classification. Detailed discussions on a variety of classical and modern clustering algorithms may be found in [13]. Recently, the approach has also been extended to the problem of clustering data streams [3], [6]. In the case of data streams, the efficiency of the computation process is a key issue, since a data point cannot be processed more than once during the course of the entire computation. In [3], a more flexible method for clustering was proposed, which allows the user to specify different kinds of input parameters (such as a historical time horizon) in order to perform the analysis. In this paper, we will propose an effective approach with the same high level of flexibility for the uncertain version of the problem. The problem of clustering uncertain data streams is especially challenging because the uncertainty in attribute values can significantly affect the clustering behavior of the data points. For example, the distance computations for a dimension with a higher level of uncertainty need to be treated differently than one with lower uncertainty because of the noise effects of the underlying error. Similarly, many other computations such as the definition of a cluster boundary are affected by the probabilistic nature of the data points. Furthermore, any such probabilistic computations need to be performed in real time, because of the large volume of the streaming data. In this paper, we will use a number of summary structures to track the statistics of the stream in real time and leverage it for the clustering process. We will also design a method for adapting the approach to evolving data streams with a decay-based approach.

This paper is organized as follows. In the next section, we will propose an algorithm for clustering uncertain data streams. We will show how the uncertain computations can be performed in real time by using an intermediate representation for the micro-clusters. In section 3, we will discuss the experimental results. Section 4 contains the conclusions and summary.

II. Clustering Uncertain Streams

In this section, we will introduce $UMicro$, the Uncertain MICROclustering algorithm for data streams. We will first introduce some additional notations and definitions. 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 many techniques in the uncertain data management literature [8] work with the assumption that the entire probability density function is available. We make the more modest assumption that the standard error of individual entries is available. In many real applications, this is a more realistic assumption for data mining purposes, since complete probability distributions are rarely available, and are usually inserted only as a modeling assumption. The interpretation of this error value can vary with the nature of the data mining application. 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.

We will develop a method for clustering uncertain data streams with the use of a micro-clustering model. The micro-clustering model was first proposed in [23] for large data sets, and subsequently adapted in [3] for the case of deterministic data streams. We will see that the uncertainty in the underlying data significantly affects the quality of the clusters with methods that use such error information.

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 2.1: An uncertain micro-cluster for a set of d-dimensional points $\overline{X_{i_1}} \ldots \overline{X_{i_n}}$ with time stamps $T_{i_1} \ldots T_{i_n}$ and error vectors $\overline{\psi(X_{i_1})} \ldots \overline{\psi(X_{i_n})}$ is defined as the $(3 \cdot d + 2)$ 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

Algorithm *UMicro*(Number of Clusters: n_{micro})
begin
$S = \{\}$;
{ S is the current set of micro-cluster
statistics. S contains at most n_{micro}
elements }
repeat
Receive the next stream point $\overline{X}$;
{ Initially, when S is null, the computations below
cannot be performed, and $\overline{X}$ is simply
added as a singleton micro-cluster to S }
Compute the *expected similarity* of $\overline{X}$ to the closest
micro-cluster $\mathcal{M}$ in S;
Compute critical uncertainty boundary of $\mathcal{M}$;
if $\overline{X}$ lies inside uncertainty boundary
add $\overline{X}$ to statistics of $\mathcal{M}$
else
add a new micro-cluster to S containing singleton
point $\overline{X}$;
if $|S| = n_{micro} + 1$ remove the least recently
updated micro-cluster from S;
until data stream ends;
end

Fig. 1. The UMicro Algorithm

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* [3] which is critical to its use in the clustering process. We restate the additive property as follows:

Property 2.1: 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.

A. The UMicro Algorithm: Overview

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 1. Next, we will discuss the process of computation of individual subroutines such as the expected distance or the uncertain boundary.

B. Computing Expected Similarity

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

Lemma 2.1: Let $\overline{Z}$ be the random variable representing the centroid of cluster $\mathcal{C}$. Then, the following result holds true:

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

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

$$Z_j = CF1(\mathcal{C})_j/n(\mathcal{C}) + \sum_{\overline{X} \in \mathcal{C}} e_j(\overline{X})/n(\mathcal{C}) \quad (2)$$

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

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

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:

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

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

$$E[||Z||^2] = \sum_{j=1}^{d} CF1(\mathcal{C})_j^2/n(\mathcal{C})^2 + \sum_{j=1}^{d} EF2(\mathcal{C})_j/n(\mathcal{C})^2 \quad (3)$$

This proves the desired result. ■

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

Lemma 2.2: 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 $\mathcal{C}$. Then, v is given by the following expression:

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

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

$$\begin{aligned} v &= E[||\overline{X} - \overline{Z}||^2] \\ &= E[||\overline{X}||^2] + E[||\overline{Z}||^2] - 2E[\overline{X} \cdot \overline{Z}] \\ &= E[||\overline{X}||^2] + E[||\overline{Z}||^2] - 2E[\overline{X}] \cdot E[\overline{Z}] \text{(indep. of } \overline{X} \text{ and } \overline{Z}) \end{aligned}$$

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:

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

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

$$E[X] \cdot E[Z] = \sum_{j=1}^{d} x_j \cdot CF1(\mathcal{C})_j/n(\mathcal{C}) \quad (4)$$

The results above and Lemma 2.1 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:

$$v = E[||\overline{X}||^2] + E[||\overline{Z}||^2] - 2E[\overline{X}] \cdot E[\overline{Z}] \quad (5)$$

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

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

The result follows. ■

The result of Lemma 2.2 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 $\mathcal{C}$. 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 [23]. 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.

C. Computing the Uncertain Boundary

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 $\mathcal{C}$, 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:

$$U = \sum_{i=1}^{r} \sum_{j=1}^{d} E[||Y_i - W||_j^2] \tag{6}$$

The expression on the right hand side of the above Equation can be evaluated by using the relationship of Lemma 2.2.

D. Storage

The behavior of the cluster statistics are stored at specific instants of time, which are defined as *snapshots*. A natural solution is to store the snapshots at evenly spaced interval. This can be overly expensive, since the clustering behavior is usually analyzed only over a horizon which includes the most recent data. In order to optimize the storage for such time horizons, we use the pyramidal pattern discussed in [3]. In the pyramidal pattern, snapshots are classified into a set of orders, which define their separation from one another. The properties of the snapshots are as follows:

- Snapshots of the ith order are separated at distances of α^i.
- For some integer $l \geq 1$, there are α^l snapshots of order i. The value of l is fixed across snapshots of different orders.

It has been shown in [3], that for any user defined horizon h, it is possible to find a snapshot at h', such that the following is true:

$$\frac{|h' - h|}{h} \leq 1/\alpha^l \tag{7}$$

This means that any horizon can be estimated to a very high degree of accuracy. This is a useful property since it means that by using the snapshot at h' in conjunction with the subtractivity property, it is possible to construct the micro-clusters for any specific time horizon. Therefore, if the current time is t_c, then the micro-clusters for the period $(t_c - h', t_c)$ are determined using the subtractivity property. Let $S(t_c)$ be the micro-cluster statistics at time t_c, and let $S(t_c - h')$ be the micro-cluster statistics at time $t_c - h'$. Then, the statistics for each micro-cluster in $S(t_c - h')$ is subtracted from the statistics of the corresponding micro-clusters in $S(t_c)$. Micro-clusters which are removed by the clustering algorithm in period $(t_c - h', t_c)$ are discarded, and micro-clusters which are created in the period $(t_c - h', t_c)$ are retained in their current form in the subtraction process. These horizon-specific macro-clusters can be used in order to perform other kinds of operations such as finding higher level macro-clusters. Thus, as in [3], the approach can be used to perform interactive and online clustering in a data stream environment.

E. Incorporating Time Decay

It is possible to incorporate time decay into the clustering process, by weighting points appropriately with the use of a temporal decay function. The temporal decay function defines the weight of a data point during the clustering process. We will use an exponential decay function in order to define the weights of the different data points for constructing the micro-clusters. Therefore, the weight $w_t(\overline{X})$ of a data point $\overline{X}$ which arrives at time $t(\overline{X})$ is defined as follows:

$$w_t(\overline{X}) = 2^{-\lambda \cdot (t_c - t(\overline{X}))} \tag{8}$$

Here λ is the decay rate. The half life of a data point $\overline{X}$ as follows:

Definition 2.2: The half-life of a data point $\overline{X}$ is the time in which the weight of the data point reduces by half for the clustering process, and is equal to $1/\lambda$.

In order to incorporate time decay, we modify the definition of the micro-clusters as follows:

Definition 2.3: An weighted error-based micro-cluster for a set of d-dimensional points $X_{i_1} \ldots X_{i_n}$ with time stamps $T_{i_1} \ldots T_{i_n}$, error vectors $\overline{\psi(X_{i_1})} \ldots \overline{\psi(X_{i_n})}$ and time decay weights $w_t(X_{i_1}) \ldots w_t(X_{i_n})$ is defined as the $(3 \cdot d + 2)$ tuple $(\overline{CF2^x}(\mathcal{C}), \overline{EF2^x}(\mathcal{C}), \overline{CF1^x}(\mathcal{C}), t(\mathcal{C}), W(\mathcal{C}))$, 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 weighted 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 $\sum_{j=1}^{n} w_t(X_{i_j}) \cdot (x_{i_j}^p)^2$. This corresponds to the weighted second moment of the data values along the p-th dimension.
- For each dimension, the weighted 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 $\sum_{j=1}^{n} w_t(X_{i_j}) \cdot \psi_p(X_{i_j})^2$. This corresponds to the weighted sum of squares of the errors in the records along the p-th dimension.
- For each dimension, the weighted 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 $\sum_{j=1}^{n} w_t(X_{i_j}) \cdot x_{i_j}^p$. This corresponds to the weighted first moment of the values along the p-th dimension.
- The total weight of points in the data is maintained in $W = \sum_{j=1}^{n} w_t(X_{i_j})$.
- The time stamp of the last update to the micro-cluster is maintained in $t(\mathcal{C})$.

We note that all the algorithmic methods discussed in this paper can be easily extended to the case of the weighted case discussed here. A lazy approach is used to update the

continuous time decay statistics. The objective of the lazy approach is to reduce the update time for the continuously decaying statistics in the micro-clusters. A strict approach of always maintaining the precisely decayed values would require updating of the micro-cluster statistics at each time instant. This can be wasteful in practice. In the lazy approach, the time decay factor for a micro-cluster is updated only when it is modified by the addition of a new data point to the micro-cluster. Because of the exponential decay, we only need to keep track of the last time instant t_s at which the micro-cluster was updated, and then multiply all statistics in the vector (other than update time) by the factor $2^{-\lambda \cdot (t_c - t_s)}$. This is because all points decay at the same rate of $2^{-\lambda}$ for each time instant elapsed. Since $(t_c - t_s)$ instants have been elapsed since the last update time, all statistics in the micro-clusters need to be updated at the rate of $2^{-\lambda \cdot (t_c - t_s)}$. This kind of approach does not affect the complexity of the algorithm significantly while maintaining modestly accurate statistics for computation purposes.

Another key pair of differences is in computing the assignment of points to clusters, and updating the clusters. The process of updating the clusters is similar to the previous case, except that we need to use the weighted data points during the addition process. This can easily be achieved by first updating the time decay weights in the remaining micro-clusters, and then adding the newly arrived point to the micro-cluster statistics. It can be shown that the results of Lemma 2.1 and 2.2 can be easily extended to the weighted case. The only difference is that the corresponding terms are scaled with the squares of the weights of the original data points. One observation is that the modifications for the time decayed approach do not significantly affect the complexity of the method. This is because the most computationally expensive operation is the updating of the micro-cluster statistics, which does not asymptotically change because of incorporation of time decay.

III. Experimental Results

In this section, we will present the experimental results for the clustering method for uncertain data streams. We will compare our technique against the micro-clustering method, and show how the uncertain data model can be more effective in such situations. For the purpose of testing, we used perturbations of a number of different data sets. For each data set, we added noise which was drawn from a gaussian distribution with zero mean and standard deviation which was determined as follows. We used a noise parameter η to determine the amount of noise to be added to each dimension in the data. The noise was added as factor of the number of standard deviations σ_i^0 of the entire data along the dimension i. In order to model the difference in noise level across different dimensions, we first defined the standard deviation σ_i along dimension i as a uniform random variable drawn from the range $[0, 2 \cdot \eta \cdot \sigma_i^0]$. Then, for the dimension i, we add error from a random distribution with standard deviation σ_i. Thus, by varying the value of η, it was possible to vary the amount of noise in the data. We also note that a choice of $\eta \geq 3$ typically creates a data set with a high level of noise that obscures most of the underlying patterns in the data.

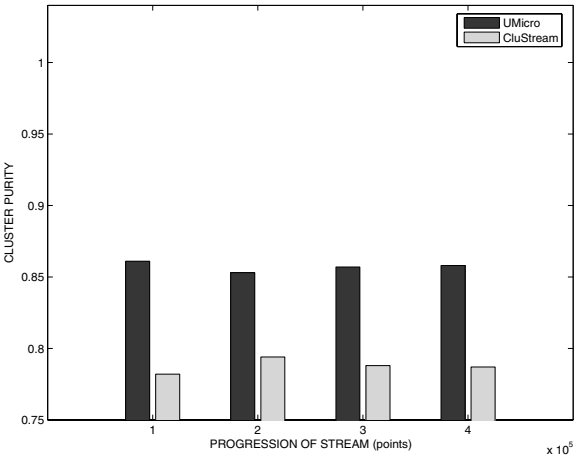

Fig. 2. Accuracy with Progression of Stream (Syndrift Data Set, $\eta = 0.5$)

We tested the method on both real and synthetic data sets. The synthetic data sets were generated using continuously drifting clusters. The relative fraction of data points which belong to the cluster i is denoted by f_i. The relative value of f_i is drawn as a uniform random variable in the range $[0, 1]$. The radius of each cluster along a given dimension is given by a random variable which is drawn from the range $(0, 1)$. The centroids of each of the clusters are initially chosen from the unit cube. Subsequently, each centroid drifts along a dimension by an amount which is drawn from the uniform distribution in the range $[-\epsilon, \epsilon]$. The radius of each cluster along a given dimension is chosen as a variable which is picked as an instantiation of the uniform random variable in the range $[0, 0.3]$. A 20-dimensional data stream containing 600,000 points was generated using this methodology. The data set was referred to as *SynDrift* corresponding to the fact that it is a synthetic data set with drifting clusters. We note that the data set is further perturbed with the addition of noise. Therefore, a data set with noise level of η is referred to as $SynDrift(\eta)$.

The Network Intrusion Detection data set consists of a series of TCP connection records from two weeks of LAN network traffic managed by MIT Lincoln Labs. Each n record can either correspond to a normal connection, or an intrusion or attack. The attacks fall into four main categories: DOS (i.e., denial-of-service), R2L (i.e., unauthorized access from a remote machine), U2R (i.e., unauthorized access to local superuser privileges), and PROBING (i.e., surveillance and other probing). As a result, the data contains a total of five clusters including the class for "*normal connections*". It is evident that each specific attack type can be treated as a sub-cluster. Most of the connections in this data set are *normal*, but occasionally there could be a burst of attacks at certain times. Also, each connection record in this data set contains 42 attributes, such as duration of the connection, the number

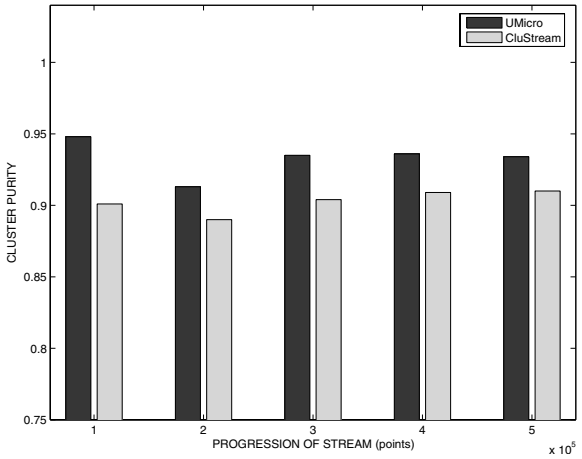

Fig. 3. Accuracy with Progression of Stream (Network Intrusion Data Set, $\eta = 0.5$)

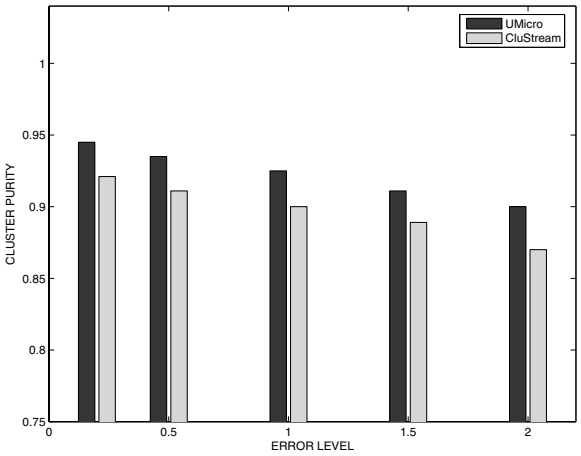

Fig. 6. Accuracy with Increase in Error Level (Network Intrusion Data Set)

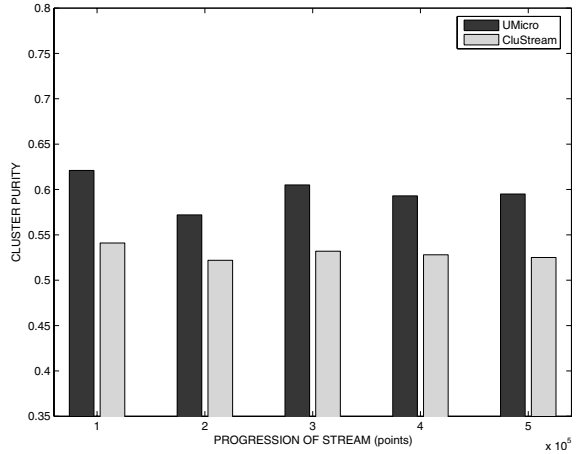

Fig. 4. Accuracy with Progression of Stream (Charitable Donation Data Set, $\eta = 0.5$)

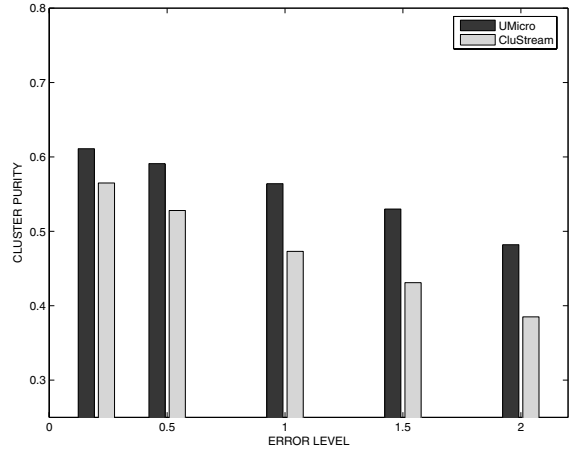

Fig. 7. Accuracy with Increase in Error Level (Forest Cover Data Set)

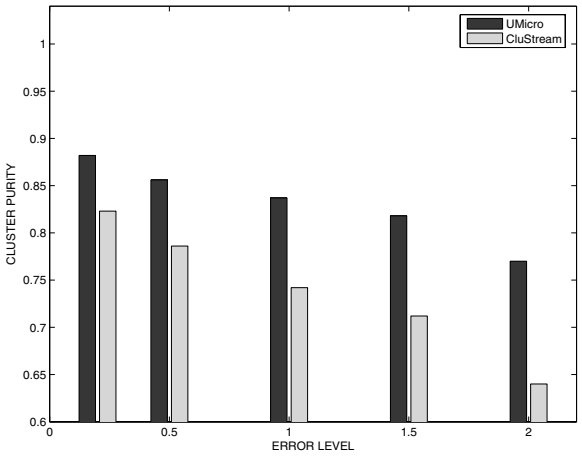

Fig. 5. Accuracy with Increase in Error Level (Syndrift Data Set)

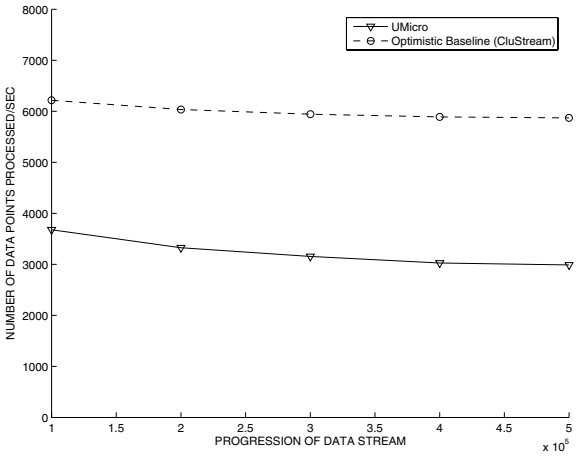

Fig. 8. Efficiency of Stream Clustering (Syndrift Data Set)

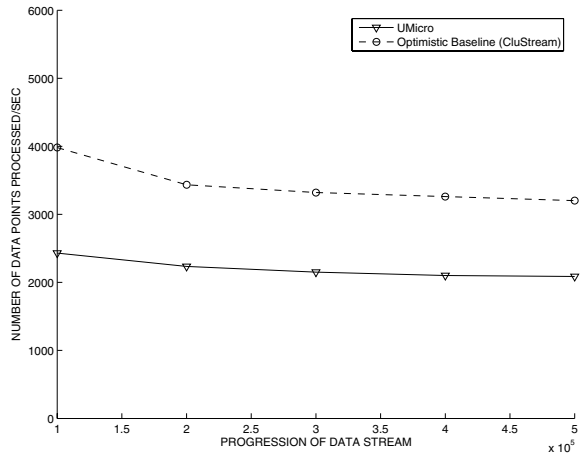

Fig. 9. Efficiency of Stream Clustering (Network Intrusion Data Set)

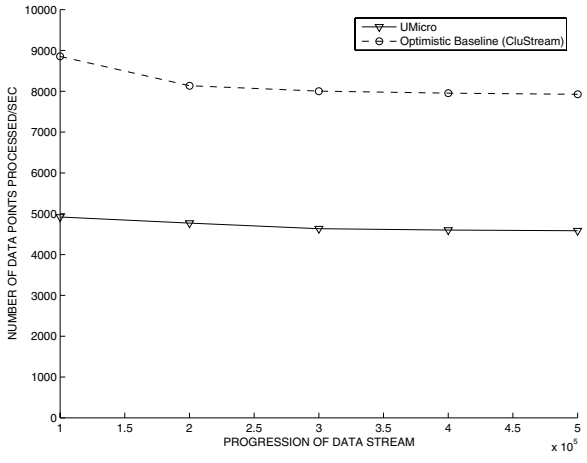

Fig. 10. Efficiency of Stream Clustering (Forest Cover Data Set)

of data bytes transmitted from source to destination (and vice versa), percentile of connections that have "SYN" errors, the number of "root" accesses, etc. As in [3], all 34 continuous attributes will be used for clustering and one outlier point has been removed. The modified Network Intrusion data set with noise level of η was referred to as $Network(\eta)$.

The last real data set we tested is the *Forest CoverType* data set and was obtained from the UCI machine learning repository web site [20]. This data set contains 581012 observations and each observation consists of 54 attributes, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. In our testing, we used all the 10 quantitative variables. This data set is converted into a data stream by taking the data input order as the order of streaming and assuming that they flow-in with a uniform speed. As in previous cases, the Forest Cover data set with added noise level of η was referred to as $ForestCover(\eta)$.

We tested the approach for effectiveness and efficiency. In terms of effectiveness, we tested the following:

- We computed the clustering accuracy with progression of the data stream, and compared against the standard stream clustering algorithm known as $CluStream$ [3]. The clustering accuracy was measured using the class labels associated with the clusters. In the case of the synthetic data sets, the class label was assumed to be the cluster identifier. We computed the percentage presence of the dominant class label in the different clusters and averaged them over all clusters. We refer to this measure as *cluster purity*, and it provides us an idea of the cluster effectiveness.
- We computed the clustering accuracy for different levels of noise. This was done by varying the noise level η. The variation in accuracy provides us an idea of how the clustering process is affected by the error both for the deterministic *CluStream* algorithm, as well as the uncertain mining algorithm presented in this paper.
- We measured the efficiency of the uncertain stream clustering method. This efficiency was tested using the number of data stream points processed per second, when the algorithm was run continuously. This provides us an idea of the maximum throughput that can be processed by the mining algorithm.

In Figures 2, 3, and 4, we have illustrated the effectiveness of the approach with progression of the stream. On the X-axis, we have illustrated the progression of the data stream in terms of the number of points. In each case, we used an error level of $\eta = 0.5$. We have illustrated for the effectiveness for both the *UMicro* and *CluStream* methods with the use of 100 micro-clusters. It is clear that in each case, the *UMicro* method provided superior accuracy to the *CluStream* method. This advantage in accuracy remained over the progression of the entire data stream. In the case of the network intrusion data set, the advantage was a little less, since most of the connections were normal connections, and therefore clusters were naturally dominated by data points corresponding to normal connections even in the case in which errors were introduced. In both the cases of the *SynDrift* and *Forest Cover* data sets, the accuracy advantage of the $UMicro$ method was quite high, and could often be greater than 10%. The advantage of the *UMicro* method follows from the fact that the algorithm uses the uncertainty of the data points in order to decide assignment of points to clusters.

We also tested the effectiveness of the method with increasing error level. In Figures 5, 6, and 7, we have illustrated the variation in effectiveness with increasing error level. On the X-axis, we have illustrated the error level. On the Y-axis, we have illustrated the accuracy of the method over the entire data stream. It is clear that in each case, the accuracy reduced with increasing error level of the underlying data stream. Furthermore, the gap between the two data sets increases with increasing level of error. The gap increase was more pronounced in the case of the *Forest Cover* and *SynDrift* data sets. This was because the class distributions were more diverse in the case of the *SynDrift* and *Forest Cover* data sets as compared to the *Network Intrusion* data set, which mostly contained normal connections.

We also tested the efficiency of the stream clustering method. All results were tested on an IBM T41p Thinkpad running the Windows XP operating system, with a 1.69 GHz processor and 1 GB of main memory. In Figures 8, 9, and 10 we have illustrated the efficiency of the clustering method on the different data sets. On the X-axis, we have illustrated the progression of the data stream in terms of the number of points, whereas on the Y-axis, we have illustrated the number of points processed per second at particular points of the data stream progression. This number was computed by using the average number of points processed per second in the last 2 seconds. We have also illustrated the $CluStream$ method as a baseline in each case. We note that the $CluStream$ method is an optimistic baseline, since it works with deterministic data in which both the data size is smaller (because of absence of error information), and the computations are much simpler. On the X-axis, we have illustrated the progression of the data stream, whereas on the Y-axis, we have illustrated the processing rate in terms of the number of data points per second. It is clear that the $UMicro$ method is able to process thousands of points per second in each case. Furthermore, while the $CluStream$ method needs to perform a much smaller number of computations, the $UMicro$ method is able to achieve about $50-70\%$ of the speed of this optimistic baseline. We note that this is quite modest considering the fact that even the input size of the $UMicro$ method is twice as large as the $CluStream$ method because of the addition of error information. Correspondingly, the processing and computational requirements are also much greater. Thus, the $UMicro$ method is not only effective, but is also a very efficient clustering method for data streams.

A. Related Work

The problem of clustering has been well studied in the database, statistics and machine learning communities [13]. Detailed discussions of a variety clustering methods may be found in [13]. Subsequently, the clustering problem was also studied in the context of large databases [12], [21], [23]. Since, the process of reading data is an expensive database operation, most of these techniques are designed with a particular focus on disk I/O requirements. For example, in the case of the CLARANS method [21], sampling is used to improve efficiency, whereas in the case of the BIRCH method [23], we use summary structures in order to improve efficiency. In the case of data streams, these requirements become even more critical, since only one pass over the data is required for computation purposes. The first well known method for clustering data streams was proposed in [6]. This technique extended the well known k-means method for data stream computation. Subsequently, a method was proposed for more flexible clustering analysis using the micro-cluster methodology of [23]. This technique is also able to support analysis of clustering trends in data streams.

The problem of uncertain data mining has recently gained attention because of numerous classical applications to missing data estimation [18] and numerous recent applications such as sensor networks [15]. In [15], a method has been proposed for cleaning RFID data streams. A variety of frameworks for uncertain data management are discussed in [11], [17], [10]. In particular, the work in [10] discusses a number of working models for uncertain data management. Subsequently, methods were developed for supporting database operations over uncertain data. A particularly well studied problem in the context of uncertain data is that of aggregate query estimation [5], [7], [8], [19]. The earliest work on uncertain data query processing was proposed in [7]. Recently, this problem has also been studied in the database context in terms of OLAP query processing [5], as well as the problem of indexing and querying large uncertain data sets [8]. In particular, the technique in [8] is able to support probabilistic threshold queries over uncertain data.

Some recent work [16], [14], [22] discusses the problem of clustering error-prone data, though this technique is not designed for the case of data streams. In [16], a density based clustering method for uncertain data sets was developed. In [14] a method for clustering seasonality patterns in the presence of errors was proposed. Neither of the two methods can be easily extended to the case of data streams. Recently, a density-based framework [1] was also proposed for mining uncertain data sets with the use of sanitized intermediate representations. This technique was applied to the case of the classification problem, and it was shown that the use of uncertainty information greatly improves the effectiveness of the mining process.

IV. Conclusions and Summary

In this paper, we proposed the UMicro algorithm for clustering uncertain data streams. Uncertain data streams are often likely to be present in many real applications because of inaccurate recording mechanisms. The uncertainty in the data stream significantly affects the clustering process, because of the varying relative behavior of the different attributes and its effect on the distance computations. We extended the approach to provide historical trends with the use of a pyramidal time frame. We also discussed how to incorporate decay into the clustering process. The decay based technique is especially useful for the evolving data streams in which the underlying patterns may change over time. We tested the technique against the CluStream algorithm for effectiveness, and showed that the UMicro has significantly higher accuracy. This accuracy advantage increases with increasing levels of error. In addition, the method is extremely efficient and can be used for very fast data streams.

References

[1] C. C. Aggarwal, "On Density Based Transforms for Uncertain Data Mining," in *ICDE Conference Proceedings*, 2007.

[2] C. C. Aggarwal, "On Futuristic Query Processing in Data Streams," in *EDBT Conference Proceedings*, 2006.

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

[4] R. Agrawal, and R. Srikant, "Privacy-Preserving Data Mining," in *ACM SIGMOD Conference Proceedings*, 2000.

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

[6] L. O'Callaghan, A. Meyerson, R. Motwani, N. Mishra, and S. Guha, "Streaming-Data Algorithms for High-Quality Clustering," in *ICDE Conference Proceedings*, 2002.

[7] 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*, vol. 8, no. 2, pp. 273–294, 1996.

[8] 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.

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

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

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

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

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

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

[15] S. R. Jeffery, M. Garofalakis, and M. J. Franklin, "Adaptive Cleaning for RFID Data Streams," in *VLDB Conference Proceedings*, 2006.

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

[17] L. V. S. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian, "ProbView: A Flexible Probabilistic Database System," in *ACM Transactions on Database Systems*, vol. 22, no. 3, pp. 419–469, 1997.

[18] R. Little, and D. Rubin, "Statistical Analysis with Missing Data Values," *Wiley Series in Prob. and Stats.*, 1987.

[19] S. I. McClean, B. W. Scotney, and M. Shapcott, "Aggregation of Imprecise and Uncertain Information in Databases," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 902–912, 2001.

[20] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, *UCI Repository of machine learning databases*,
URL: [http://www.ics.uci.edu/˜mlearn/MLRepository.html].

[21] R. T. Ng, and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," in *VLDB Conference Proceedings*, 1994.

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

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