Outlier Detection on Uncertain Data: Objects, Instances, and Inferences

Bin Jiang
Simon Fraser University
Burnaby, BC, Canada
bjiang@cs.sfu.ca

Jian Pei
Simon Fraser University
Burnaby, BC, Canada
jpei@cs.sfu.ca

***Abstract*—This paper studies the problem of outlier detection on uncertain data. We start with a comprehensive model considering both uncertain objects and their instances. An uncertain object has some inherent attributes and consists of a set of instances which are modeled by a probability density distribution. We detect outliers at both the instance level and the object level. To detect outlier instances, it is a prerequisite to know normal instances. By assuming that uncertain objects with similar properties tend to have similar instances, we learn the normal instances for each uncertain object using the instances of objects with similar properties. Consequently, outlier instances can be detected by comparing against normal ones. Furthermore, we can detect outlier objects most of whose instances are outliers. Technically, we use a Bayesian inference algorithm to solve the problem, and develop an approximation algorithm and a filtering algorithm to speed up the computation. An extensive empirical study on both real data and synthetic data verifies the effectiveness and efficiency of our algorithms.**

I. Introduction

Uncertainty is inherent in data collected in various applications, such as sensor networks, marketing research, and social science. Uncertain data poses significant challenges for data analytic tasks. In this paper, we investigate a fundamental data mining problem – outlier detection – on uncertain data.

Example 1 (Motivation – product evaluation): A digital camera manufacturer carries a whole series of products. Each product has some inherent properties, such as pixel density, sensor type, and aperture range. The similarity between two products can be measured by the proximity of their inherent properties.

By analyzing customer evaluations on individual products, the manufacturer can study the market and adjust its strategies on marketing and product development. One product may receive multiple evaluations which may vary to one another. The reviews of the product are not certain and can be modeled as an uncertain object, where each evaluation is regarded as an instance. Similar products are expected to receive similar evaluation grades from customers.

For a product, some evaluations may be very different from the majority. Those outlier evaluations need to be examined. They may be interesting if they capture some issues in rare scenarios or reflect opinions of specific user groups. They may be excluded from analysis if they are noise, such as spam reviews. Some products may receive evaluations very different from similar products. Those outlier products are particularly interesting since they may provide important hints on customer/market interests.

Clearly, outlier detection on uncertain data at both the instance level and the object level is practically useful. ■

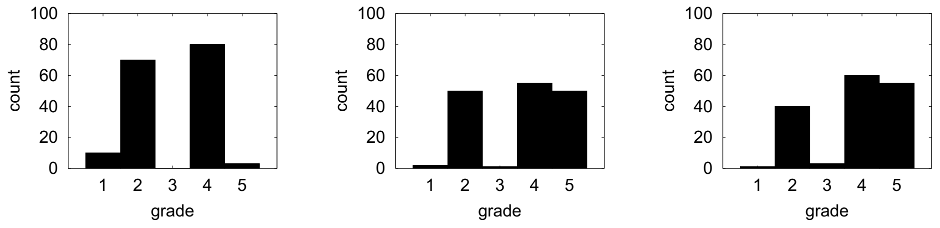
Example 2 (Motivation – sensor data): As an extensively adopted approach for environment surveillance, sensors are deployed to cover an area of interest. Each sensor keeps reporting its readings to monitor several sensing measures at its location. Each sensor has some inherent properties, such as its spatial location. The spatial distance between two sensors can be measured.

In many applications, the target sensing measure such as temperature can be regarded stable in a short period such as 30 minutes. However, the true value of temperature cannot be accurately obtained due to limitations of measuring equipment. Instead, sensor readings are collected in order to approximate the true temperature. So the true value of the temperature at a certain location is an uncertain object, where multiple readings collected from a sensor at this location are the instances of this uncertain object. Often, it is expected that the target sensing measures at nearby locations are similar, so are the readings of the sensors.

For a sensor, some readings may be very different from the true values of the target sensing measure due to factors like dynamic errors, drifts, and noise. It is important to clean those outlier readings to improve the accuracy of the sensor. Moreover, some sensors may deviate significantly from their neighbors. Such outlier sensors may be caused by malfunctioning sensor units or sensors at specific locations such as a deep hole on the ground.

Again, detecting outliers from uncertain data at both the instance level and the object level is meaningful. ■

One may think that outlier detection on uncertain data is straightforward – we can first detect and remove outlier instances within each object, and then detect outlier objects using aggregates (such as mean or median) of objects as the representatives. Unfortunately, such a simple approach may not work well in practice. Critically, instances in an uncertain object may follow a probability mixture model. For example, customer evaluations on a product, instead of converging to an unanimous agreement, may likely be a mixture of opinions from multiple groups. Therefore, using aggregates such as mean or median may not represent an object properly for outlier detection at the object level. Moreover, within one



(a) A (b) B (c) C

Fig. 1. An example of three proximate products.

object, outlier detection without considering the neighbor objects may lose useful information.

Example 3: Suppose we have three products, A, B, and C, which are similar to each other. Figure 1 shows the distribution of customer evaluations on the three products. If we only consider product A, the evaluations of grade 5 are likely to be regarded as outliers. However, considering that similar products B and C receive a substantial amount of evaluations of grade 5, the evaluations of grade 5 for product A should not be considered as outliers. Instead, the evaluations of grade 1 should be the suspects of outliers, since they are rare for products B and C. ■

In Section VI, we will empirically demonstrate that the straightforward method cannot detect outliers satisfactorily.

In order to detect outliers at both the object level and the instance level, we need a comprehensive model of uncertain objects and their instances. In this paper, we take a general model where an uncertain object is associated with some inherent properties described by a set of *conditioning attributes* and a set of instances which are described by a set of *dependent attributes*. For example, we can model the target measures of a sensor as an uncertain object. The factors determining the target measures of the sensor, such as latitude, longitude, terrain, and altitude of the location of the sensor, are the conditioning attributes. The readings are instances with dependent attributes like temperature and precipitation which are modeled as probability distributions [1], [2], [3]. Essentially, we model the conditioning attributes as latent variables that generate the dependent attributes. Figure 2 shows the graphical illustration.

In order to determine outlier instances, we first need to know normal instances. In many applications, it is reasonable to assume that objects with similar properties tend to have similar instance distributions. The central idea of our outlier detection technique is to learn the normal instances of each object by taking into account the instances of objects with similar properties.

Under this model, our outlier detection approach learns the conditional distribution of dependent attributes given the conditioning attributes. We measure the normality, which is the opposite of outlierness, of an instance by its conditional probability. After finding outlier instances, we can then detect outlier objects most of whose instances are outliers. The normality of an object is defined as the geometric mean of the normality of all its instances, which is meaningful in probability theory.

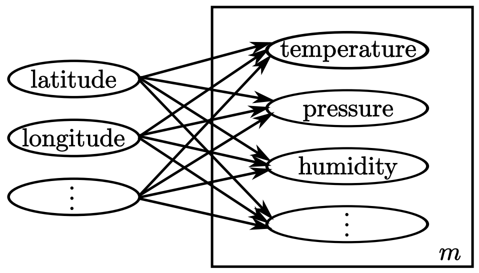


Fig. 2. An example of the uncertain object model.

Furthermore, as the conditional distribution is learned from the existing instances, more objects with similar properties give us more accurate results. Hence, the density of the conditioning attribute space provides us a confidence of the accuracy of the learned conditional distribution and the detected outliers.

Technically, we use Bayesian inference to learn the conditional distribution. In many applications, we have very little information about the true probability density function of the dependent attribute distribution or even as to its form. The probability density functions are often very complex. Therefore, we do not make any assumption on the conditioning attributes of objects or the population of instances of objects. Instead, we propose to employ kernel density estimation to estimate a probability distribution. To speed up the computation, we further develop an approximation algorithm that approximates the probability density of a target instance using instances within its ε-neighborhood, and a filtering algorithm that tries to filter instances without computing their exact normality.

In the rest of the paper, we first formally define our problem in Section II. After that, we develop the basic algorithm, the approximation algorithm, and the filtering algorithm in Sections III, IV, and V, respectively. Section VI presents an extensive empirical study. We review the related work in Section VII and conclude the paper in Section VIII.

II. Problem Definition

We assume a *space of conditioning attributes* (*C-space* for short) of dimensionality d_C and a *space of dependent attributes* (*D-Space* for short) of dimensionality d_D. The two spaces are exclusive.

An *uncertain object* (*object* for short) R_i is a tuple $(l_i, \{r_{ij} | 1 \leq j \leq m_i\})$ where l_i is a point in the C-space describing the inherent properties of R_i, and $\{r_{ij}|1 \leq j \leq m_i\}$ is a set of m_i *instances* in the D-space. It is possible that multiple objects coincide in the C-space (i.e., have the same property values). We also overload R_i by writing $R_i = \{r_{ij}|1 \leq j \leq m_i\}$.

Often, we use a variable R for objects whose conditioning attribute values form a random variable l in the C-space. We use a variable r for instances of R in the D-space.

By assuming that the instances of an object depend on the its conditioning attributes in the C-space, we measure the *normality of an instance* $r_{ij} \in R_i$ by the conditional probability of r_{ij} given l_i of R_i, that is,

$$nor(r_{ij}) = P(r = r_{ij}|R = R_i, l = l_i). \quad (1)$$

For the sake of simplicity, Equation (1) is also written as

$$nor(r_{ij}) = P(r_{ij}|R_i, l_i).$$

The normality defined as such measures the likelihood of an instance that is consistent with the uncertain objects and their instances observed. We note that the normality is the opposite of the outlierness commonly used in the literature. We use this form in order to preserve the probability meaning.

Then, the *normality of an object* R_i can be measured by the geometric mean of the normality of all instances of R_i,

$$nor(R_i) = \Big(\prod_{j=1}^{m_i} P(r_{ij}|R_i, l_i)\Big)^{\frac{1}{m_i}}. \quad (2)$$

Here, we use geometric mean because $\prod_{j=1}^{m_i} P(r_{ij}|R_i, l_i)$ measures the likelihood that R_i is consistent with the uncertain objects and their instances observed, and the m_i-th root normalizes the difference in number of instances among various objects. Please note that our outlier detection framework does not rely on the specific form of $nor(R_i)$. Generally, it can be applied to normality defined in other forms.

By the above definition, the smaller the normality of an instance or an object, the larger its outlierness and the more likely it is an outlier.

As a byproduct, the *confidence* of detecting an outlier object R_i and outlier instances of R_i is proportional to the probability density of its conditioning attribute l_i, that is,

$$conf(R_i) = P(l_i). \quad (3)$$

In this paper, we tackle the problem of finding outlier instances and outlier objects given a user specified normality threshold $\delta > 0$.

Problem Definition: Given a normality threshold $\delta > 0$ and a set of n uncertain objects $\mathbb{R} = \{R_i|1 \leq i \leq n\}$, where object $R_i = (l_i, \{r_{ij}|1 \leq j \leq m_i\})$, the problem of **outlier detection on uncertain data** is to find all instances r_{ij} such that $nor(r_{ij}) < \delta$ and all objects R_i such that $nor(R_i) < \delta$.

Please note that, to keep our discussion simple, we use the same threshold for both objects and instances. In general, different thresholds can be used to find outliers at the object level and the instance level, respectively.

Table I summaries the frequently used symbols.

Symbol	Description
d_C	the dimensionality of the C-space
d_D	the dimensionality of the D-space
n	the number of objects
R_i	the (distribution/sample of) the i-th object
m_i	the number of instances in the sample of R_i
r_{ij}	the j-th instance of the i-th object R_i
$r_{ij}.k$	the value of the k-th dependent attribute of r_{ij}
h_{ik}	the kernel bandwidth of the k-th dependent attribute of R_i
$\mathcal{N}_d(x|s, \Sigma)$	a d-dimensional Gaussian function with mean s and covariance matrix Σ

TABLE I
THE SUMMARY OF SYMBOLS

III. The Basic Algorithm

This section develops a basic algorithm to calculate the normality for instances and objects. Since we estimate the probability density of distributions, we first briefly review kernel density estimation in Section III-A. Then, Section III-B presents a Bayesian inference approach.

A. Kernel Density Estimation

To estimate the probability density of a distribution given its sample, we make use of kernel density estimation [4], [5] with Gaussian kernels. Given a sample S of a distribution f, the *kernel density estimate* approximates the probability density function f by the sum of $|S|$ Gaussian kernel functions, each of which is centered at a sample point $s \in S$ with variance h^2, where h is called the bandwidth and is used to control the level of smoothing. A popular choice of the bandwidth is the Sliverman approximation rule [4] for which $h = 1.06\sigma|S|^{-\frac{1}{5}}$, σ being the standard deviation of the sample. For the 1-dimensional case, the density estimate is

$$\hat{f}(x) = \frac{1}{|S|}\sum_{s \in S} \mathcal{N}(x|s, h^2),$$

where $\mathcal{N}(x|s, h^2)$ is the density of x in a Gaussian function with mean s and variance h^2. For the d-dimensional case ($d \geq 2$), assuming dimensions are independent to each other, the kernel function is the product of d Gaussian functions, each with its own bandwidth h_k ($1 \leq k \leq d$), the density estimate is

$$\hat{f}(x) = \frac{1}{|S|}\sum_{s \in S} \mathcal{N}_d(x|s, \Sigma),$$

where $\mathcal{N}_d(x|s, \Sigma)$ is a d-dimensional Gaussian function with mean s and covariance matrix $\Sigma = \begin{pmatrix} h_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & h_d^2 \end{pmatrix}$.

Algorithm 1 The basic algorithm.

Input: a set of objects $\mathcal{R}$; a normality threshold δ;
Output: the set of outlier instances OI and the set of outlier objects OO;
Description:

```
1: $OI = \emptyset$; $OO = \emptyset$;
2: for all $R_i \in \mathcal{R}$ do
3:     for all $r_{ij} \in R_i$ do
4:         calculate $nor(r_{ij})$ by Equation (8);
5:         if $nor(r_{ij}) < \delta$ then $OI = OI \cup \{r_{ij}\}$;
6:     end for
7:     calculate $nor(R_i)$ by Equation (2);
8:     if $nor(R_i) < \delta$ then $OO = OO \cup \{R_i\}$;
9: end for
10: return $OI$, $OO$;
```

B. Bayesian Inference

The core task in our outlier detection is to infer $P(r_{ij}|R_i, l_i)$. By Bayes' theorem, we have

$$P(r_{ij}|R_i, l_i) = \frac{P(r_{ij}|R_i)P(l_i|r_{ij}, R_i)}{P(l_i|R_i)} = \frac{P(r_{ij}|R_i)P(l_i|r_{ij})}{P(l_i)}, \tag{4}$$

since the C-space variable l is not dependent on the object variable R.

The priors $P(r_{ij}|R_i)$ and $P(l_i)$ can be estimated directly from the data obtained by kernel density estimation as

$$P(r_{ij}|R_i) = \frac{1}{m_i}\sum_{j'=1}^{m_i} \mathcal{N}_{d_D}(r_{ij}|r_{ij'}, \Sigma_{R_i}), \tag{5}$$

$$P(l_i) = \frac{1}{n}\sum_{i'=1}^{n} \mathcal{N}_{d_C}(l_i|l_{i'}, \Sigma_C), \tag{6}$$

where Σ_{R_i} is the covariance matrix of the instances of R_i and Σ_C is the covariance matrix of the conditioning attributes of objects.

To calculate the likelihood $P(l_i|r_{ij})$, we can first calculate the probability of r_{ij} given any object $R_{i'}$, i.e., $P(r_{ij}|R_{i'})$. Then the probability of l_i given the observation of r_{ij} is calculated using kernel density estimation where every point $l_{i'}$ is weighted by $P(r_{ij}|R_{i'})$, that is,

$$P(l_i|r_{ij}) = \frac{\sum_{i'=1}^{n}\left(P(r_{ij}|R_{i'})\mathcal{N}_{d_C}(l_i|l_{i'}, \Sigma_C)\right)}{\sum_{i'=1}^{n} P(r_{ij}|R_{i'})}. \tag{7}$$

Plugging Equations (5), (6), and (7) into Equation (4), we can evaluate the normality $nor(r_{ij})$ of instance r_{ij} as

$$nor(r_{ij}) = \frac{P(r_{ij}|R_i)\sum_{i'=1}^{n}\left(P(r_{ij}|R_{i'})\mathcal{N}_{d_C}(l_i|l_{i'}, \Sigma_C)\right)}{\sum_{i'=1}^{n} P(r_{ij}|R_{i'})P(l_i)} \tag{8}$$

C. Implementation

Algorithm 1 shows the pseudo code of the basic algorithm. It exhaustively calculates the normality of all instances of all objects according to Equation (8). Then, the normality of objects is evaluated using Equation (2). Checking with the normality threshold δ, the algorithm can identify outlier instances and outlier objects.

Essentially, we need to estimate the probability density of each point l_i in the C-space, and the probability density of each instance r_{ij} in the distributions of all n objects. In total, the number of Gaussian functions evaluated in the kernel density estimation is $O(n^2m^2)$, where $m = \frac{\sum_{i=1}^{n} m_i}{n}$ is the average number of instances of all objects. To reduce the computational cost, we will develop an approximation algorithm and a filtering algorithm in Sections IV and V, respectively.

IV. The Approximation Algorithm

To evaluate the density of a target point in a distribution represented by a sample of m points, the kernel density estimate requires to sum up the density contributions of all m points. However, because a Gaussian function decays exponentially with respect to the distance to its mean, the density contribution is small if a sample point is far away from the target point. By neglecting the effect of distant points to the target point, we save the computation with small cost in accuracy.

A. Approximation

To be concrete, we estimate the probability density using only points within an ε-range. For each instance r_{ij} ($i \in [1, n], j \in [1, m_i]$), we define the *$\varepsilon$-neighborhood instances* as

$$N(r_{ij}) = \{r \in R_{i'}, R_{i'} \in \mathbb{R} | \max_k\{|r.k - r_{ij}.k|\} \le \varepsilon\},$$

where $max_k\{|r.k - r_{ij}.k|\}$ is the maximum projected distance on any dimension between r and r_{ij}. Essentially, $N(r_{ij})$ consists of the instances in a hyper-cube centered at r_{ij} with edge length 2ε. $R_{i'}$ is called an *ε-neighborhood object* of r_{ij} if at least one instance of $R_{i'}$ is in the ε-neighborhood of r_{ij}. We also denote the set of ε-neighbor objects of r_{ij} as

$$NS(r_{ij}) = \{R_{i'} | \exists r \in R_{i'}, r \in N(r_{ij})\}.$$

Example 4: Figure 3 illustrates an example of ε-neighborhood instances and ε-neighborhood objects. The dashed square is a hyper-cube centered at r_{ij} with edge length 2ε. r is an ε-neighborhood instance. R_i and $R_{i'}$ are ε-neighborhood objects. ■

$P(r_{ij}|R_i)$ and $P(l_i|r_{ij})$ can be approximated using the ε-neighborhood instances in $N(r_{ij})$ and the ε-neighborhood objects in $NS(r_{ij})$ as follows.

$$P(r_{ij}|R_i) = \frac{1}{m_i}\sum_{\forall r_{ij'} \in N(r_{ij})} \mathcal{N}_{d_D}(r_{ij}|r_{ij'}, \Sigma_{R_i}), \tag{9}$$

$$P(l_i|r_{ij}) = \frac{\sum_{\forall R_{i'} \in NS(r_{ij})}\left(P(r_{ij}|R_{i'})\mathcal{N}_{d_C}(l_i|l_{i'}, \Sigma_C)\right)}{\sum_{\forall R_{i'} \in NS(r_{ij})} P(r_{ij}|R_{i'})}. \tag{10}$$

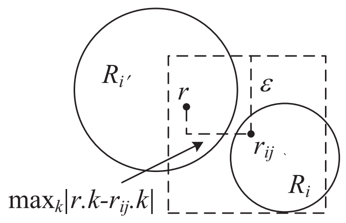


Fig. 3. An example of ε-neighborhood instance and ε-neighborhood object.

Algorithm 2 The approximation algorithm.

Input: a set of objects $\mathcal{R}$; a normality threshold δ;

Output: the set of outlier instances OI and the set of outlier objects OO;

Description:

```
1: $OI = \emptyset$; $OO = \emptyset$;
2: index the instances of all objects in $\mathcal{R}$ into an R*-tree $T$;
3: self-join $T$ to find the $\varepsilon$-neighborhood of every instance;
4: for all $R_i \in \mathcal{R}$ do
5:     for all $r_{ij} \in R_i$ do
6:         estimate $nor(r_{ij})$ using its $\varepsilon$-neighborhood;
7:         if $nor(r_{ij}) < \delta$ then $OI = OI \cup \{r_{ij}\}$;
8:     end for
9:     calculate $nor(R_i)$ by Equation (2);
10:    if $nor(R_i) < \delta$ then $OO = OO \cup \{R_i\}$;
11: end for
12: return $OI$, $OO$;
```

Similarly, we can also estimate the probability density $P(l_i)$ for each point l_i. Then, we can approximate the normality in Equation (8).

When we approximate the Gaussian function $\mathcal{N}_{d_D}(r_{ij}|r_{ij'}, \Sigma_{R_i})$, the approximation error happens if $\max_k\{|r_{ij}.k - r_{ij'}.k|\} > \varepsilon$. Therefore, the approximation error is at most $\frac{\exp(-\sum_{k=1}^{d_D} \frac{\varepsilon^2}{2h_{i,k}^2})}{(2\pi)^{\frac{d_D}{2}} \prod_{k=1}^{d_D} h_{ik}}$, where h_{ik} is the bandwidth of R_i on the k-th dimension. In practice, we can choose ε to be about the average standard deviation σ of the objects. Suppose h_{ik} is selected to be $1.06\sigma m_i^{-\frac{1}{5}}$. When we have $m_i = 100$ instances in a $d_D = 3$ dimensional D-space, $\exp(-\sum_{k=1}^{d_D} \frac{\varepsilon^2}{2h_{i,k}^2}) \approx \exp(-d_D m_i^{\frac{2}{5}}) \approx 10^{-8}$. The approximation error is very small.

Consider the approximation of $P(r_{ij}|R_i)$. The density contribution of r_{ij} itself is $\mathcal{N}(r_{ij}|r_{ij}, R_i) = \frac{1}{(2\pi)^{\frac{d_D}{2}} \prod_{k=1}^{d_D} h_{ik}}$, which is 10^8 times larger than the approximation error stated above. Hence, we expect our approximation of $P(r_{ij}|R_i)$ and Equation (8) to be accurate. This is verified by our experiments.

B. Implementation

To efficiently implement the approximation algorithm, we need to find the ε-neighborhood for every instance. R*-tree [6] is one of the practical data structures to solve this problem. A node of an R*-tree contains a set of entries. Each entry in a leaf node contains an instance. Each entry in a non-leaf node has the form $\langle child, \mathit{MBB}\rangle$ where $child$ refers to a child node, and *MBB* is the minimum bounding box of the child node.

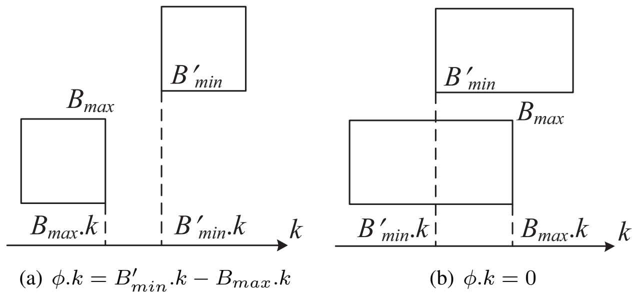


Fig. 4. An illustration of calculating $\phi.k$.

We index all instances of all objects into a single R*-tree. By self-joining this R*-tree, we can find the ε-neighborhood for every instance with a single pass of the tree.

In detail, we maintain two copies of the R*-tree and traverse them synchronously. We start from the root of both copies. In each iteration, a pair of nodes B and B' from the two copies, respectively, is processed. If the minimum distance between B and B' is no less than ε, then any instance in B will not be in the ε-neighborhood of any instances in B'. Thus, the pair of B and B' can be excluded for further consideration. Otherwise, if B and B' are not leaf nodes, for every child node b of B and every child node b' of B', the pair b and b' will be examined in the same manner. In the case where B and B' are leaf nodes, then every pair of instances from the two nodes is checked to see if they are ε-neighbors.

The minimum distance between two nodes B and B' is the minimum projected distance on any dimension between their minimum bounding boxes, which is given by *mindist*$(B, B') = \min_k\{\phi.k\}$, where

$$\phi.k = \begin{cases} B'_{min}.k - B_{max}.k & \text{if } B_{max}.k \le B'_{min}.k; \\ B_{min}.k - B'_{max}.k & \text{if } B'_{max}.k \le B_{min}.k; \\ 0 & \text{otherwise.} \end{cases}$$

Here, B_{max} and B_{min} are the maximum and minimum corners of the minimum bounding box of B, respectively. Figure 4 illustrates the formula of $\phi.k$.

Algorithm 2 shows the pseudo code of the approximation algorithm.

Since the approximation algorithm calculates the normality of every instance using only the ε-neighborhood, it evaluates $O(mnt)$ Gaussian functions in total, assuming on average there are t instances in the ε-neighborhood of each instance. The cost to build the R*-tree is approximately $O(mn\log(mn))$ and the cost to retrieve the ε-neighborhood of every instance is $O(mn\log(mn))$. Thus, the total cost of the approximation algorithm is $O\big(mn(t + 2\log(mn))\big)$.

The basic algorithm and the approximation algorithm both calculate the normality of every instance individually. In the next Section, we develop a filtering algorithm such that we can determine whether an instance is an outlier without actually computing its normality value.

V. The Filtering Algorithm

Heuristically, in an object, instances similar to a normal instance are likely to be normal as well. In a smooth distribution, similar instances should have similar probability densities. We use Gaussian distribution, which is smooth, in our approximation method. Based on this property, once we find a normal instance, we have a chance to determine that some similar instances are also normal without computing their exact normality values.

A. Bounding the Normality

The normality values of two instances have the following relationship.

Lemma 1 (Bounding normality): For two instances r_{ij} and $r_{ij'}$ of object R_i located at l_i, if

$$\max_k\{|r_{ij}.k - r_{ij'}.k|\} \le \alpha, (\alpha > 0),$$

then

$$\frac{nor(r_{ij'})}{nor(r_{ij})} \ge \exp\big(-\frac{3}{2}(\alpha^2 + 2\varepsilon\alpha)\beta\big),$$

where

$$\beta = \max_{i'}\{\sum_{k=1}^{d_D} \frac{1}{h_{i'k}^2} | R_{i'} \in NS(r_{ij}) \cup NS(r_{ij'})\}.$$

Proof: We first prove, for any object $R_{i'}$,

$$\exp\big(-(\frac{\alpha^2}{2}+\varepsilon\alpha)\sum_{k=1}^{d_D}\frac{1}{h_{i'k}^2}\big) \le \frac{P(r_{ij'}|R_{i'})}{P(r_{ij}|R_{i'})} \le \exp\big((\frac{\alpha^2}{2}+\varepsilon\alpha)\sum_{k=1}^{d_D}\frac{1}{h_{i'k}^2}\big). \quad (11)$$

For any instance $r \in R_{i'}$, following the approximation strategy, we have $\max_k\{|r_{ij}.k - r.k|\} \le \varepsilon$. Then, for the k-th dependent attribute ($k \in [1, d_D]$), we have

$$\begin{aligned}(r_{ij'}.k - r.k)^2 &= \big((r_{ij'}.k - r_{ij}.k) + (r_{ij}.k - r.k)\big)^2 \\ &\le (r_{ij}.k - r.k)^2 + 2\varepsilon\alpha + \alpha^2.\end{aligned}$$

Plugging the above inequality into Gaussian function,

$$\mathcal{N}(r_{ij'}|r, \Sigma_{R_{i'}}) \ge \mathcal{N}(r_{ij}|r, \Sigma_{R_{i'}}) \exp\big(-(\frac{\alpha^2}{2}+\varepsilon\alpha)\sum_{k=1}^{d_D}\frac{1}{h_{i'k}^2}\big),$$

where $\Sigma_{R_{i'}}$ is the covariance matrix of $R_{i'}$ and $h_{i'k}$ is the bandwidth of the k-th dimension of $R_{i'}$. So,

$$P(r_{ij'}|R_{i'}) \ge P(r_{ij}|R_{i'}) \exp\big(-(\frac{\alpha^2}{2}+\varepsilon\alpha)\sum_{k=1}^{d_D}\frac{1}{h_{i'k}^2}\big).$$

On the other hand,

$$\begin{aligned}&\big((r_{ij'}.k - r.k) - (r_{ij'}.k - r_{ij}.k)\big)^2 = (r_{ij}.k - r.k)^2, \\ &(r_{ij'}.k - r.k)^2 \ge (r_{ij}.k - r.k)^2 - 2\varepsilon\alpha - \alpha^2.\end{aligned}$$

Algorithm 3 The filtering algorithm.

Input: a set of objects $\mathcal{R}$; a normality threshold δ;
Output: the set of outlier instances OI and the set of outlier objects OO;
Description:

```
1: $OI = \emptyset$; $OO = \emptyset$;
2: for all $R_i \in \mathbb{R}$ do
3:     while $R_i \neq \emptyset$ do
4:         find a stationary point $r_{ij}$ of $R_i$ using mean shift;
5:         compute $nor(r_{ij})$ by Algorithm 2;
6:         if $nor(r_{ij}) > \delta$ then
7:             compute the filtering region $F(r_{ij})$ (Theorem 1);
8:             $R_i = R_i \setminus (\{r_{ij}\} \cup \{r_{ij'} \in F(r_{ij})\})$;
9:         else if $nor(r_{ij}) < \delta$ then
10:            $OI = OI \cup \{r_{ij}\}$; $R_i = R_i \setminus \{r_{ij}\}$;
11:        end if
12:    end while
13:    determine whether $R_i$ is an outlier;
14: end for
15: return $OI$ and $OO$;
```

Similarly, we have

$$P(r_{ij'}|R_{i'}) \le P(r_{ij}|R_{i'}) \exp\big((\frac{\alpha^2}{2}+\varepsilon\alpha)\sum_{k=1}^{d_D}\frac{1}{h_{i'k}^2}\big).$$

Thus, Inequality (11) holds. Applying it to the normality formula (Equation (8)) and let

$$\beta = \max_{i'}\{\sum_{k=1}^{d_D} \frac{1}{h_{i'k}^2} | R_{i'} \in NS(r_{ij}) \cup NS(r_{ij'})\},$$

we have

$$\frac{P(r_{ij'}|R_i, l_i)}{P(r_{ij}|R_i, l_i)} \ge \frac{\exp\big(-2(\frac{\alpha^2}{2}+\varepsilon\alpha)\beta\big)}{\exp\big((\frac{\alpha^2}{2}+\varepsilon\alpha)\beta\big)}.$$

Thus, the lemma is proved. ■

Using Lemma 1, once we find a normal instance r_{ij}, we can determine its filtering region within which any instance is also normal.

Theorem 1 (Filtering region): For two instances r_{ij} and $r_{ij'}$ of object R_i located at l_i such that $\max_k\{|r_{ij}.k - r_{ij}.k|\} \le \alpha$, if $nor(r_{ij}) > \delta$ and

$$\alpha < \sqrt{\frac{2}{3\beta}\ln\frac{nor(r_{ij})}{\delta} + \varepsilon^2} - \varepsilon,$$

then $nor(r_{ij'}) > \delta$.

Proof: By Lemma 1, if $\exp\big(-\frac{3}{2}(\alpha^2 + 2\varepsilon\alpha)\beta\big) > \frac{\delta}{nor(r_{ij})}$, then $nor(r_{ij'}) > \delta$. Solving this inequality, we have the theorem. ■

B. Implementation

In order to maximize the filtering power of Theorem 1, a heuristic approach is to find an instance in the dense area of an object so that the filtering region is large and dense, and contains a good number of instances. We borrow the idea used in mean shift [7], [8] to achieve this goal.

Given a set of data points in a multidimensional space, mean shift is a simple iterative procedure that shifts a query point to the mean of data points in its neighborhood. Eventually, the procedure converges at a stationary point which has the maximum local density.

In our problem, for each object R_i, we identify a *stationary instance* r_{ij} which is in a dense region of the instance distribution of R_i. To identify a stationary instance, we first randomly pick a query point $r^{(0)}$ in the D-space. In the z-th iteration $(z \geq 1)$, the query point $r^{(z-1)}$ is shifted to its local center $r^{(z)}$ according to the following formula, for $k \in [1, d_D]$,

$$r^{(z)}.k = \frac{\sum_{j=1}^{m_i} \mathcal{N}(r_{ij}.k | r^{(z-1)}.k, h_{ik}^2) r_{ij}.k}{\sum_{j=1}^{m_i} \mathcal{N}(r_{ij}.k | r^{(z-1)}.k, h_{ik}^2)}.$$

Like the approximation algorithm, we can also evaluate $r^{(z)}$ using only its ε-neighborhood. Limited by space, we omit the details here. Once the distance between $r^{(z)}$ and $r^{(z-1)}$ is smaller than a parameter γ, we stop the iteration and find the instance r_{ij} closest to $r^{(z)}$. r_{ij} is the stationary instance in the distribution of R_i.

Algorithm 3 presents the pseudocode of the filtering algorithm. For each object R_i, we first find a stationary instance r_{ij} and compute the normality of r_{ij} (lines 4, 5). If we find that r_{ij} is a normal instance, then its filtering region $F(r_{ij})$ is computed according to Theorem 1 (lines 6, 7). The instances in $F(r_{ij})$ are normal instances and thus can be pruned for further computation (line 9). Otherwise, r_{ij} is an outlier instance (line 10).

All outlier instances and some normal instances have their normality computed in the algorithm. For those filtered normal instances, their normality are greater than the threshold δ. Therefore, by Equation (2), we have a lower bound of the normality of an object R_i. If this lower bound passes δ, then R_i is a normal object. Otherwise, the exact value of $nor(R_i)$ is computed by the approximation algorithm. As the amount of outlier instances is small, we expect that the lower bound works well in practice, as verified by our experiment results.

VI. Empirical Study

We conducted extensive experiments on synthetic data sets and a real data set to evaluate the effectiveness and efficiency of the three algorithms developed in this paper, namely, the basic algorithm (denoted by BS), the approximation algorithm (denoted by AP), and the filtering algorithm (denoted by FT). For comparison, we implemented a density-based outlier detection algorithm (denoted by DS) which simply measures the normality of each instance $r_{ij} \in R_i$ by its probability density $P(r_{ij}|R_i)$ in R_i.

All methods were implemented in C++ and compiled by GCC 4.3.3. All experiments were conducted on a computer with an Intel Core 2 Duo P8700 2.53GHz CPU and 4GB main memory running Ubuntu 9.10. All programs run in main memory. The I/O cost is not reported.

A. Results on Synthetic Data

Since mixture Gaussian models can approximate a large class of probability density functions, we use mixture Gaussian models to generate the conditioning attributes of objects and the normal instances of an object. The outlier instances of an object are drawn from a uniform distribution which is totally different from the mixture Gaussian model.

The dimensionality d_C of the C-space varies from 2 to 6, and the dimensionality d_D of the D-space varies from 2 to 10. To generate a synthetic data set, we first generate a set of n points in a d_C-dimensional space following a mixture Gaussian model consisting of 5 equally weighted processes having random means and variances. The i-th point is associated with an object R_i having m instances. We randomly select 10 objects as outlier objects. The others are normal objects. For each normal object, 99% of the instances are normal, and the rest 1% are outlier instances. The amount of outlier instances of an outlier object is randomly picked between 50% and 90%.

The normal instances of each object are drawn from a d_D-dimensional mixture Gaussian model which is generated in a specific way so that two objects have similar instance distributions if their conditioning attributes are similar. The mixture Gaussian model of each object consists of 3 equally weighted process having the same variance 0.1. The mean of each Gaussian process is generated by a mapping f from the C-space to the D-space such that this mapping preserves the locality between the two spaces. That is to say, given two points l, l' in the C-space, their corresponding points r, r' in the D-space satisfy $||r - r'|| = ||l - l'||$, where $||.||$ is the Euclidean distance.

To construct a locality preserving mapping from C-space to D-space, we require $d_C \leq d_D$ to guarantee there exists such a mapping. The mapping is constructed as

$$f((l.1, \cdots, l.d_C)) = (r.1, \cdots, r.d_D),$$

where

$$r.i = \sum_{j=1}^{d_C} w_j.i \times l.j \text{ for } i \in [1, d_D],$$

and $w_j.i$'s picked such that

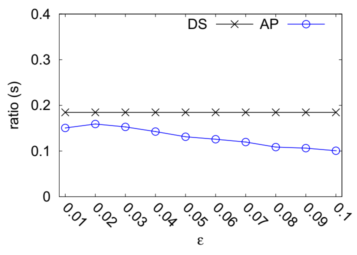
$$\sum_{i=1}^{d_D} w_j.i^2 = 1 \text{ for } \forall j \in [1, d_C]$$

and

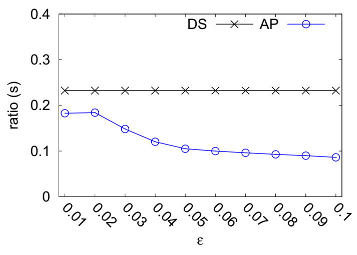
$$\sum_{i=1}^{d_D} w_j.i \times w_k.i = 0 \text{ for } \forall j, k \in [1, d_C], j \neq k.$$

This can be done by the following iterative method with respect to j from 1 to d_C. For every j, we first randomly pick $w_j.i$ for $i \in [j, d_D]$. Then, we solve a linear system of equations

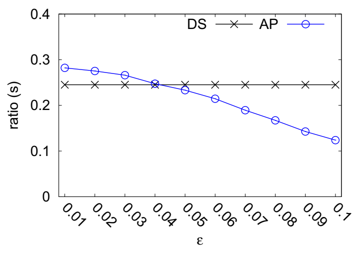
$$\sum_{i=1}^{d_D} w_j.i \times w_k.i = 0 \text{ for } k \in [1, j-1]$$



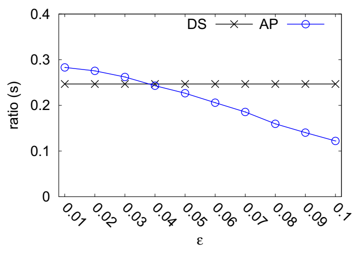
(a) $d_D = 3, m = 100$



(b) $d_D = 3, m = 1000$

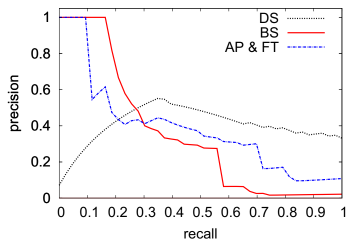


(c) $d_D = 10, m = 100$

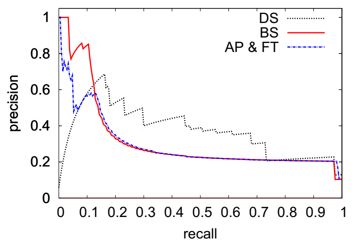


(d) $d_D = 10, m = 1000$

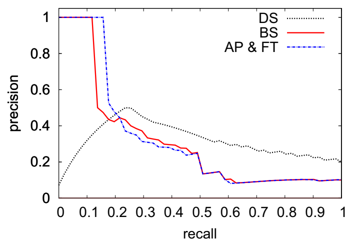
Fig. 5. Accuracy on small data sets ($n = 100, d_C = 2, \delta = 0.1$).



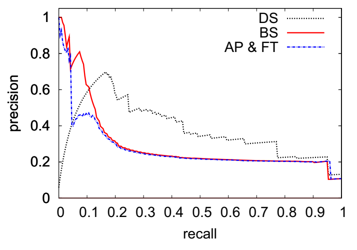
(a) $d_D = 3, m = 100$



(b) $d_D = 3, m = 1000$

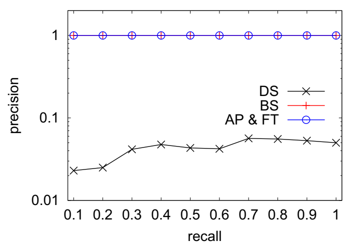


(c) $d_D = 10, m = 100$

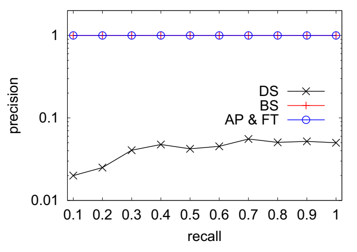


(d) $d_D = 10, m = 1000$

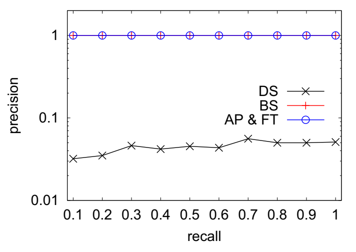
Fig. 6. Precision and recall on detecting outlier instances on small data sets ($n = 100, d_C = 2, \delta = 0.1$).



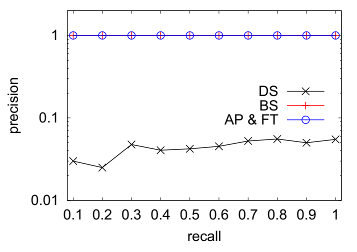
(a) $d_D = 3, m = 100$



(b) $d_D = 3, m = 1000$



(c) $d_D = 10, m = 100$



(d) $d_D = 10, m = 1000$

Fig. 7. Precision and recall on detecting outlier objects on small data sets ($n = 100, d_C = 2, \delta = 0.1$).

to determine $w_j.i$ for $i \in [1, j-1]$. After that, we normalize $w_j.i$ for $i \in [1, d_D]$ to make $\sum_{i=1}^{d_D} w_j.i^2 = 1$.

For each object, we also generate outlier instances following uniform distribution. We note that it is very likely that some outlier instances might be generated in an area being populated by normal instances. Thus, even a good outlier detection mechanism cannot find such outlier instances.

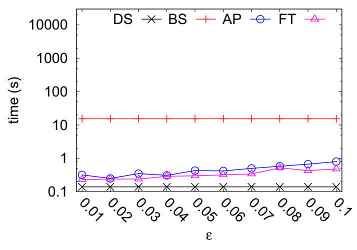
1) Effectiveness: We first conduct experiments to evaluate the approximation error of the normality by AP developed in Section IV. The error ratio of the normality of an instance is computed as $|appr-exact|/exact$, where *appr* and *exact* are the approximated and exact normality of an instance, respectively. *exact* is computed by BS. We plot the average error ratio of all instances in Figure 5. DS simply uses the density of each instance as its normality. The ratio is plotted with ε varying from 0.01 to 0.1. AP gets more accurate approximation as we increase ε. AP always outperforms DS on 3-dimensional data sets, and is better than DS on 10-dimensional data sets when $\varepsilon > 0.04$. In the rest of our experiments, be default, $\varepsilon = 0.05$.

Figure 6 shows the precision-recall graphs on detecting outlier instances of the 4 algorithms on 4 data sets, each of which has $n = 100$ objects. The D-space dimensionality d_D and the number m of the instances of each object are shown in the subtitle of each figure. AP and FT output the same result and they are represented by a single curve. Generally, when the recall level is low, our algorithms BS, AP, and FT achieve high precision, while the precision of DS is low. The precision of our algorithms decreases as the recall increases, because outliers may be sampled into the dense area of the normal instances, which cannot be found.

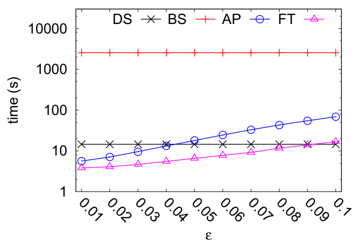
Figure 7 shows the precision-recall graphs on detecting outlier objects. The graphs are drawn in logarithmic scale. Clearly, BS, AP, and FT can rank all outlier objects into top positions perfectly. However, DS cannot correctly identify outlier objects.

In summary, our algorithms, BS, AP, and FT, can detect the outlier instances not only in normal objects but also in outlier objects. But DS cannot detect outlier instances in outlier objects thus it fails to detect outlier objects.

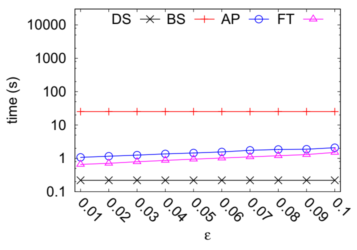
2) Efficiency: Figure 8 shows the running time in logarithmic scale of the 4 algorithms on small data sets with only 100 objects . BS spends more than $2,000$ seconds to process a data set of 100 objects, each object having $1,000$ instances. Thus, BS cannot handle large data sets. AP and FT are much



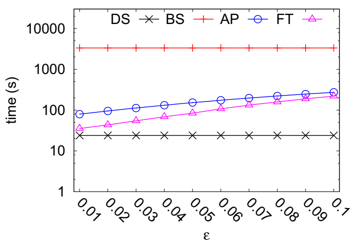
(a) $d_D = 3, m = 100$



(b) $d_D = 3, m = 1000$

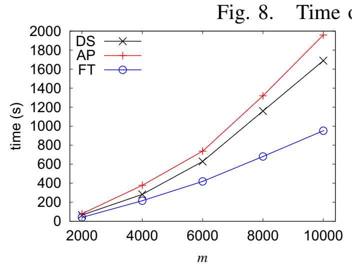


(c) $d_D = 10, m = 100$

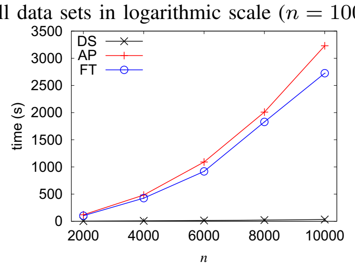


(d) $d_D = 10, m = 1000$

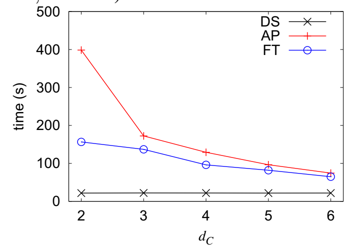
Fig. 8. Time on small data sets in logarithmic scale ($n = 100, d_C = 2, \delta = 0.1$).



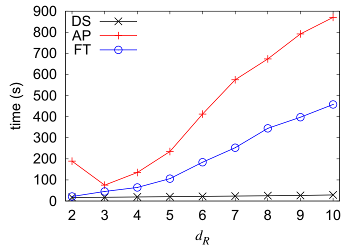
(a) number of instances with $n = 100$



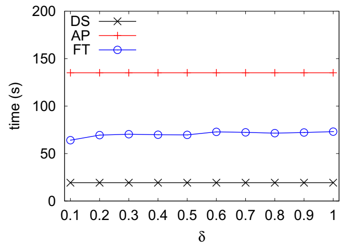
(b) number of objects with $n = m = 500$



(c) d_C with $n = m = 500$



(d) d_D with $m = 100$



(e) threshold with $n = m = 500$

Fig. 9. Time on large data sets (by default $d_C = 2, d_D = 4, \delta = 0.1$).

faster than BS thanks to the approximation techniques. FT outperforms AP because of the filtering. The running time increases when ε increases.

Figure 9 shows the scalability of AP and FT with respect to 5 different factors, the number m of instances per object, the number n of objects, the dimensionality d_C of the C-space, the dimensionality d_D of the D-space, and the threshold δ. FT outperforms AP in all cases, especially when there are a large number of instances and d_D is large. The running time of AP and FT increases super-linearly with respect to m and n, and sub-linearly with respect to d_D. However, the running time decreases as d_C increases. This is because the overlap among objects in both spaces decreases according to the mapping used to generate the data sets, so the cost of computing the normality of each target instance is reduced since there are fewer instances in the ε-neighborhood of the target instance. The runtime also increases slightly as the threshold δ increases.

Figure 10 investigates the filtering power of FT on the corresponding data sets. This explains the reason why FT runs faster then AP. In most cases, FT can filter out 30% instances. And the filtering power is as high as 60% when each object has a large number of instances.

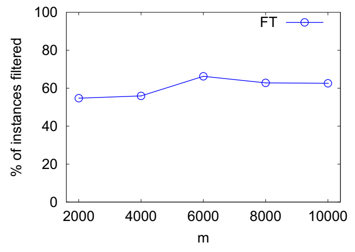
In conclusion, FT runs faster than AT in all cases thanks to the filtering technique. FT works especially well when objects have a large number of instances.

B. Results on a Real Data Set

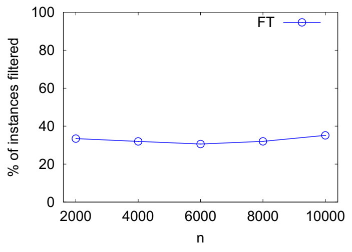
We obtained a weather data set from the National Center for Atmospheric Research data archive (http://dss.ucar.edu/datasets/ds512.0/). The data set consists of $7,437$ stations around the world. We use the longitude and latitude of each station as its conditioning attributes. Each station contains 366 daily records in 2008, which are treated as instances. Each record has 3 attributes, average temperature, precipitation, and average humidity. The data set is normalized so that the range of each attribute is roughly the same.

As a case study, Figure 11(a) shows the temperature distributions of 4 nearby cities, Vancouver, Calgary, Edmonton, and Prince George. We observe, except for Vancouver, the other 3 cities have similar temperature distributions. The temperature distribution of Vancouver is very different due to its different terrain. Figure 11(b) shows the distributions of instance normality of the 4 cities. Clearly, instances of Vancouver have much lower normality scores, thus are more likely to be outliers. This case clearly shows how the instance normality can help to define the object normality.

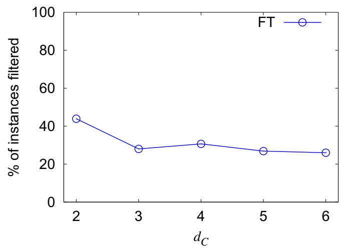
Figure 12 shows the running time and the filtering power of FT. The results on this real data set are consistent with those



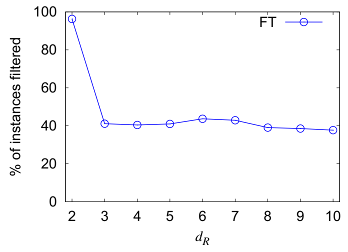
(a) number of instances with $n = 100$



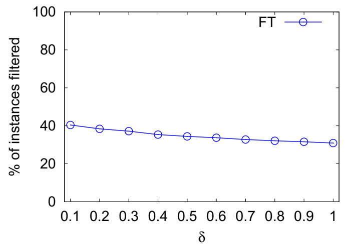
(b) number of objects with $m = 100$



(c) d_C with $n = m = 500$

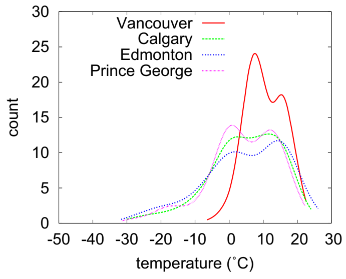


(d) d_D with $n = m = 500$

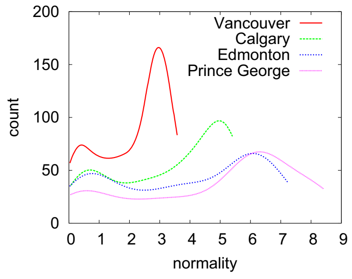


(e) threshold with $n = m = 500$

Fig. 10. Filtering power of the filtering algorithm (by default $d_C = 2, d_D = 4, \delta = 0.1$).

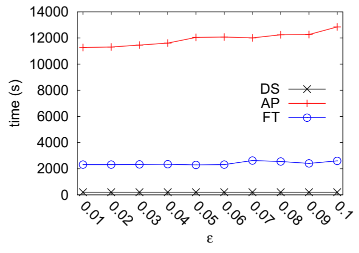


(a) Temperature

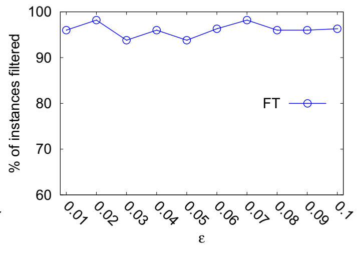


(b) Normality

Fig. 11. A case study of 4 cities.



(a) Running time



(b) Filtering power

Fig. 12. Results on the weather data set.

on the synthetic data sets. The runtime of FT is much faster than that of AP, because the objects in the weather data set are heavily overlapping. On the one hand, the overlap increases the cost of computing the normality of each instance. Thus, AP runs much slower than it does on the synthetic data sets. On the other hand, the overlap enables AP to filter out a large amount of instances as shown in Figure 12(b).

VII. Related Work

Uncertain data has been attracting a lot of attention in research and development. In the data mining field, various mining techniques, such as clustering [9], [10], [11], [12], [13], [14] and frequent pattern mining techniques [15], [16], [17], are migrated from certain data to deal with uncertain data. In this section, we first review the models of uncertain data in Section VII-A, and then compare our work with the previous work on outlier detection on certain data and uncertain data, in Section VII-B and Section VII-C, respectively.

A. Uncertain Data Models

In the literature, uncertain data can be represented in two ways, the *probabilistic database* model and the *uncertain object* model, following the possible world semantics [18], [19], [20], [21], [22].

A probabilistic database [23], [21], [24] is a finite set of probabilistic tables. A probabilistic table T contains a set of (uncertain) tuples, where each tuple $t \in T$ is associated with a membership probability $P(t) > 0$, namely, t takes probability $P(t)$ to appear in a possible world. The relationships among tuples are modeled by generation rules which specify a set of exclusive tuples. Among all tuples involved in a generation rule, only one appears in a possible world.

An uncertain object [1], [2], [3] is conceptually described by a probability density function f in a data space D. Practically, the probability density function is often unavailable explicitly. Instead, a set of samples are drawn or collected in the hope of approximating the probability density function. Correspondingly, an uncertain object is modeled as a set of points as its instances, denoted by $U = \{u_1, \ldots, u_m\}$, and associated with a probability mass function over the set of instances. An object has only one instance in a possible world.

The probabilistic database model and the uncertain object model are equivalent in the discrete case and can be converted to each other [25]. A set of uncertain objects can be represented as a probabilistic database as follows. For each instance u of an uncertain object U, a tuple is created and associated with a membership probability $f(u)$. Each uncertain object U corresponds to a generation rule such that only one of its instances can appear in a possible world. To convert a probabilistic database, an instance is created for each tuple and an object is created for a generation rule which contains all instances of the corresponding tuples in this generation rule.

In this paper, we present our model and techniques following the uncertain object model. We also extend the uncertain object model to allow an object to have some inherent properties in its conditioning space. As the two models are equivalent and can be converted to each other, our model and techniques work for general uncertain data.

B. Outlier Detection on Certain Data

Many techniques have been developed for outlier detection on certain data. Chandola et al. [26] gave a comprehensive survey. There are two major types of outliers, point outliers and contextual outliers. We briefly review them in this section.

1) Point Outliers: Point outliers consider an individual data point as outliers with respect to the rest of the data. This is the simplest type of outliers and is the major focus in the research community. Among various methods on detecting point outliers, the statistical distribution-based approach [27] is the most relevant one to our method.

Given a certain data set, the statistical distribution-based approach makes a hypothesis which assumes a distribution or a probabilistic model for the data set. Then, it identifies outliers with respect to the model using a hypothesis test. The test verifies whether an object (i.e., a point in the certain data case) is significantly different from the assumed distribution. If so, the object is found as an outlier.

Our method for outlier detection on uncertain data also uses distributions. An instance of an object is outlier if it is very different from the true instance distribution of the object. However, the fundamental difference is that in order to make an efficient hypothesis the traditional statistical distribution-based approach requires the knowledge of the object, that is, the type of the distribution that the instances follow in our problem. Our method learns the instance distribution of the object using other objects. Moreover, most statistical hypothesis tests are for a single attribute, they are not suitable for solving our problem in a multidimensional space.

Besides statistical distribution-based outlier detection methods, there are also other point outlier detection approaches developed for certain data, including distance-based outlier detection [28], [29], density-based outlier detection [30], and deviation-based detection [31]. Compared to our method, these approaches more or less assume the knowledge of the normal data points. So the outlier detection becomes a task to find outliers which are different from the normal data points according to various measurements. However, in our problem, the normal instances of an object cannot be determined by the object itself. To identify outlier instances, we first need to learn what normal instances are.

2) Contextual Outliers: Contextual outliers are also called conditional outliers [32]. A data point is a contextual outlier if it is an outlier in a specific context, but not otherwise. Similar to our model, a data point is associated with a set of *contextual attributes* and a set of *behavioral attributes*, which are analogous to our terminology conditioning attributes and dependent attributes. Contextual outliers have applications in spatial data and time series data, etc. Contextual attributes define the context, such as the longitude and latitude of a location in spatial data and timestamps in time series data. Behavioral attributes are used to determine outliers. For example, in a spatial data set of average precipitation of the world, the amount of the average precipitation is the behavioral attribute. A low precipitation is an outlier in a rainforest, but the same value may be normal in a desert. Similarly, in a time series data set, a low precipitation in June may be an outlier, but it may not be so in Winter.

One approach for contextual outliers is to apply a point outlier detection technique as follows [26]. First, the context is identified using the contextual attributes and a mapping from the context to the behavioral attributes is learned. Then, an outlier score is computed using the behavioral attributes within the context. The methods of identifying context vary from one problem to the other. Most of them came up with a function using specific domain knowledge (e.g., Euclidean distances in spatial data) to measure the outlierness.

Our framework adopts a similar strategy. However, the major difference is that the dependent/behavioral attributes of an uncertain object are a distribution instead of a single value. Techniques developed for certain data cannot solve our problem. Moreover, our Bayesian inference based methods do not use specific domain knowledge, thus are more general.

C. Outlier Detection on Uncertain Data

There are two studies in the literature on detecting outliers on uncertain data.

Aggarwal *et al.* [33] are the first to investigate the problem of outlier detection on uncertain data. In their work, an uncertain point (object) is represented by a probability density function. They define an uncertain point X to be a (δ, η)-outlier if the probability of X lying in a region in some subspace with density at least η is less than δ. Similar to our work, the density is estimated by Gaussian kernel density estimation. However, their work only focuses on detecting outlier objects without considering outlier instances.

Wang *et al.* [34] propose the distance-based outlier detection on an uncertain table consisting of a set of tuples, each of which is associated with an appearing probability. Their definition of outliers is based on the possible world semantics [18], [19], [20], [21], [22]. A tuple t is an (up, ud, λ)-outlier if the sum of the probabilities of the possible worlds, each of which consists of more than $(1 - up)\%$ of tuples within a distance ud to t, is less than λ. They only consider tuple level uncertainty, that is, a tuple either appears or not, but not a multi-representation uncertain object. Thus, they only consider detecting outliers at the object level.

To the best of our knowledge, we are the first to detect both outlier instances and outlier objects on uncertain data.

VIII. Conclusions

This paper studies the problem of outlier detection on uncertain data. An uncertain object is characterized by a set of conditioning attributes and associated with a set of instances which are described by a set of dependent attributes. In our model, dependent attributes are conditionally dependent on conditioning attributes. Objects with similar conditioning attribute values likely have similar dependent attribute values. Under this assumption, we learn the true values of the dependent attributes of each object, then detect outliers at both the object level and the instance level. To the best of our knowledge, this paper is the first to detect outlier instances as well as outlier objects.

Acknowledgement

This research is supported in part by an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplement Grant. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

References

[1] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. ACM, 2003, pp. 551–562.

[2] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proceedings of the 33rd International Conference on Very Large Data Bases*. ACM, 2007, pp. 15–26.

[3] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar, "Indexing multi-dimensional uncertain data with arbitrary probability density functions," in *Proceedings of the 31st International Conference on Very Large Data Bases*. ACM, 2005, pp. 922–933.

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

[5] D. W. Scott, *Multivariate Density Estimation: Theory, Practical, and Visualization*. Wiley, New York, 1992.

[6] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, 1990, pp. 322–331.

[7] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 8, pp. 32–40, 1975.

[8] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[9] W. K. Ngai, B. Kao, C. K. Chui, R. Cheng, M. Chau, and K. Y. Yip, "Efficient clustering of uncertain data," in *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2006, pp. 436–445.

[10] B. Kao, S. D. Lee, D. W. Cheung, W.-S. Ho, and K. F. Chan, "Clustering uncertain data using voronoi diagrams," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2008, pp. 333–342.

[11] S. D. Lee, B. Kao, and R. Cheng, "Reducing uk-means to k-means," in *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2007, pp. 483–488.

[12] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2005, pp. 672–677.

[13] ——, "Hierarchical density-based clustering of uncertain data," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 2005, pp. 689–692.

[14] P. B. Volk, F. Rosenthal, M. Hahmann, D. Habich, and W. Lehner, "Clustering uncertain data with possible worlds," in *Proceedings of the 25th International Conference on Data Engineering, ICDE*. IEEE, 2009, pp. 1625–1632.

[15] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 29–38.

[16] Q. Zhang, F. Li, and K. Yi, "Finding frequent items in probabilistic data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*, 2008, pp. 819–832.

[17] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Züfle, "Probabilistic frequent itemset mining in uncertain databases," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 119–128.

[18] S. Abiteboul, P. C. Kanellakis, and G. Grahne, "On the representation and querying of sets of possible worlds," in *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data Annual Conference*. ACM Press, 1987, pp. 34–48.

[19] T. Imielinski and W. L. Jr., "Incomplete information in relational databases," *Journal of ACM*, vol. 31, no. 4, pp. 761–791, 1984.

[20] N. N. Dalvi and D. Suciu, "Management of probabilistic data: foundations and challenges," in *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM, 2007, pp. 1–12.

[21] A. D. Sarma, O. Benjelloun, A. Y. Halevy, and J. Widom, "Working models for uncertain data," in *Proceedings of the 22nd International Conference on Data Engineering, ICDE*. IEEE Computer Society, 2006, p. 7.

[22] 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 *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*, 2008, pp. 687–700.

[23] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom, "ULDBs: Databases with uncertainty and lineage," in *Proceedings of the 32nd International Conference on Very Large Data Bases*. ACM, 2006, pp. 953–964.

[24] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: a probabilistic threshold approach," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*. ACM, 2008, pp. 673–686.

[25] J. Pei, M. Hua, Y. Tao, and X. Lin, "Query answering techniques on uncertain and probabilistic data: tutorial summary," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*, 2008, pp. 1357–1364.

[26] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Survey*, vol. 41, no. 3, 2009.

[27] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[28] E. M. Knorr and R. T. Ng, "A unified notion of outliers: Properties and computation," in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, 1997, pp. 219–222.

[29] ——, "Algorithms for mining distance-based outliers in large datasets," in *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases*, 1998, pp. 392–403.

[30] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000, pp. 93–104.

[31] A. Arning, R. Agrawal, and P. Raghavan, "A linear method for deviation detection in large databases," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996, pp. 164–169.

[32] X. Song, M. Wu, C. M. Jermaine, and S. Ranka, "Conditional anomaly detection," *IEEE Transaction on Knowledge and Data Engineering*, vol. 19, no. 5, pp. 631–645, 2007.

[33] C. C. Aggarwal and P. S. Yu, "Outlier detection with uncertain data," in *Proceedings of the SIAM International Conference on Data Mining, SDM*, 2008, pp. 483–493.

[34] B. Wang, G. Xiao, H. Yu, and X. Yang, "Distance-based outlier detection on uncertain data," in *Ninth IEEE International Conference on Computer and Information Technology*, 2009, pp. 293–298.