Knowledge-Based Systems 51 (2013) 60–71

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Outlier detection on uncertain data based on local information

Jing Liu *, HuiFang Deng

Department of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, PR China

ARTICLE INFO

Article history:
Received 9 January 2013
Received in revised form 28 May 2013
Accepted 11 July 2013
Available online 20 July 2013

Keywords:
Outlier detection
Uncertain data
Local information

ABSTRACT

Based on local information: local density and local uncertainty level, a new outlier detection algorithm is designed in this paper to calculate uncertain local outlier factor (*ULOF*) for each point in an uncertain dataset. In this algorithm, all concepts, definitions and formulations for conventional local outlier detection approach (*LOF*) are generalized to include uncertainty information. The least squares algorithm on multi-times curve fitting is used to generate an approximate probability density function of distance between two points. An iteration algorithm is proposed to evaluate *K–η–distance* and a pruning strategy is adopted to reduce the size of candidate set of nearest-neighbors. The comparison between *ULOF* algorithm and the state-of-the-art approaches has been made. Results of several experiments on synthetic and real data sets demonstrate the effectiveness of the proposed approach.

1. Introduction

In recent years, a large amount of uncertain data are generated and collected due to new techniques of data acquisition, which are widely used in many real-world applications, such as wireless sensor networks, meteorology and mobile telecommunication. However, due to randomness and complexity caused by uncertain data, it is difficult to deal with using traditional data mining algorithms for deterministic data. Therefore, many researchers put efforts in developing new techniques of data processing and mining on uncertain data [2,6,24].

Outlier detection is one of the key problems in data mining area which can reveal rare phenomenon and behaviors, find interesting patterns, and have significant applied merits in many fields, such as detection of credit card fraud, network intrusion and abnormal climate. There are many published works on outlier detection [9,19,37]. Inclusion of uncertainty to the data makes the problem far more difficult to tackle, as this will further limits the accuracy of subsequent outlier detection. Therefore, how to effectively detect outliers on uncertain data is of great importance.

There are many challenges raised ahead to affect outlier detection on uncertain data:

How to fully employ uncertainty information of uncertain data? The uncertainty level of a point in an uncertain dataset refers to the degree to which the point is uncertain. The uncertainty level of an uncertain dataset refers to the degree to which the dataset is uncertain. Without considering uncertain information, even for an effective conventional method of outlier detection, it is hard to get accurate results on uncertain data. Let us use a 2-dimensional uncertain data set to explain. In Fig. 1, there are 12 circles of different radius and each circle indicates the range of possible value of a point. The larger the size of the circle, the higher the uncertainty of the point. If we apply the traditional local outlier detection algorithm (*LOF*) [8] to detect outlier on this dataset using distance expectation value to represent the distance between two different points, it is easy to recognize that the 8th-point is an outlier, because it is isolated from the neighbors, but the 9th-point would be probably missed as an outlier, as on the whole, the distance expectation value of it to its neighbors is roughly the same as normal points. However, its uncertainty level is relatively higher because the radius is larger than that of neighbors. As a result, there is a great probability that the 9th-point is an outlier. These observations show that the local uncertain behavior plays an important role in the process of detecting outlier on uncertain data. As it is stated in [5]. "In general, a higher level of uncertainty of a given record can result in it behaving like an outlier, even though the record itself may not be an outlier". So it is meaningful to develop a new method which can fully employ local uncertainty behavior of uncertain data to detect outlier.

How to efficiently evaluate K nearest-neighbor (*KNN*) for a given query point in uncertain data? For deterministic data, the set of K nearest-neighbors for a given query point is clear and no ambiguity. But the situation on uncertain data is quite different. For example, we need to find 5-nearest-neighbors for query point q in Fig. 1. There are only three points (o_1, o_2, o_3) that are entirely contained within radius R while four other points (o_4, o_5, o_7, o_{10}) are partially covered. Obviously, points of o_1, o_2, o_3 are what we are searching for. But which would be the next two neighbors? In fact, any two points among the four points (o_4, o_5, o_7, o_{10}) have

* Corresponding author. Tel.: +86 20 87114028; fax: +86 20 87114638.
E-mail addresses: liujing.dm@qq.com (J. Liu), hdeng2008@gmail.com (H. Deng).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.07.005

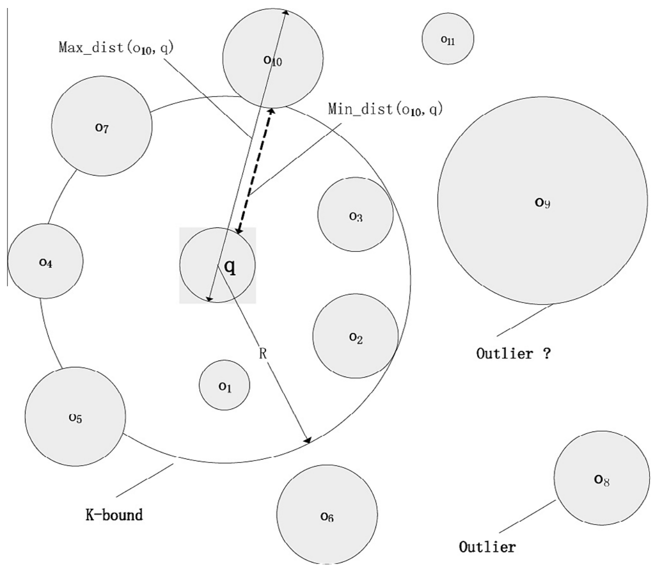


Fig. 1. The distribution of points in uncertain data.

a certain possibility to be the choice. Unfortunately, it needs an exponential amount of time to list every possible combination. This poses particular challenges for *KNN*-based algorithms, such as local outlier factor (*LOF*) algorithm. Therefore, how to effectively solve the problem of *KNN* query on uncertain data can be regarded as one of the most challenging problems in outlier detection on uncertain data.

Our contribution First, an outlier detection algorithm based on local information, termed as *ULOF* is proposed to assign an uncertain local outlier factor for each point in uncertain data set. According to the probability model of uncertain data, extended definitions which can exploit full uncertainty information are given in *ULOF* algorithm. Second, two optimization strategies are proposed to improve the efficiency of *K–η–distance* calculation and *K*-nearest neighbors selection. Third, characteristics of *ULOF* algorithm are analyzed in detail, and the influence of several mainly considered parameters on detection accuracy is investigated. At last, several experiments have been done to demonstrate the effectiveness and efficiency of *ULOF* algorithm.

The rest of this paper is organized as follows. In Section 2 we discuss related previous work. In Section 3 we give a detailed description of *ULOF* algorithm as well as two optimization strategies. The experimental results are presented in Section 4. The paper is concluded in Section 5.

2. Related work

There are many data mining techniques that have been proposed to solve corresponding problems on uncertain data, such as clustering [4,27], classification [30], frequent pattern mining [3] and probabilistic skylines [1,14,18]. In this section, we start with an introduction to two important models of uncertain data, and then give a brief review of previous works on outlier detection on deterministic data and uncertain data.

2.1. Uncertain data models

Models which are used to describe uncertain data are the basis for further research. Several approaches have been proposed to represent uncertainties. Fuzzy set theory [22], is an extension of classical set theory where set membership is defined as a possibility distribution. In possibility theory [23], uncertainty is described both by dual possibility and necessity measures. Rule-based modeling [7] is an approach that uses a set of rules to infer uncertainty and imprecision. The Dempster–Shafer theory of evidence [11], also known as the theory of belief functions, is an approach to obtain degrees of belief for one question from subjective probabilities for a related question and combine degrees of belief derived from independent items of evidence. Probabilistic models are widely used in many real applications to describe uncertain dataset. In this paper, we mainly focus on uncertainty in a probabilistic setting. Based on semantics of possible world [2,15], there are two main types of models of uncertain data: (1) attribute-level uncertainty model and (2) tuple-level uncertainty model.

In attribute-level uncertainty model, each attribute of a record is associated with an independent probability distribution function. In other words, the possible values for each attribute of a record are determined by a function that takes values based on a probability distribution. For example, when detecting air content, humidity and temperature information in a region, each of them would be subject to a certain independent probability distribution.

In tuple-level uncertainty model, each tuple is subject to its own probability of appearing in a possible world. These tuples are generally independent of each other as knowing one tuple would not be associated with others.

The attribute-level uncertainty model and tuple-level uncertainty model are equivalent in discrete case and can be transformed to each other [13,15]. Without loss of generality, we use attribute-level uncertainty model to describe uncertain data in this paper.

2.2. Outlier detection on uncertain data

There are many outlier detection methods on deterministic data in the literature. These techniques can be classified into five categories. They are statistical methods, distance-based methods, density-based methods, depth-based methods and angle-based methods. Statistical methods[12,25] (also known as model-based methods) assume that most data points are generated by a statistical model, while outliers do not fit the model. A distance-based outlier could be detected if there are not enough neighbor points within a specified distance [35,36]. A point is identified as density-based outlier if its density is relatively lower than that of surrounding points [8,20,28]. Depth-based approach [10,16] is based on the idea that outliers are located at the border of data region while normal points lie in the center. Angle-based methods [21,29] are especially useful for high-dimensional data, which argues that the variance of the angles between an outlier and other points is smaller than that of normal points.

Below, we will give a brief introduction to several studies in recent years of outlier detection on uncertain data:

Aggarwal et al. [5] first proposed a density-based outlier detection technique on uncertain data. They assumed that an outlier shows up in at least one subspace such that whose density is abnormally low. Therefore, to determine whether the point is an outlier, the density is calculated for each subspace where the point lies in. Formally, a point X is defined as a (δ,η)-*outlier* if it is lying in a subspace such that the probability of whose density being greater than η is lower than δ.

Wang et al. [34] first proposed distance-based outlier detection method on uncertain data. The tuple-level uncertainty model is used to describe uncertain dataset. Each tuple in uncertain dataset is affiliated with a confidence value to describe the probability of the tuple appearing in a possible world. A point in uncertain dataset can be considered as a distance-based outlier if the probability that we can find enough neighbor points within a specified distance is very low. The formal definition is that a point o is an $(up,$-

ud, λ)*-outlier* if the probability that there are at least $(1 - up) \times |D|$ points within the distance ud is less than λ. Here, $|D|$ is the size of dataset. Moreover, in Ref. [32], the authors focused on distance-based outlier detection on uncertain data of Gaussian distribution.

In Ref. [26], the authors proposed to build a global classifier for outlier detection by incorporating local uncertainty confidence information obtained by *LOF* or *K* mean clustering algorithm into a generalized Support Vector Data Description (*SVDD*) framework. For a point *X*, if its distance to the center of hyper sphere is greater than *R*, i.e. $\|X - O\|^2 > R$, *X* is considered as an outlier. Otherwise, it is accepted as a normal data point. But the percentage of outliers in dataset is extremely small compared with the percentage of normal data point. So it is difficult to build a clear boundary between normal data and abnormal data.

Jiang and Pei [13] first proposed to detect abnormal instance as well as abnormal object on uncertain data. In their model, each object contains a number of instances. They assumed that objects with similar properties tend to have similar instance features. They measure the normality of instances by learning conditional distribution of dependent attributes and then detect the outlier object, most of whose instances are outliers. Formally, all instances r_{ij} and all objects R_i are considered as outliers if $nor(r_{ij}) < \delta$and $nor(R_i) < \delta$, where $nor()$ denotes the normality of an instance or an object.

All above four algorithms have a common problem that they cannot assign a score (say, a *ULOF* as proposed here) to each point to show the outlier degree. Though the local uncertainty information is used to build a global classifier, the *SVDD*-based algorithm is still a binary-method. And it has difficulty in establishing an effective classifier if the number of outlier is very small. The distance-based outlier detection algorithm and the density-based outlier detection algorithm take a global view when deciding whether a point is an outlier. However, the locality information of data is ignored by both of them which may result in loss of accuracy on the datasets with a very complicated structure. In many real-world applications, a point can be recognized as an outlier just because its properties and behavior are very different from those of its neighborhoods. In our work, we assign each point an uncertain local outlier factor (*ULOF*) by calculating local density and local uncertainty level of each point to describe the degree of it being an outlier. In this way, we can identify more number of real outliers from the uncertain dataset with a very complicated structure. Moreover, the score of uncertain local outlier factor can let us more explicitly explain and describe an outlier in the uncertain dataset.

3. Local outlier detection on uncertain data

For more effective discussion, we first give some necessary notations and definitions of the probabilistic model of uncertain dataset. And then we will give the detailed formulation of the *ULOF* algorithm, which extends the concept of local density-based outlier detection on deterministic data [8] by fully utilizing uncertainty information on uncertain data. Next, we focus on three important subsections of the calculation process of *ULOF* algorithm. And a detailed discussion is given in the last part of this section. Table 1 lists all the commonly used terminologies in this paper.

3.1. Definition of uncertain local outlier factor on uncertain data

Let us assume that there are *n* points in uncertain dataset *D*. Each point has *d* attribute fields, known as *d* dimensions. o_i denotes the *i*th point in the database. Probability density function (*PDF*) of the *j*th dimension of o_i is denoted by $f_i^j(.)$ and cumulative distribution function (*CDF*) is denoted by $F_i^j(.)$. For simplicity, we use $f_i(.)$ and $F_i(.)$ to denote *PDF* and *CDF* of all dimensions of point o_i.

Table 1
Summary of the symbols.

Symbol	Description
D	Uncertain dataset
L	An order of points in dataset D
o_i	The *i*th point of $D(i = 1, \ldots, \|D\|)$
$f_i^j(.)/F_i^j(.)$	*PDF*/*CDF* of *j*th dimension of o_i
$Fd_{ij}(.)/\ FD_{ij}(.)$	*PDF*/*CDF* of the distance between o_i and o_j
$K\text{–}\eta\text{–}distance(.)$	The minimum distance that the probability of at least K nearest neighbors within it is not less than η
$N_{K\text{–}\eta\text{–}distance}(.)$	$K\text{–}\eta\text{–}distance$ neighborhoods of object o_i
$A_o(q)$	The probability of the point q that is $K\text{–}\eta\text{–}distance$ neighbor of point o
$P_o(k\_\ \mathrm{d})$	The probability that point o has at least K neighbors within k_d distance
$Min\text{–}dist(q,o)$	Minimum distance between q and o
$Max\text{–}dist(q,o)$	Maximum distance between q and o
$K\text{–}min\text{–}max(o)/K\text{–}min\text{–}min(o)$	The Kth minimum value among all the $Max\text{–}dist(q,o)/Min\text{–}dist(q,o)$, $q \in D - \{o\}$
$P(S,o_i,k)$	The probability that possible world S occurs
$P(L,o_i,k)$	The probability that o_i lies in the kth position under the order L
$P(D,i,j)$	The probability of j points that appears in the first i points under the order L

Definition 1. Given two different points o_i, $o_j \in D$, the distance between o_i and o_j is denoted by r_{ij}. The *PDF* of r_{ij} is denoted by $fd_{ij}(.)$ and the *CDF* of r_{ij} denoted by $FD_{ij}(.)$.

Extra care needs to be taken in Definition 1 that there is no specific order in point o_i and point o_j. As a result, r_{ij} is equivalent to r_{ji}, so as the $fd_{ij}(.)$ and $FD_{ij}(.)$. Due to the complexity of the distribution of each point in uncertain dataset, it is not easy to directly derive the formula for $fd_{ij}(.)$ and $FD_{ij}(.)$. In Section 3.2.1, an approximate polynomial distance function will be fitted by using least square algorithm.

Definition 2 (*K–η–distance of query point o*). $K\text{–}\eta\text{–}distance(o)$ is the minimum distance, within which the probability of a query point o having at least K nearest neighbors is not less than η(where K is a positive integer, $\eta \in\ (0,1]$).

Definition 3 (*K–η–distance neighbors of query point o*). Only if the minimum distance between the two points is less than $K\text{–}\eta\text{–}distance(o)$, a point q could be considered as one of $K\text{–}\eta\text{–}distance$ neighbors of query point o:

$$N_{k-\eta-distance}(o) = \{q | Min - dist(q,o) < k\text{—}\eta\text{—}distance(o)\}.$$

Unless otherwise specified, we use N_{k_η} as a shorthand for $N_{K\text{–}\eta\text{–}distance}$ in what follows. In this definition, the degree of a point q being one of $K\text{–}\eta\text{–}distance$ neighbors of query point o is denoted by a probability value $A_o(q)$. There are three special cases for $A_o(q)$.

Case 1. If the minimum distance between point q and o($Min - - dist(q,o)$, see Fig. 1) is bigger than $K\text{–}\eta\text{–}distance(o)$, then the probability value $A_o(q) = 0$. It also means that the point q would never be a $K\text{–}\eta\text{–}distance$ neighbor of point o.

Case 2. If the maximum distance between point q and o ($Max - dist(q,o)$, see Fig. 1) is smaller than $K\text{–}\eta\text{–}distance(o)$, then the probability value $A_o(q) = 1$. At the same time, it means that the point q is certainly a $K\text{–}\eta\text{–}distance$ neighbor of point o.

Case 3. If the $K\text{–}\eta\text{–}distance$ is greater than $Min - dist(q,o)$ but smaller than $Max - dist(q,o)$, then the probability value $A_o(q) = FD_{o,q}(k\text{–}\eta\text{–}distance)$, where $FD_{o,q}(.)$ is cumulative distribution function of r_{oq}.

Overall, case 1 and case 2 are two special cases of case 3. If cumulative distribution function and *K–η–distance* are known, it is easy to calculate the value of $A_o(q)$. However, this will also bring up a question: how to get the distance distribution function and the *K–η–distance*? Obviously, this question is the same as we have already raised in the Introduction: how to effectively evaluate K nearest-neighbor ($K-NN$) query on uncertain data. A detailed analysis and study of the question will be presented in next subsection.

Definition 4 (*Reachability distance of a point q to query point o*). The reachability distance of a point q with respect to query point o is defined as:

$$\text{reach-dist}_{k\_\eta}(\mathrm{q},\mathrm{o}) = \int_{Min-dist(q,o)}^{Max-dist(q,o)} fd_{oq}(r) \cdot max\{k{-}\eta{-}distance(o), r\}dr,$$

where fd_{oq} represents probability density function of r_{oq}.

In Definition 4, the actual distance value is replaced by *K–η–distance*(o) if the value is smaller than *K–η–distance*(o). In this way, we can reduce distance volatility.

Definition 5 (*Uncertain local reachability density of query point o*). The uncertain local reachability density of query point o is defined as:

$$ulrd_k(o) = \sum_{q\in N_{k\_\eta}} A_o(q) \Big/ \sum_{q\in N_{k\_\eta}} \text{reach-dist}_{k\_\eta}(o,q)\cdot A_o(q)$$

Note that the default value of $A_o(q)$ equals 1 in the classical *LOF* algorithm.

Definition 6 (*Uncertain Local outlier factor of query point o*). The uncertain local outlier factor of query point o is defined as:

$$ULOF_k(o) = \frac{\sum_{q\in N_{k\_\eta}} \frac{ulrd_k(q)}{ulrd_k(o)}\cdot A_o(q)}{\sum_{q\in N_{k\_\eta}} A_o(q)}$$

As a whole, the main difference between *ULOF* algorithm and *LOF* algorithm lies that the formulation of *ULOF* is given as a probability form which reflects local uncertainty information of points in uncertain data set.

3.2. Computing aspects

In this section, an approximate distance probability density function is derived by using least squares algorithm on multi-times curve fitting. Then, the evaluation of *K–η–distance* and an effective pruning strategy are presented in this subsection.

3.2.1. An approximate distance distribution function

According to the attribute-level uncertainty model, we already know $f_i(.)$ and $F_i(.)$ of each point o_i in uncertain dataset. We can use the inverse form of function of $F_i(.)$, denoted as $InF_i(.)$, to generate sample points which satisfy the density distribution of point o_i. The basic idea is to choose a number u drawn independently from a uniform distribution on the numbers between 0 and 1. The value x which is generated by the inverse function $InF_i(u)$, can be regarded as a desired sample point. This method has been proved to have a rock-solid working performance for most common probabilistic distributions[19].

Formally, the number of sample points for each point in uncertain dataset is set to m. So, for any two different points o_i, $o_j \in D$, there are $m*m$ different distance values. We break up the range of r_{ij} into intervals that do not overlap with each other. Then, we count the number of distance values which are contained within each interval. These statistical results can be used to generate a multi-times polynomial function $PolFun_{i,j}(.)$ by the least squares algorithm on multi-times curve fitting. The $PolFun_{i,j}(.)$ can be considered as an approximate distance cumulative distribution function of r_{ij}. So, the distance cumulative distribution function can be denoted as:

$$FD_{i,j}(r) = \begin{cases} 0, & r < Min-dist(o_i, o_j) \\ 1, & r > Max-dist(o_i, o_j) \\ PolFun(.), & \text{otherwise} \end{cases}$$

Distance probability density function $fd_{i,j}(.)$ can be derived by taking derivatives of $FD_{i,j}(.)$ with respect to distance value r. In Section 4, we can find that the two approximate functions work very well in all the experiments we have done.

3.2.2. K–η–distance evaluation

In this section, we give a detailed description of the computing process of determining *K–η–distance* and *K–η–distance* nearest-neighbors in the uncertain dataset D. Let us use $P_o(k\_d)$ to denote the probability that query point o has at least k neighbors within k_d, where k_d is a distance value. The calculation problem of $P_o(k\_d)$ can be efficiently solved by using an dynamic programming approach. In particular, $P_o(k{-}\eta{-}distance) = \eta$.

Lemma 1. *The value of $P_o(k\_d)$ is a monotonic increase (not strictly) function of k_ d value, where $k\_d \in (0, Rmax]$, Rmax is the maximum distance determined by any two points in uncertain dataset.*

Proof. For each point $q \in D$, the value of $A_o(q)$ will not decrease for a larger k_d value. In other words, the probability of a point q appearing within k_d distance of point o becomes bigger with increasing in k_d value. As a result, given a larger value of k_d, there will be more points with higher probability being the neighbors of point o. Therefore, we get Lemma 1. □

Algorithm 1. Iteration Algorithm

```
Input:
  Uncertain Dataset $D$, confidence $\eta$, maximum error $\varepsilon$, query
  object $o$;
Output:
  $k-\eta-distance$ and $k-\eta-distance$ nearest-neighbors $N$;
1: $N \leftarrow \emptyset$, $l = 0$, $up = Rmax$;
2: $k\_d = (l + up)/2$;
3: while $|P_o(k\_d) - \eta| \geqslant \varepsilon$ do
4:    if $P_o(k\_d) < \eta$ then
5:       $l = k\_d$;
6:    else
7:       $up = k\_d$;
8:    end if
9:    $k\_d = (l + up)/2$;
10: end while
11: $k-\eta-distance = k\_d$;
12: for each point $q \in D - \{o\}$ do
13:    if $Max-dist(q,o) < k-\eta-distance$ then
14:       insert $q$ into $N_{k\_\eta}$;
15:    end if
16: end for
17: return $k-\eta-distance$, $N_{k\_\eta}$;
```

Based on Lemma 1, a basic iteration algorithm for estimating *K–η–distance* value is proposed. The detailed process of the iteration algorithm is given in Algorithm 1. Essentially, this algorithm is just an example of a divide and conquer algorithm. The algorithm first sets k_d equal to the middle value of the possible *K–η–distance* range, then determines whether the value of $P_o(k\_d)$ comes bigger or smaller than the user-defined parameter η. If not, continue to search for the remaining half in the same manner or return the final result. This algorithm has a good convergence because the range of values allocated for k_d can be halved after each comparison. However, the range: $(0, Rmax]$, is too big to do a meaningful calculation. We will give a tighter upper bound and lower bound for k_d in Section 4.3.

The returned value of iteration algorithm is an approximation value of *K–η–distance* and *K–η–distance* nearest-neighbors set N_{k_η}. However, a simple, efficient and accurate method for the estimation of $P_o(k\_\ d)$ is needed in Algorithm 1. A very intuitive formula for $P_o(k\_\ d)$ is defined as following:

$$P_o(k\_d) = \sum_{S \subseteq D, |S| \geqslant k} P(S) = \sum_{S \subseteq D, |S| \geqslant k} \prod_{q \in S} A_o(q) \cdot \prod_{p \in D-S} (1 - A_o(p)),$$

From this formula we could see that it is going to take exponential amount of the time to list all the subsets who contains at least K points. Fortunately, a dynamic programming approach which offers a nearly linear complexity in time with the size of dataset and the value of K can be used to solve the calculation problem. We will give a simple description of the method as well as some simple proofs below.

To help understanding, a subset S is expressed as a possible world S. Give an arbitrary order L, the uncertain dataset D can be sorted as $D = \{o_1, o_2, \ldots, o_n\}$, where n is the size of uncertain dataset D. All the points which lie in any possible world also should be placed in order L. For example, if two points o_i and $o_j (i < j)$ appear in a possible world S, the point o_i must be in front of the point o_j. In the following discussions, we assume that all uncertain datasets and possible worlds are placed in a same order L.

The kth position in a possible world S is the key node in the calculation of $P_o(k\_d)$. To emphasize this, we extend symbol S to (S, o_i, k) to highlight the information that the kth position of possible world S is the point o_i. Within a given $k\_\ d$ distance radius around query point o, (1) $P(S, o_i, k)$ denotes the probability that possible world S occurs ($|S| \geqslant k$); (2)$P(L, o_i, k)$ represents the probability that o_i lies in the kth position; (3) $P(D, i, j)$ denotes the probability that there are exactly j points that appear in the first i points.

Lemma 2. *Given an order L for uncertain dataset D. The probability that query point o has at least K neighbors within k_d distance can be solved by the equation below:*

$$P_o(k\_d) = \sum_{i=k}^{n} P(L, o_i, k) = \sum_{i=k}^{n} A_o(o_i) \cdot P(D, i-1, k-1)$$

Proof.

1. Under the order L, each point in $\{o_1, o_2, \ldots, o_{k-1}\}$ is impossible to be the kth point of any possible world, i.e., $P(L, o_i, k) = 0$, $(i = 1, 2, \ldots, k-1)$. As a result, the kth point of each possible world S ($|S| \geqslant k$) can only be discovered in $\{o_k, o_{k+1}, \ldots, o_n\}$ and is unique. Therefore, the sum of $P(L, o_i, k)$ for all $o_i (i = k, k+1, \ldots, T)$ is exactly equal to $P_o(k\_\ d)$.
2. For each $P(L, o_i, k)$, point o_i must appear in the kth position. So, except for point o_i, there are exactly $k-1$ points appearing in the first $i-1$ points of uncertain dataset D under the order L. So, we have the equation $P(L, o_i, k) = A_o(o_i) \cdot P(D, i-1, k-1)$. □

Based on Lemma 2, instead of traversing all the subsets of uncertain dataset D, we only need to calculate the value of $P(D, i-1, k-1)$. Moreover, depending on whether point o_{i-1} appears in the $(k-1)$th position, the computation of $P(D, i-1, k-1)$ can be divided in two subproblems:

$$P(D, i-1, k-1) = A_o(o_{i-1}) \cdot P(D, i-2, k-2) + (1 - A_o(o_{i-1})) \cdot P(D, i-2, k-1)$$

Obviously, this is a typical recursive procedure and can be realized by a dynamic programming method with a matrix of size $(k-1) \times (n-1)$. A more detailed discussion can be found in [34] where initial conditions, boundary condition and the detailed computing processes were given.

3.2.3. An effective pruning strategy

The basic approach discussed above considers the whole uncertain dataset D as the candidate set for *K–η–distance* nearest-neighbor. However, there are many points that can never be a *K–η–distance* nearest-neighbor of query point o. In order to get a faster computing, we should remove these impossible points before conducting iterative calculation. Here, we propose a pruning strategy to reduce the scale of candidate set. At the same time, the lower and upper bounds of *K–η–distance* can be derived based on this strategy.

Definition 7 (*K–min–max distance of query point o*). The *K–min–max* distance of query point o is the kth minimum value among all $Max\text{–}dist(q, o)$, where $q \in D - \{o\}$.

Definition 8 (*K–min–min distance of query point o*). The *K–min–min* distance of query point o is the kth minimum value among all $Min\text{–}dist(q, o)$, where $q \in D - \{o\}$.

Lemma 3. *For any point $q \in D - \{o\}$, if $Min\text{–}dist(q,o) > K -\ min -- max(o)$, then point q can never be a K–η–distance nearest-neighbor candidate point. In other words, the probability of point q being K–η–distance nearest-neighbor of o equals 0.*

Proof. From Definition 7, we know that there are at least k points lying entirely within $k\text{–}min\text{–}max(o)$. That is, the value of *K–η–distance*(o) is not bigger than $k\text{–}min\text{–}max(o)$. Therefore, it is easy to reach the conclusion of Lemma 3 by the definition of *K–η–distance* neighborhood. □

Lemma 4. *For any point $q \in D - \{o\}$, if $Max\text{–}dist(q,o) < K\text{–}min\text{–}min(o)$, point q must be a K–η–distance nearest-neighbor, i.e. the probability of point q being K–η–distance nearest-neighbor of o is equal to 1.*

Proof. From Definition 8, we know that there are at most k points lying within $K\text{–}min\text{–}min(o)$. In other words, *K–η–distance*$(o) >= k\text{–}min\text{–}min(o)$. Therefore, for any point $q \in D\text{–}\{o\}$, if $Max\text{–}dist(q,o) < K\text{–}min\text{–}min(o)$, it implies that point q lies entirely within *K–η–distance*(o). So, point q must be one of *K–η–distance* neighborhoods of query point o. □

Lemma 5. *For a query point o, the possible value for K–η–distance(o) is within the range of [K–min–min(o), K–min–max(o)].*

From Lemmas 3 and 4, It is not difficult to get the conclusion of Lemma 5. Based on Lemma 5, the lower-bound and upper-bound of *K–η–distance*(o) are $K\text{–}min\text{–}min(o)$ and $K\text{–}min\text{–}max(o)$, respectively. Obviously, we can get a better convergence rate for the iteration algorithm in this narrower range.

Algorithm 2. Pruning Algorithm without indexing structure

```
Input:
  Uncertain Dataset $D$, query object $o$;
Output:
  Candidate set $C$, set $E$;
1: $C \leftarrow \emptyset$, $S \leftarrow \emptyset$;
2: Using max-heap and min-heap of size $K$ to calculate
   $K - min - min(o)$ and $K - min - max(o)$, respectively;
3: for each point $q \in D - \{o\}$ then
4:    if $Max - dist(q,o) < K - min - min(o)$
5:       insert $q$ into $E$;
6:    else if $Min - dist(q,o) < K - min - max(o)$
7:       insert $q$ into $C$;
8:    end if
9: end for
10: return $E$, $C$;
```

Based on the above lemmas, a pruning algorithm is proposed to choose a relatively small candidate set for *K–η–distance* nearest-neighbors of query point *o*. Algorithm 2 describes the general calculation process of pruning algorithm without using indexing structure. In Algorithm 2, the computation of $K\text{–}min\text{–}min(o)$ and $K\text{–}min\text{–}max(o)$ can be completed by scanning the database for only one time with a max-heap/min-heap of size *K*. At the same time, *U-tree*—a multi-dimension information indexing structure is proposed in [33] to index multi-dimensional uncertain data with arbitrary probability density functions. Moreover, the authors there also gave an efficient method on range retrieval on multi-dimensional imprecise data. Therefore, we can use *U-tree* to speed up the calculation process of pruning. All prune operations can be completed in a nearly logarithmic time scale with the index structure considered. Due to space limitation, here we cannot fully explore the use of *U-tree* in detail.

After pruning algorithm is executed, only the points that lie in the relatively small set *C* need to be considered as *K–η–distance* nearest-neighbor candidate points. The points that lie in the set E are determined to be *K–η–distance* nearest-neighbor of query point *o*: $A_o(q) = 1$, for any point $q \in E$. In Section 4.3, we will show the efficiency of pruning algorithm.

3.3. The analysis of Ulof

Time complexity—We name the basic algorithm discussed in Section 3.2.2 as *BULOF*, and the optimized algorithm discussed in Section 3.2.3 as *ULOF*. The size of candidate set *C* is denoted by $|C|$. According to the discussions above, the run time for *ULOF* is made up of several parts. First, we need to fit approximate distance density functions of query point *o* to each point in candidate *C*. The time complexity of this part is $O(m^2n)$ for *BULOF* and $O(m^2|C|)$ for *ULOF*, where *m* is the number of sample points for each uncertain point. Because the distance is symmetrical, the two distance distribution functions D_{ij} and D_{ji} need to be computed for only once. Second, the time complexity of executing pruning algorithm is $O(n \cdot log(k))$. And it can be reduced to $O(log(n) \cdot log(k))$ if *U-Tree* index structure is used on uncertain data *D*. *BULOF* algorithm cannot deliver this because it does not include the pruning tactics. Third, the time complexity of iteration algorithm to estimate *K–η–distance*(o) value is $O((k-1) * (|C|-1) * ITE)$ for *ULOF* and $O((k-1) * (n-1) * BITE)$ for *BLOF*, where *ITE* and *BITE* are the average number of iterations for *ULOF* and *BULOF*, respectively. Therefore, in the worst case, the total time complexity is $O(n(m^2|C| + log(n) \cdot log(k) + (k-1) * (|C|-1) * ITE))$ for *ULOF* algorithm and $O(n(m^2n + (k-1) * (n-1) * BITE))$ for *BULOF* algorithm. Compared with the big size of data set *D*, $|C|$ is a very small number, $|C| \ll n$. And for a tighter bound, the value of *ITE* has reason to be smaller than that of *BITE*. So the advantage of *ULOF* algorithm is prominent.

The generalization—In general, *ULOF* algorithm mentioned here can be regarded as a generalization of *LOF* algorithm. In *ULOF*, attribute values for each point are uncertainty. Deterministic data is a special case of uncertain data which uncertain level being zero. Again, just as an example, we use distance between two points to explain this. In deterministic data, the distance value between two points is equal to some fixed value. While in uncertain data, the distance is some kind of probability density function.

Let us summarize the generalization of concepts and definitions on uncertain dataset as follows:

1. The attribute values used to describe the points are no longer fixed values; rather, they are some types of probability distribution functions.
2. The distance between two points is also a kind of distance density function.
3. Instead of relying on a simple binary decision, we use probability value $A_o()$ to measure the degree of a point being a neighbor.
4. The neighborhoods of a query point are no longer very clear in uncertain dataset. In *ULOF* algorithm, we use a probability value to express the degree that there are at least *K* neighborhoods within *K–η–distance*.

Obviously, the local information based outlier detection method, with taking into account the most of local density information and local uncertain level information is of critical application significance in finding outlier on uncertain dataset with complicated data structure. In contrast, without concern for uncertain level information, even a lovely traditional outlier detection algorithm invented to solve the special case cannot be applied on a general case directly. The applicability of *ULOF* algorithm is not only formulated in theory here, but the experimental results showed as below also demonstrate its effectiveness.

4. Experimental results

We have conducted several experiments to examine the effectiveness and efficiency of our method on two real datasets from UCI Machine Learning repository as well as three synthetic datasets. The experimental setup is described in Section 4.1. Result analysis is presented in the following sections. All experiments run on a *HP xw*6600 workstation with 2 Intel Xeon DP *E*5440 2.83 GHz CPUs and 8 GB main memory.

4.1. Experimental setup

In this paper, each considered cluster in the dataset is produced by different mechanisms, i.e., the density and degree of uncertainty for each cluster are different.

For the synthetic datasets, we use a Gaussian mixture model consisting of three different classes to generate normal data points. And there are other 100 points that are generated by an independent uniform distribution model on the complete data space for each synthetic dataset. Because the 100 points are generated by a mechanism different from the one for three main clusters, so they can be considered as outliers in each dataset. Three artificial datasets of 10-dimension are generated to assess the performance of our method, the sizes are 10 K, 50 K and 100 K.

For the real data case, we employ Localization Data for Person Activity Data Set (*LDPA*) [17] and *MAGIC* Gamma Telescope Data

Table 2
Default values for main parameters.

Parameters	Default values
m (Number of samples for each point)	500
K (The number of neighbors)	10–35
η (Degree of having K neighbors)	0.85
f (Uncertain level)	1.5

Set (*MAGIC*) [31] from the UCI Machine Learning Repository as two real datasets. The data set *LDPA* contains recordings of five people performing different activities. Each person wears four sensors (tags) while performing the same scenario five times and each instance is a localization data for one of the tags. There are 7 attributes (not including class attribute) and 155,952 instances (removing the class that the number of instances belongs to it is less than 5000) in *LDPA* dataset. The data of *MAGIC* are generated to simulate registration of high energy gamma particles in an atmospheric Cherenkov telescope. There are 10 attributes (not including the class attribute) and 19,020 instances in *MAGIC* dataset.

In order to model the uncertainty level of the 5 datasets, we use the same method as presented in Ref. [5] to add uncertainty level to each dataset. Each attribute of a point is described by a normal probability density function with mean μ and variance σ^2. The mean μ is equal to original attribute value. So the degree of uncertainty for each attribute is determined by standard deviation σ. The parameter $\sigma$is drawn from a uniform distribution in the range $[0, 2 \cdot f] \cdot \sigma^*$, where σ^* is standard deviation of each dimension in the underlying data. Thus, the dataset can be set in different uncertainty level by changing the value of *f*. We can observe the change of accuracy and running time with different uncertainty level *f*. It should be noted that a higher value of *f* indicates a higher uncertainty level of the dataset. In order to make sense in discussions, the maximum value of *f* is not greater than 3 in all experiments.

To make a detailed comparison, in addition to basic outlier detection algorithm (*BULOF*) and optimized algorithm (*ULOF*), we also implemented *DensitySamp* algorithm mentioned in [5], *DPA* algorithm mentioned in [34] (an optimized version (*GPA*) only suitable for 2-*D* space) and traditional *LOF* algorithm [8]. Both *DensitySamp* algorithm and *DPA* algorithm are the state-of-the-art methods for outlier detection on uncertain data. We tried many different parameter settings to achieve the best result of the two algorithms. And *LOF* algorithm is a well-established method for outlier detection on deterministic data. It should be mentioned that there are two other studies [13,26] in the literatures about detecting outliers on uncertain data. However, we will not compare our method to them because data models used in the two studies and the type of problems to be solved are entirely various from ours.

Moreover, in this paper, we use least squares algorithm to fit approximate distance distribution functions. Because least squares algorithm has been widely tested and verified in various practical applications, here the parameter settings of it are given directly. The number of sample points for each point is set to 500 ($m = 500$), the number of intervals is set to 1000 and the highest degree in the polynomial is set to 6. To facilitate reading, the default values for main parameters are displayed in Table 2 and several principal experimental results are following:

1. Pruning rate is proportional to the data size and inversely proportional to the uncertainty level.
2. The average number of iterations tends to be a certain small constant and increases with the level of uncertainty.
3. The *ULOF* algorithm can obtain a high precision when the value of uncertainty level is smaller than 2.5, but it is still open in testing on distinguishing outliers from normal points if *f* is too high.
4. By using the strategy of pruning, *ULOF* algorithm can get a better time performance than *BULOF*. However, the pruning strategy gradually loses its effect with the increasing uncertainty level.

4.2. The variations of k and η

The first interesting question in our experiments is that how *ULOF* value changes over arguments *K* and η? How to determine the value of parameter *K* is an open question. In reference [8], authors pointed out that local outlier factor (*LOF*) neither decrease nor increase monotonously for an increasing sequence of *K* values. Whether the conclusion is probably true as well for uncertain data? A similar experiment has been done to test the *ULOF* values for one of the three clusters in the 50 K artificial dataset. The minimum, maximum and mean, as well as the standard deviation of *ULOF*, are shown in Fig. 2. Default value for η is 0.85 and uncertainty level is set to 1.5. As can be seen from Fig. 2, with *K* value changing from 2 to 50, the maximum *ULOF* value goes up and down. However, the mean *LOF* value is very stable around 1. This shows that (1) *ULOF* value for most normal point is close to 1. (2) It is hard to find a reasonable value of *K* to obtain the best outcome. Following the advice in Ref. [8], we set a range for parameter *K*. For each point, we compute *ULOF* for each possible value of *K* in this range and choose the maximum *ULOF* value as the final result. So, we set the value of *K* not less than 10 and no greater than 35 in the remaining experiments.

How to determine a best value of ηalso should be given a careful study. The value of η indicates that the degree for each query point o has at least *K* nearest neighbors within *K*–η–*distance*. As a special case, the η value for *LOF* algorithm is equal to 1. However, in a general sense, the possible value for η is in range of (0, 1]. For detection

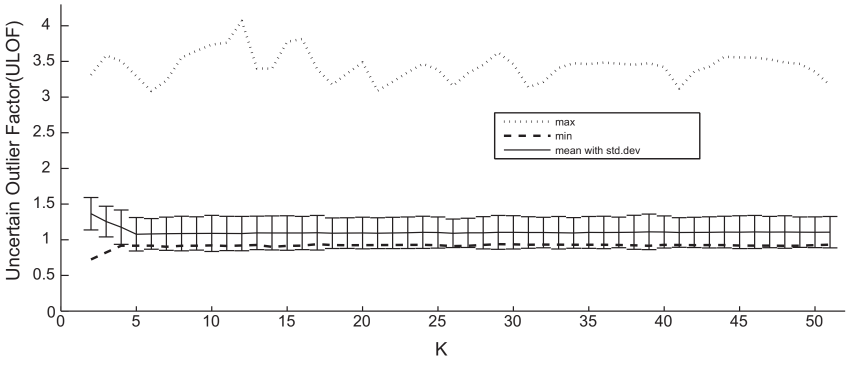
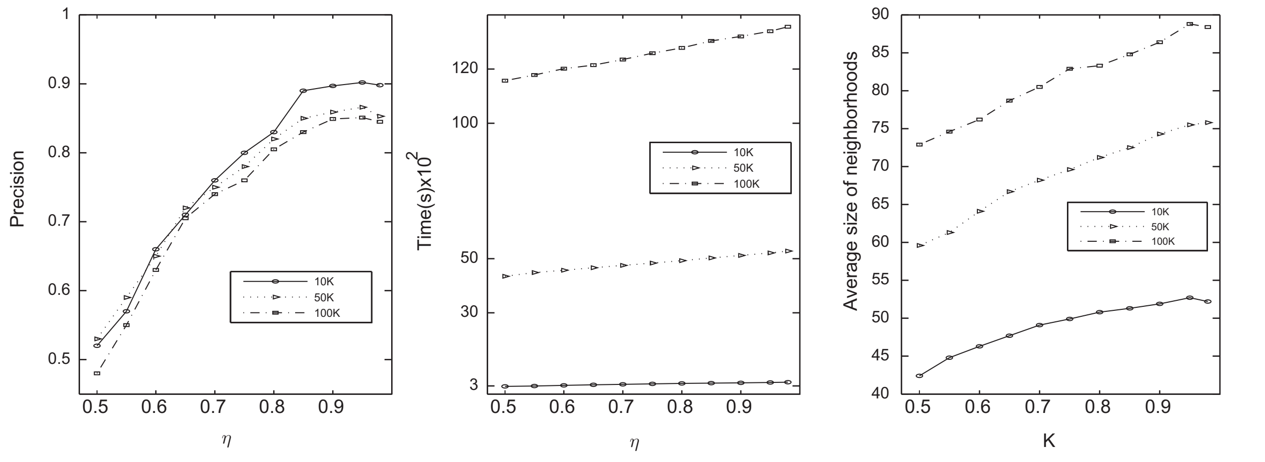
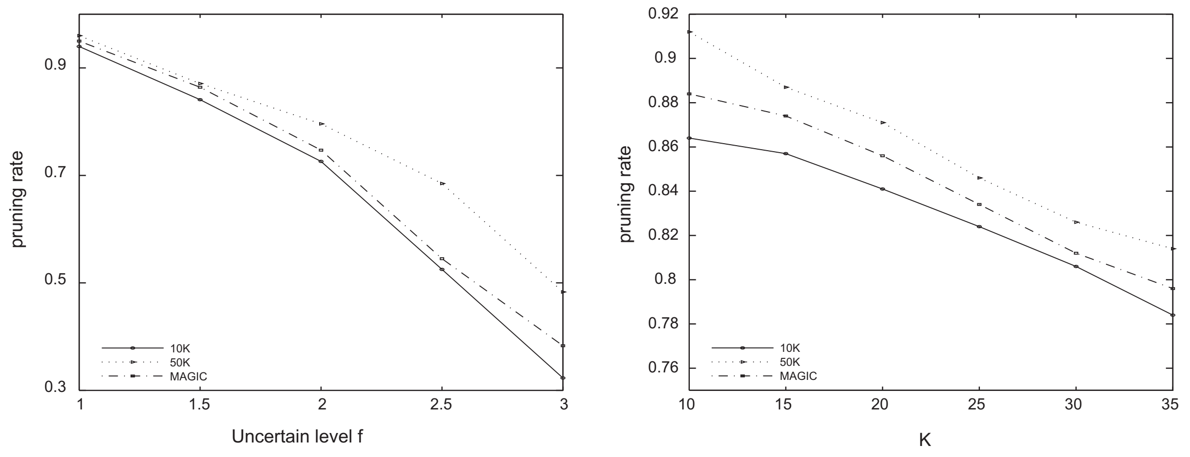


Fig. 2. Changes of *ULOF* value with different *K*.



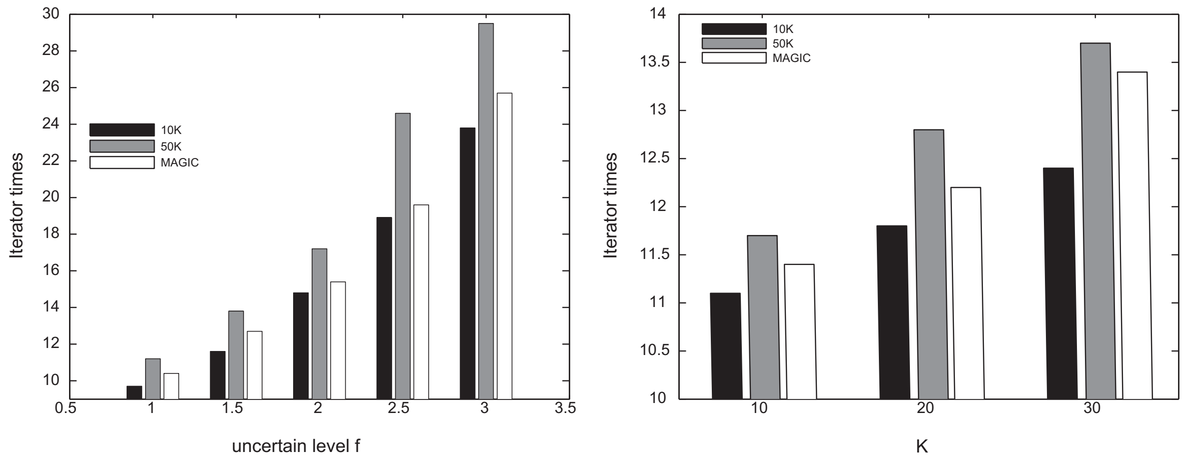
(a)Precisionchangeswithη. (b)Timechangeswithη. (c)Neighborhoodchangeswith η,k=20.

Fig. 3. The influence of parameter η.



(a) pruning rate with increasing uncertainly, K=20. (b) pruning rate with increasing K, f=1.5.

Fig. 4. Effectiveness of pruning algorithm.



(a) Iterator times with increasing uncertainly. (b) Iterator times with increasing K, f=1.5.

Fig. 5. Efficiency of iteration algorithm.

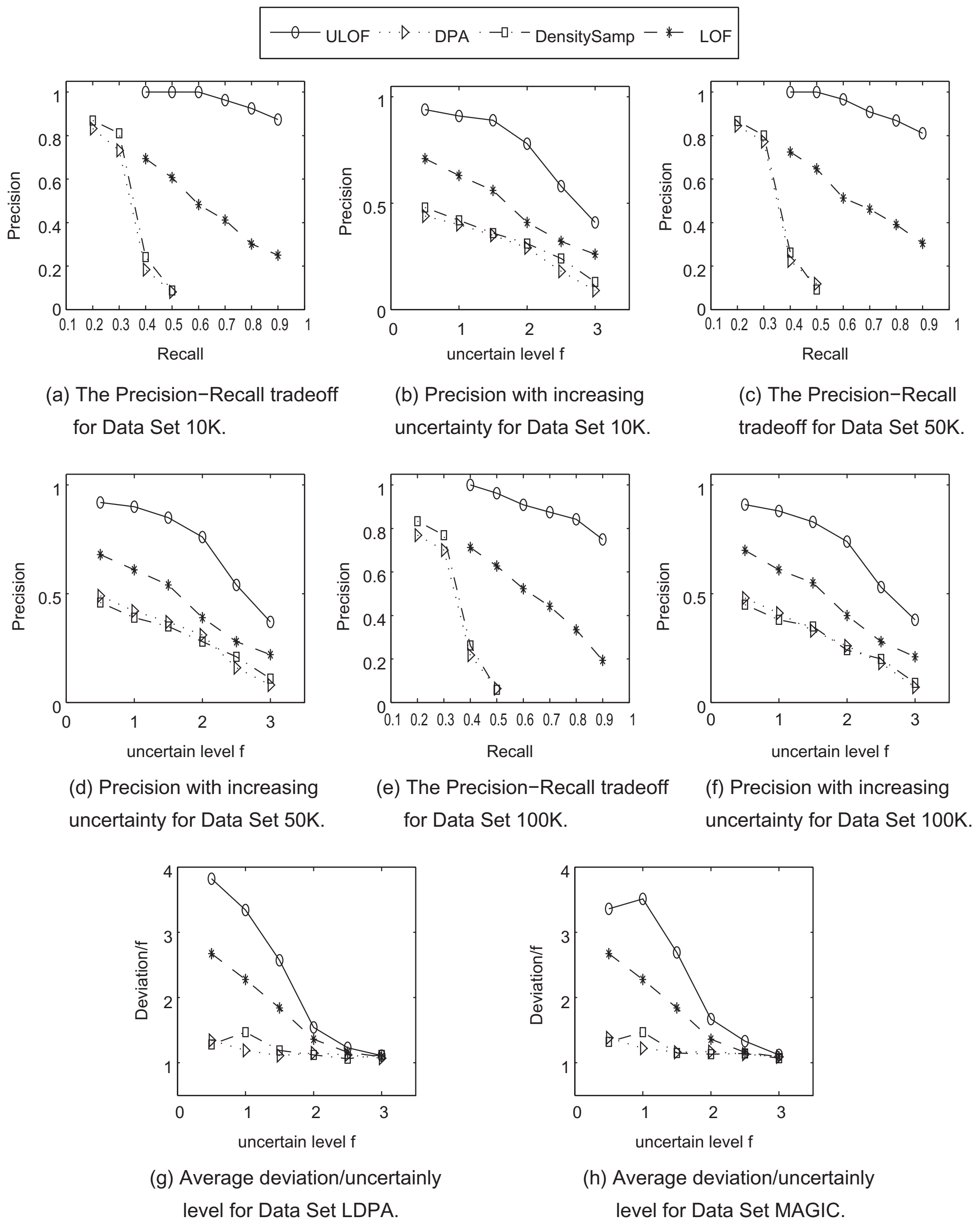


Fig. 6. Precision and recall on detecting outlier instances on five data sets.

precision, whether the value of η is the bigger the better? Fig. 3(a) shows the level of precision (the precision is expressed as the fraction of true outliers in the top 100 points, sorted by the score of *ULOF*) changing with varying ηvalue on 10 K, 50 K and 100 K datasets. Default value of uncertainty level is 1.5. From Fig. 3(a), we can see that the level of precision is improved with increasing of η value. However, the precision is stabilized at certain level when the value of η is more than 0.85. It means that 0.85 is a suitable value for η. Moreover, Fig. 3(b) shows the changing of running time over the three datasets. We can clearly see the running time increases with η. From Definition 3 we know that the number of neighborhoods for a point is not fixed, but more than the parameter K. The average number of neighborhoods for each dataset is showed in Fig. 3(c), which can come to a conclusion that the bigger η value, the more possible neighborhoods for each point in dataset and then more computing time is needed. For time-effectiveness trade-off, we set the value of η to be 0.85 in all remaining experiments.

4.3. Prune & iterator

Now, let us study the efficiency of pruning algorithm and iteration algorithm. Fig. 4 shows the scale of candidate set after pruning. As you can see in the figure: (1) average sizes of candidate set for all the three datasets is significantly less than their original dataset sizes; (2) we can get a higher pruning rate on a larger data set; (3) the efficiency drops in pruning algorithm with increasing of uncertainty level. Obviously, with most of irrelevant points removed, a vast number of unnecessary computations are avoided. However, from Fig. 4(a) we can find that there is a low pruning rate when the value of *f* is over 2.5. A reason for the change of pruning rate is that the range in which a point may appear will become wider if a higher uncertainty level is given. With increasing of the range, it is fairly hard to gain a sufficiently small value of *K–min–max*(o) which is used as a benchmark to filter out unnecessary points. From Fig. 4(b), we can see that the influence of pruning algorithm de-

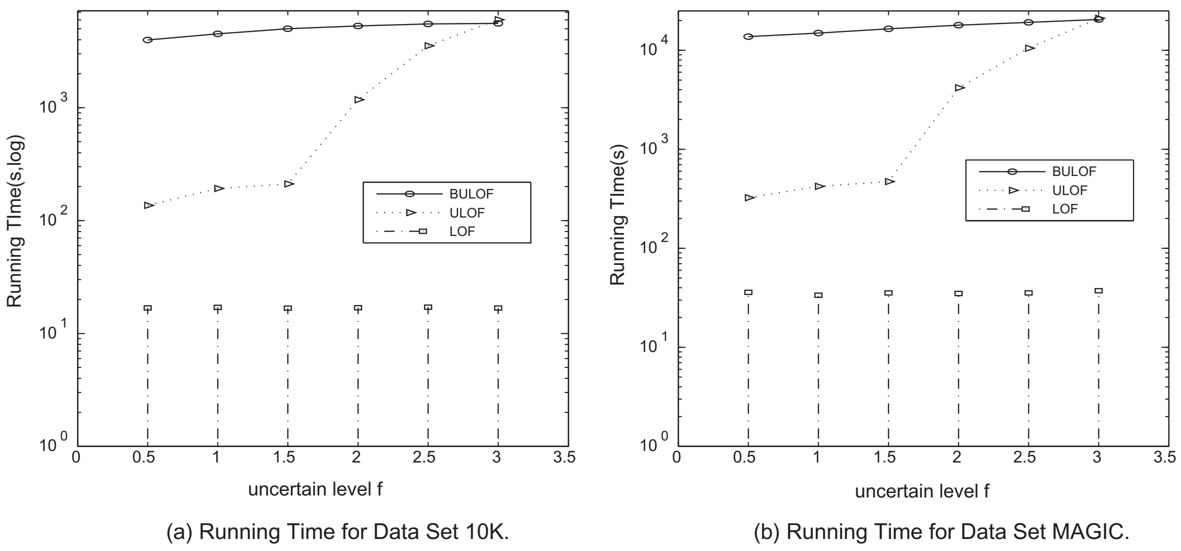


Fig. 7. The comparison of running time.

creases with increasing value of parameter *K*. Nevertheless, parameter *K* has a relatively small impact on ultimate effect. Because even the biggest value of *K*, we can still get a very high pruning rate. Therefore, the level of uncertainty is an important influence factor of the efficiency of pruning algorithm.

Based on Lemma 4, an iteration algorithm is raised to estimate the value of $K–\eta–distance(o)$. Convergence rate of iteration algorithm is showed in Fig. 5 by counting the mean iteration times. Fig. 5(a) and (b) shows convergence rate on two synthetic datasets and a real dataset with different parameter settings. From Fig. 5(a), it is plain to see that the efficiency of iteration algorithm is also impacted by uncertainty level. The average number of iterations tends to be a certain small constant, such as 12, when uncertainty level is low. What is more, the maximum average number of iterations is less than 30 in our experiment. As for different values of parameter *K*, we cannot see a significant change of mean iteration times in Fig. 5(b). Therefore, as far as iterations are concerned, the performance of iteration algorithm is very good. Surely, true running time analysis can give more revelation of iteration algorithm. We will discuss overall time performance of *ULOF* algorithm in next section.

4.4. Accuracy & running time

Accuracy—In this section, we give experimental results on accuracy and time performance on three synthetic datasets and two real datasets. Precision and recall are two important terms frequently used by outlier detection study. In this paper, precision is the fraction of points in resultant set that are indeed outliers and recall is the fraction of outliers that are detected. In other words, high recall means that most of outliers are found in resultant set. While high precision means that most points in current result set are definitely outliers.

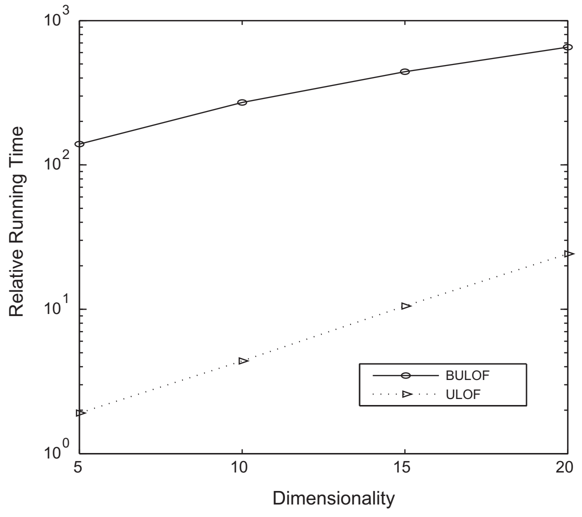
Fig. 6(a)–(f) shows results on three different synthetic data sets. Default value for uncertainty level is set to 1.5. Because some outliers lie in center of cluster, it is hard to pick out this kind of points from the entire dataset. So it can be observed that all methods have difficulty in detecting all outliers. Just as we have expected, *ULOF* algorithm can deliver the best results on all the three synthetic datasets. *DensitySamp* algorithm and *DPA* algorithm lose the accuracy as they should have been. It is because that the two methods determine outliers from a global perspective. They lose sight of local density feature and local uncertainty level feature. What is surprising is the traditional *LOF* algorithm gets a higher precision compared with the two global algorithms. This illustrates that local feature is an important factor for detecting outlier on uncertain dataset with a complex structure. However, with losing the information of uncertainty level, *LOF* algorithm does not perform as well as ours.

Again, we can see that the level of uncertainty has a significant impact on detection accuracy. When the level of uncertainty is low, most outliers in dataset can be detected. As the level of uncertainty increasing, *ULOF* algorithm has difficulty in distinguishing outliers from normal points. Moreover, *LOF* algorithm also obtains a high precision when the value of *f* is very small, which further confirmed that *LOF* algorithm is a special case of *ULOF* algorithm where uncertainty level is equal to zero.

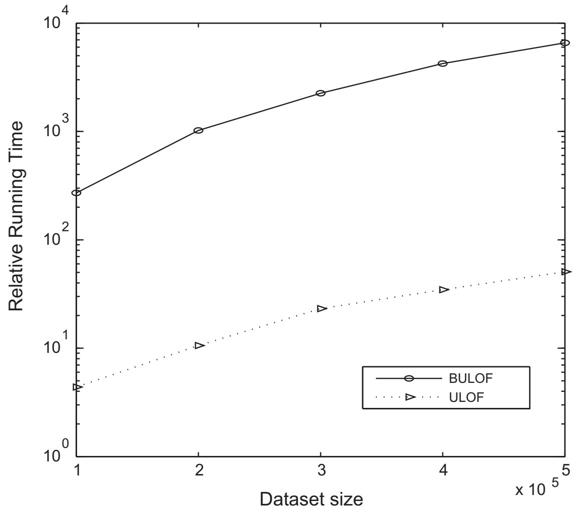
Because of different viewpoint on the notion of what outliers look like, it is hard to draw recall-precision graph for the two real data sets. Instead, we count the mean standard deviation σ for those points which are detected as outliers and then compare with the uncertainty level of uncertain dataset. The measurement may not be perfectly precise, but a larger deviation value can also indicate a higher degree of abnormality. Generally speaking, the trend of deviation curves in Fig. 6(g), and (h) is in accordance with that of precision curves in Fig. 6(b), (d), and (f). The level of the curve for *ULOF* is significantly higher than that of other curves. This means that the points detected by *ULOF*, to some extent, are true outliers. Furthermore, if uncertainty level is greater than 2.5, there is no effective way to find outliers.

Running Time—Here, we show running time performance of three different algorithms: *BULOF*, *ULOF* and *LOF*. Since accuracy is too low, *DensitySamp* algorithm and *DPA* algorithm are omitted in this experiment. The results of comparison are shown in Figs. 7 and 8.

From Fig. 7, it is easy to see that traditional *LOF* algorithm has the best running time performance in both the synthetic and real data sets. There are a number of factors that could account for this. (1) *LOF* algorithm does not need to fit approximate distance distribution function; (2) Neighborhoods of each point are evident in *LOF* algorithm; and (3) $K–\eta–distance$ used in *LOF* can be fixed promptly. However, *ULOF* algorithm, as a generalization form of *LOF* algorithm, increased a lot of extra computation. The comparison between *BULOF* and *ULOF* algorithm can further verify the efficiency of pruning algorithm. Because most unnecessary points have been pruned, *ULOF* algorithm can get a better time perfor-



(a) Scalability with increasing dimensionality, f=1.5, dataset size=100K.



(b) Scalability with increasing dataset size, f=1.5, dimensionality=10.

Fig. 8. Scalability with increasing dimensionality and dataset size.

mance than *BULOF*. Nevertheless, if the level of uncertainty is over 2.5, *ULOF* algorithm does not really have a big advantage in running time. Under a high uncertainty level, even plenty of extra time is spent to execute pruning strategy, can only *ULOF* algorithm obtain a very low pruning rate. As the difference in performance is noticeable to the three algorithms, we omit the same time curves with increasing level of uncertainty on other datasets. In Fig. 8, we compared the efficiency of *ULOF* and *BULOF* with increasing dimensionality and dataset size. Datasets of various dimensionality and size used for this testing were generated by the same approach as the aforementioned. Both algorithms show roughly linear run times as the number of dimensions, while superlinearly run times as the dataset size. This is evident from Fig. 8 that *ULOF* algorithm is hundreds of times faster than *BULOF*, regardless of the dimensionality and size of uncertain dataset. This provides good support for the efficiency of pruning strategy.

5. Conclusions and summary

In this paper, we presented a new outlier detection algorithm based on local information to detect abnormal points on uncertain data, we call it *ULOF*. As far as we know this is the first time to generalize the definitions of traditional LOF algorithm by including uncertainty information. Local level of uncertainty together with local density information determines the degree of outlier for each point in dataset. We give several effective measures to solve computational problems of evaluating *K–η–distance* and neighbors. Theory analysis and performance experiments have been done to verify the advantages and superiorities of *ULOF* algorithm. Results show that uncertainty level has a major impact on running time and detection accuracy. The following work is to develop more effective methods for abnormal behavior detection on uncertain data with a very complex structure, such as high-dimensional uncertain data. In addition, a more effective means of estimating distance function is important for data mining techniques on uncertain data.

References

[1] P.K. Agarwal, S.W. Cheng, K. Yi, Range searching on uncertain data, ACM Transactions on Algorithms 8 (2012) 43.
[2] C.C. Aggarwal, Managing and Mining Uncertain Data, Advances in Database Systems, vol. 35, Kluwer, 2009.
[3] C.C. Aggarwal, Y. Li, J. Wang, J. Wang, Frequent pattern mining with uncertain data, in: KDD, pp. 29–38.
[4] C.C. Aggarwal, P.S. Yu, A framework for clustering uncertain data streams, in: ICDE, IEEE, 2008, pp. 150–159.
[5] C.C. Aggarwal, P.S. Yu, Outlier detection with uncertain data, in: SDM, SIAM, 2008, pp. 483–493.
[6] C.C. Aggarwal, P.S. Yu, A survey of uncertain data algorithms and applications, IEEE Transactions on Knowledge Data Engineering 21 (2009) 609–623.
[7] D. Bray, M.D. Levin, C.J. Morton-Firth, Predicting temporal fluctuations in an intracellular signalling pathway, Journal of Theoretical Biology 192 (1998) 117–128.
[8] M.M. Breunig, H.P. Kriegel, R.T. Ng, J. Sander, Lof: Identifying density-based local outliers, in: SIGMOD Conference, ACM, 2000, pp. 93–104.
[9] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computational Survey 41 (2009) 15:1–15:58.
[10] Y. Chen, X. Dang, H. Peng, H.L.B. Jr., Outlier detection with the kernelized spatial depth function, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 288–305.
[11] A.P. Dempster, Upper and lower probabilities induced by a multivalued mapping, Annals of Mathematical Statistics 38 (1967) 325–339.
[12] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, T. Kanamori, Statistical outlier detection using direct density ratio estimation, Knowledge Information System 26 (2011) 309–336.
[13] B. Jiang, J. Pei, Outlier detection on uncertain data: objects, instances, and inferences, in: ICDE, IEEE Computer Society, 2011, pp. 422–433.
[14] B. Jiang, J. Pei, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data: model and bounding–pruning–refining methods, Journal of Intelligence and Information System 38 (2012) 1–39.
[15] W. Jin, A.K.H. Tung, J. Han, W.W. 0010, Ranking outliers using symmetric neighborhood relationship, in: PAKDD, Lecture Notes in Computer Science, vol. 3918, Springer, 2006, pp. 577–593.
[16] T. Johnson, I. Kwok, R.T. Ng, Fast computation of 2-dimensional depth contours, in: KDD, pp. 224–228.
[17] B. Kaluza, V. Mirchevska, E. Dovgan, M. Lustrek, M. Gams, An agent-based approach to care in independent living, in: AmI, Lecture Notes in Computer Science, vol. 6439, Springer, 2010, pp. 177–186.
[18] M.E. Khalefa, M.F. Mokbel, J.J. Levandoski, Skyline query processing for uncertain data, in: CIKM, ACM, 2010, pp. 1293–1296.
[19] H.P. Kriegel, P. Kroger, E. Schubert, A. Zimek, Interpreting and unifying outlier scores, in: SDM, SIAM/Omnipress, 2011, pp. 13–24.
[20] H.P. Kriegel, P. Kroger, E. Schubert, A. Zimek, Loop: local outlier probabilities, in: CIKM, ACM, 2009, pp. 1649–1652.
[21] H.P. Kriegel, M. Schubert, A. Zimek, Angle-based outlier detection in high-dimensional data, in: KDD, ACM, 2008, pp. 444–452.
[22] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[23] L.A. Zadeh, Fuzzy sets as a basis for theory of possibility, Fuzzy Sets and Systems 1 (1978) 3–28.
[24] C.K.S. Leung, Mining uncertain data, Wiley Interdiscovery Rewind: Data Mining and Knowledge Discovery 1 (2011) 316–329.
[25] C. Li, W.H. Wong, Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection, Proceedings of the National Academy of Sciences 98 (2001) 31–36.
[26] B. Liu, J. Yin, Y. Xiao, L. Cao, P.S. Yu, Exploiting local data uncertainty to boost global outlier detection, in: ICDM, IEEE Computer Society, 2010, pp. 304–313.

[27] W.K. Ngai, B. Kao, C.K. Chui, R. Cheng, M. Chau, K.Y. Yip, Efficient clustering of uncertain data, in: ICDM, IEEE Computer Society, 2006, pp. 436–445.
[28] S. Papadimitriou, H. Kitagawa, P.B. Gibbons, C. Faloutsos, Loci: fast outlier detection using the local correlation integral, in: ICDE, IEEE Computer Society, 2003, pp. 315–326.
[29] N. Pham, R. Pagh, A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data, in: KDD, ACM, 2012, pp. 877–885.
[30] B. Qin, Y. Xia, F. Li, A Bayesian classifier for uncertain data, in: SAC, ACM, 2010, pp. 1010–1014.
[31] R.K. Bock, P. Savicky, UCI Machine Learning Repository, 2007.
[32] S.A. Shaikh, H. Kitagawa, Distance-based outlier detection on uncertain data of gaussian distribution, in: APWeb, Lecture Notes in Computer Science, vol. 7235, Springer, 2012, pp. 109–121.
[33] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, S. Prabhakar, Indexing multi-dimensional uncertain data with arbitrary probability density functions, in: VLDB, ACM, 2005, pp. 922–933.
[34] B. Wang, G. Xiao, H. Yu, X. Yang, Distance-based outlier detection on uncertain data, in: CIT (1), IEEE Computer Society, 2009, pp. 293–298.
[35] X. Wang, X. Wang, D.M. Wilkes, A fast distance-based outlier detection technique, in: ICDM, IbaI Publishing, 2008, pp. 25–44.
[36] Y. Wang, S. Parthasarathy, S. Tatikonda, Locality sensitive outlier detection: a ranking driven approach, in: ICDE, IEEE Computer Society, 2011, pp. 410–421.
[37] Y. Zhang, N. Meratnia, P.J.M. Havinga, Outlier detection techniques for wireless sensor networks: a survey, IEEE Communications Surveys and Tutorials 12 (2010) 159–170.