On Indexing High Dimensional Data with Uncertainty

Charu C. Aggarwal* Philip S. Yu†

Abstract

In this paper, we will examine the problem of distance function computation and indexing uncertain data in high dimensionality for nearest neighbor and range queries. Because of the inherent noise in uncertain data, traditional distance function measures such as the L_q-metric and their probabilistic variants are not qualitatively effective. This problem is further magnified by the sparsity issue in high dimensionality. In this paper, we examine methods of computing distance functions for high dimensional data which are qualitatively effective and friendly to the use of indexes. In this paper, we show how to construct an effective index structure in order to handle uncertain similarity and range queries in high dimensionality. Typical range queries in high dimensional space use only a subset of the ranges in order to resolve the queries. Furthermore, it is often desirable to run similarity queries with only a subset of the large number of dimensions. Such queries are difficult to resolve with traditional index structures which use the entire set of dimensions. We propose query-processing techniques which use effective search methods on the index in order to compute the final results. We discuss the experimental results on a number of real and synthetic data sets in terms of effectiveness and efficiency. We show that the proposed distance measures are not only more effective than traditional L_q-norms, but can also be computed more efficiently over our proposed index structure.

1 Introduction

In recent years, many advanced technologies have been developed to store and record large quantities of data continuously. In many cases, the data may contain errors or may be only partially complete. For example, sensor networks typically create large amounts of uncertain data sets. In other cases, the data points may correspond to objects which are only vaguely specified, and are therefore considered uncertain in their representation. Similarly, surveys and imputation techniques create data which is uncertain in nature. This has created a need for *uncertain data management* algorithms and applications. A survey of different kinds of uncertain data algorithms may be found in [4].

In uncertain data management, data records are represented by probability distributions rather than deterministic values. Therefore, a data record is represented by the corresponding parameters of a multidimensional probability distribution. Some examples in which uncertain data management techniques are relevant are as follows:

- The uncertainty may be a result of the limitations of the underlying equipment. For example, the output of sensor networks is often uncertain. This is because of the noise in sensor inputs or errors in wireless transmission.

- In many cases such as demographic data sets, only partially aggregated data sets are available. Thus, each aggregated record is actually a probability distribution.

- In privacy-preserving data mining applications, the data is perturbed in order to preserve the sensitivity of attribute values. In some cases, probability density functions of the records may be available [3].

- In some cases, data attributes are constructed using statistical methods such as forecasting or imputation. In such cases, the underlying uncertainty in the derived data can be estimated accurately from the underlying methodology.

The problems of distance function computation and indexing are closely related, since the construction of the index can be sensitive to the distance function. Furthermore, effective distance function computation is inherently more difficult in the high dimensional or uncertain case. Direct extensions of distance functions such as the L_q-metric are not very well suited to the case of high dimensional or uncertain data management. This is because these distances are most affected by the dimensions which are most dissimilar. In the high dimensional case, the statistical behavior of the sum of these dissimilar dimensions leads to the *sparsity problem*. This results in similar distances between every pair of points,

*IBM T. J. Watson Research Center, charu@us.ibm.com
†University at Illinois at Chicago, psyu@cs.uic.edu

and the distance functions are often *qualitatively ineffective* [12]. Furthermore, the dimensions which contribute most to the distance between a pair of records are also likely to have the greatest uncertainty. Therefore, the effects of high dimensionality are magnified by the uncertainty, and the contrast in distance function computations is lost. The challenge is to design a distance function which continues to be both *qualitatively effective* and *index-friendly*.

The problem of indexing has been studied extensively in the literature both for the case of deterministic data [6, 7], and for the case of uncertain data [8, 9, 15, 16]. However these techniques do not deal with some of the unique challenges in the similarity indexing of high dimensional or uncertain data. These unique challenges are as follows:

- Similarity functions need to be carefully designed for the high dimensional and uncertain case in order to maintain contrast in similarity calculations. Furthermore, the distance function needs to be sensitive to the use of an index.

- In most cases, the similarity or range queries are only performed on a small subset of the dimensions of high dimensional data. For example, in many applications, we are likely to perform a range query only over 3 to 4 dimensions of a 100-dimensional data set. Such queries cannot be processed with the use of traditional index structures, which are designed for full-dimensional queries.

The queries which can be resolved with the use of our index structure are as follows:

- Determine the nearest neighbor to a given target record in conjunction with an effective distance function.

- Determine the nearest neighbor to the target T by counting the expected number of dimensions for which the points lie within *user-specified* threshold distances $t_1 \ldots t_d$. In practical applications, only a small number of the threshold values $t_1 \ldots t_d$ may be specified, and the remaining are assumed to be ∞. In such cases, the nearest neighbor search is performed only over a small number of projected dimensions which are relevant to that application.

- For a given subset of dimensions S, and a set of ranges $R(S)$ defined on the set S, determine the points which lie in $R(S)$ with probability greater than δ. We note that this particular query is referred to as a *projected range query*, since we are using only a subset of the dimensions.

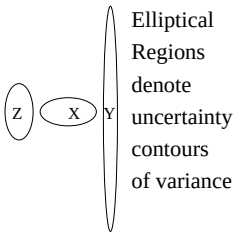

Figure 1: Effects of uncertainty on distance

We will see that the key is to construct a distance function which can be computed efficiently in the high dimensional case, and is both qualitatively effective and index-friendly. We will refer to this index structure as UniGrid (or UNcertain Inverted GRID Structure).

This paper is organized as follows. In section 2, we will examine the issue of uncertain distance function construction. In section 3, we will introduce the UniGrid index structure for this distance function. In section 4, we will discuss query processing techniques with the UniGrid index structure. The experimental results are discussed in section 5. Section 6 contains the conclusions and summary.

2 Uncertain Distance Function Construction

In this section, we will study the problem of distance function computation for uncertain data. We will show how the uncertainty in the data can influence the behavior of the underlying distance function. Then, we will discuss different ways of constructing an uncertain function for distance computation, and their effectiveness in the high dimensional case. We will also discuss a dimensionality-dependent approach for optimizing the design of the distance function. We will first introduce some notations and definitions.

We assume that the uncertain data base $\mathcal{D}$ contains a set of N records, each with a dimensionality of d. The records in $\mathcal{D}$ are denoted by $\overline{X_1} \ldots \overline{X_N}$. The individual components of $\overline{X_i}$ are denoted by $\left[(x_i^1, f_i^1(\cdot)) \ldots (x_i^d, f_i^d(\cdot))\right]$. Here x_i^j denotes the value for the ith record on the jth dimension, and $f_i^j(\cdot)$ denotes the probability density function (pdf) for the ith record on the jth dimension. We assume that the pdfs across the different dimensions are independent of one another. We note that x_i^j is the mean of the pdf $f_i^j(\cdot)$, and therefore the value x_i^j can be omitted entirely without loss of completeness in record description. Nevertheless, we will preserve it for ease in notation.

A straightforward design of a distance function would simply generalize the standard L_k metric to the case of uncertain data sets without using the uncertainty information. Thus, the raw values of x_i^j can be used

for computation purposes. However, such a definition may over-estimate or under-estimate the distance values when there is *skew across the uncertainty behavior of different attributes*. By "skew" we refer to the fact that the relative level of uncertainty across different attributes may be very different. An example is illustrated in Figure 1. Here, we have illustrated a two dimensional example, in which each of the three data points has different level of uncertainty along different dimensions. In each case, we have illustrated the variance behavior of the corresponding data point with the use of an elliptical contour. At first sight, it might seem that data point X is closer to Y than Z if the mean position of each point is used. However, in reality, the data point X is much closer to Z in expectation. This is because the *expected distance* of X to Z is lower than the expected distance of X to Y. Thus, the different level of skew in the uncertainty of different attributes affects the final distance computation.

A natural alternative is to use the expected distance between two data points. We denote the random variable for the distance between data points $\overline{X_i}$ and $\overline{X_j}$ along the kth dimension by $d_k(\overline{X_i}, \overline{X_j})$. The expected value of this random variable is denoted by $E[d_k(\overline{X_i}, \overline{X_j})]$. The expected distance between points $\overline{X_i}$ and $\overline{X_j}$ along the dimension k is denoted by $E[||x_i^k - x_j^k||]$. This distance can be simplified to the following expression:

$$E[||x_i^k - x_j^k||] = \int_{x,y} ||x - y|| f_i^k(x) f_j^k(y) dx.dy \tag{2.1}$$

We note that the above expression is designed for the case of the Manhattan metric. It is possible to design similar metrics for the case of the general L_p-metric. The random variable for the total distance $d(\overline{X_i}, \overline{X_j})$ is defined by the following relationship:

$$E[d(\overline{X_i}, \overline{X_j})] = \sum_{k=1}^{d} E[d_k(\overline{X_i}, \overline{X_j})] \tag{2.2}$$

We note that the use of expected distances can sometimes result in noisy distance functions when the error is large. This is because metrics such as the L_p-metric are dominated by the large terms which are created by errors along individual dimensions. As a result, the similarity function may lose its effectiveness with increasing uncertainty. A more effective distance function is one which only counts dimensions that are probabilistically close to the dimensions above a certain threshold. These functions are actually *similarity* functions rather than distance functions, since larger values imply greater similarity. Furthermore, such functions can be specifically tailored to *contrast conditions* [12], which guarantee effectiveness with increasing dimensionality. We define the probabilistic proximity functions $G(\overline{X}, \overline{Y}, s^1 \ldots s^d)$ for thresholds $s^1 \ldots s^d$ as follows:

DEFINITION 1. *The probabilistic function $G(\overline{X}, \overline{Y}, s^1 \ldots s^d)$ is defined as the expected number of dimensions for which the distance between the kth attribute values in $\overline{X}$ and $\overline{Y}$ is less than s^k. The value of s^k is chosen in an automated way by analyzing the local behavior of the underlying data along the kth dimension.*

Let us define the probability that the distance of $\overline{X}$ and $\overline{Y}$ along the dimension k is less than s^k by $h_k(\overline{X}, \overline{Y}, s^k)$. This probability is defined by the following relationship:

$$h_k(x_i^k, x_j^k, s^k) = \int_{||x-y|| \leq s^k} f_i(x) \cdot f_i(y) dx.dy \tag{2.3}$$

Then, the expected number of dimensions $G(\overline{X_i}, \overline{X_j}, s^1 \ldots s^d)$ on which $\overline{X_i}$ and $\overline{X_j}$ are closer to one another than the threshold s^k is given by:

$$G(\overline{X_i}, \overline{X_j}, s^1 \ldots s^d) = \sum_{k=1}^{d} h_k(x_i^k, x_j^k, s^k) \tag{2.4}$$

We note that the computation of $f_k(\cdot)$ is typically more locality sensitive, since we need to consider only values of the attribute within immediate locality of the target. This is useful from an indexing point of view, since we are typically trying to find records within the locality of a particular target or a pre-specified range.

In some practical applications, it may be desirable to let the user pick only a subset of dimensions over which the similarity is computed. In the most general case, even the threshold for each dimension may vary. For example, the thresholds over the d different dimensions may be set to $t_1 \ldots t_d$. We note that the only difference between this query and the previous query is that the thresholds $t_1 \ldots t_d$ are chosen by the user, whereas the thresholds $s^1 \ldots s^d$ are chosen in an automated way. Some of the values of t_i may be set to ∞, which results in (differential) counting over only a relevant subset of dimensions. If desirable, some of the values of t_i may be set of ∞, whereas the other values corresponding to s^i may be chosen in an automated way as discussed subsequently. This results in a classical projected similarity query in which only a small subset of the dimensions is used for counting. This is a particularly difficult query to handle with the use of typical index structures. However, we will show that the UniGrid structure is very efficient in resolving these kinds of queries. As in the previous case, this kind of query has a dimension-specific locality which is inherently index-friendly. Next, we will discuss how the thresholds $s^1 \ldots s^d$ are chosen in a dimension-specific way.

2.1 Automated Choice of Locality Thresholds

In this section, we will discuss the process of choosing the locality thresholds in an automated way. We note that the issue of locality thresholds is more relevant to the high dimensionality issue, than it is to the uncertainty issue. Therefore, in order to simplify the analysis for choosing the threshold, we will use only the means of the records without using the probability density function. Our approach will be to set the value of $s^1 \ldots s^k$ in such a way that the same fraction f of the means of the other records lie within the threshold of target in each dimension. This ensures that the fraction f is chosen in a locality specific way. However, we need to choose f appropriately with respect to some dimensionality specific parameters.

One of the key properties of high dimensional data [12] is that the distance to the farthest and nearest neighbor is quite similar, as a result of which the *contrast* of the similarity function is lost. This contrast problem is further magnified by the inherent noise present in uncertain data. The key is to choose the locality thresholds in such a way that a high dimensional contrast condition discussed in [12] is satisfied. We will see that this contrast condition is affected by the choice of f. In particular, smaller values of f lead to greater contrast. The contrast condition is defined as follows. Let $\mu(f)$ and $\sigma(f)$ be the mean and standard deviations of the distance calculations to different data points from a target record, when a fraction f of the records are used for constructing the thresholds. Then, the contrast ratio [12] is given by $\sigma(f)/\mu(f)$. The larger this value, the greater the contrast of the distance function in terms of relative distances to different data points.

There is an inherent tradeoff in proper choice of the parameter f. In general, if f is chosen to be too small, then only a small number of dimensions may intersect with the user specified range, and this may result in loss of information. On the other hand, if f is chosen to be too large, then the data loses its contrast for high dimensional data. In general, we would like to choose the largest value of f, so that the contrast is not lost for high dimensional data. Since the most difficult case for indexing is that of uniformly distributed data, we will perform an analysis for this case. We assume that the means of the different records are uniformly distributed throughout the data.

We will analyze the distance behavior of the target record to a randomly chosen record with mean denoted by $\overline{Z}$. Let Q_i be a binary random variable on dimension i which takes on the value of 1, if the mean for a given record lies within the distance s^i from the target along a particular dimension. We note that Q_i is a Bernoulli random variable which has mean f. We define the number of dimensions along which the two records are within the threshold s^i by $M = \sum_{i=1}^{d} Q_i$. Note that the random variable M has mean $\mu(f) = d \cdot f$ and variance which is given by $\sigma^2(f) = d \cdot f \cdot (1-f)$. We note that the condition for contrasts on the distance functions is given by:

$$\lim_{d \Rightarrow \infty} \sigma(f)/\mu(f) > 0 \tag{2.5}$$

This condition translates to the following:

$$\lim_{d \Rightarrow \infty} (1-f)/(f \cdot d) > 0 \tag{2.6}$$

This means that f reduces with increasing dimensionality and the value of f should be proportional to at most $1/d$. We assume that, for each dimension, we use a threshold which is given by some value $f = C/d$, where C is a constant which is at most equal to 1. Furthermore, C should be chosen such that $1/f$ is an integer. As we will see, this is useful for creating the UniGrid index structure. In the next section, we will discuss the structure of this index.

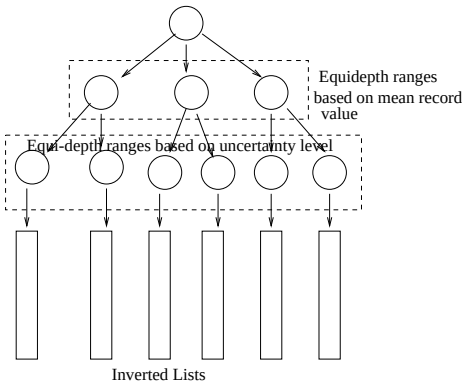

Figure 2: The two level UniGrid index structure

3 The UniGrid Index

In this section, we will discuss the UniGrid structure for uncertain data indexing. We will show how to use the structure for the purpose of range queries. We will also see that this structure is suited to processing lower dimensional projectional queries of high dimensional data.

In the UniGrid structure, we construct a two-level inverted partitioning of the data. In this technique, we create an inverted list of record identifiers separately for each attribute in the data. We create an inverted list of record identifiers for all points whose mean value and uncertainty lie within certain pre-specified ranges. The first level of the partitioning uses the mean value of the probability density function for that record, and the

second level further partitions the set of records with mean value in a given range by using the uncertainty in the range for the corresponding probability density function.

In order to construct the first level of the inverted partitioning, we divide each attribute into a set of equi-depth ranges. The boundaries for the equi-depth ranges for attribute i are contained in $[l(i,1), u(i,1)], [l(i,2), u(i,2)] \ldots [l(i,q), u(i,q)]$, and we assume that $u(i,k) = l(i,k+1)$ for each $k \in \{1 \ldots q-1\}$. The value of q is typically chosen to be a multiple of $1/f$. Since f is chosen such that the value of $1/f$ is chosen so as to be an integer, it follows that q is an integer as well.

Here $[l(i,r), u(i,r)]$ represents the rth range for the attribute i. All record identifiers whose ith attribute lies in the range $[l(i,r), u(i,r)]$ can be found in one of a group of inverted lists for this range. A second level of the partitioning is defined by the uncertainty level. This second level uniquely defines the inverted list for a given record identifier from within the group of lists belonging to the range $[l(i,r), u(i,r)]$. There are approximately $\lfloor N/q \rfloor$ record identifiers in each range of the first level of the partitioning. For the second level of the partitioning, we construct ranges for the uncertainty by using the behavior of the probability density function of the uncertainty. We make the assumption that the probability distribution functions are defined over a finite range. This is without loss of generality, since the insignificant tails for the uncertainty function can be removed. For example, for a gaussian uncertainty function, any tail beyond 3 standard deviations from the mean can be ignored to a level of accuracy beyond 99.99%. The *span* of the uncertainty function is defined by half the distance between the left and right boundaries of the uncertainty function. The data points within the range $[l(i,r), u(i,r)]$ are divided into a set of s equi-depth ranges depending upon the corresponding span. It is assumed that the absolute upper and lower bounds for the range corresponding to the tth span are denoted by $[ls(i,r,t), us(i,r,t)]$. Thus, the length of the tth span is $(us(i,r,t) - ls(i,r,t))/2$. Each span points to an inverted list which contains approximately $\lfloor N/qs \rfloor$ record identifiers. The two level partitioning of the data into inverted lists along a particular attribute is illustrated in Figure 2. Each entry of the inverted list corresponding to the record $\overline{X_m}$ contains the following information:

- The mean value x_m^i of the corresponding record $\overline{X_m}$ and attribute i.
- The probability density function for the record $\overline{X_m}$ and the attribute i, which is denoted by $f_m^i(\cdot)$.

In addition, we store some meta-information along both levels of the hierarchy of the inverted lists. This meta-information is as follows:

- For each node at the first level of the hierarchy, we store the upper and lower bound of the corresponding discretized attribute range. These upper and lower bounds are denoted by $u(i,r)$ and $l(i,r)$ respectively.
- For each node at the second level of the hierarchy, we store the upper and lower bounds for the uncertainty span which are denoted by $us(i,r,t)$ and $ls(i,r,t)$ respectively.

This meta-information is chosen in order to facilitate the selection of certain inverted lists which are used in processing similarity and range queries. We refer to this structure as the UniGrid structure for query processing.

At this point, we would like to point out the storage requirement of the UniGrid structure with respect to the original data set. We assume that the key storage bottleneck is the probability density function for each attribute in the uncertain data set. Thus, if the storage of the probability density function requires M times the storage of a single value, then (for a data set with N records and d dimensions) the total storage requirement of the probability density functions *in the original data set* is given by $N \cdot M \cdot d$. In addition, since each attribute value is represented exactly once, the total storage requirement of the original data is given by $N \cdot (M+1) \cdot d$.

Next, we will examine the storage requirement of the UniGrid structure. For the case of the Unigrid structure, each probability density function is represented exactly once in the form of its position in exactly one inverted list. In addition, we need to store the record identifiers and the values of the corresponding attributes. This adds up to a storage requirement of $N \cdot (M+2) \cdot d$. Since the relative space required for the storage of a probability density function is significantly larger than that required for the storage of a single value, it follows that the total storage requirement of both the original data and the inverted structure is given by $O(N \cdot d \cdot M)$. Thus, the overhead of the UniGrid structure is not significant, when compared to the original data set.

4 Query Processing with UniGrid

In this section, we will discuss methods for utilizing the UniGrid structure for query processing. First, we will discuss the problem of similarity queries. Then, we will discuss how to handle projected range queries with this technique.

Algorithm SimilarityQuery(Database: $\mathcal{D}$, Target Point: $(\overline{Y}, h(\cdot))$, Thresholds: $t_1 \ldots t_d$);
{ $(z^1 \ldots z^d)$ represent midpoints of corresponding probability density function spans.}
begin
Determine the span for the uncertainty function $h(\cdot)$ along the different dimensions and denote by $\theta^1 \ldots \theta^d$;
Determine all the inverted lists for which the range $[z^k - \theta^k - t_k, z^k + \theta^k + t_k]$ intersects with the corresponding inverted list range $[ls(k, r, t), us(k, r, t)]$;
Compute similarity values for the different record identifiers in these inverted lists by aggregating the computation of Equation 2.3 over different dimensions;
Report largest possible similarity value and record identifier;
end

Figure 3: Similarity Search Processing

4.1 Similarity Queries For the case of similarity queries, we assume that we have a target record $\overline{Y}$, along with the uncertainty function $h(\cdot)$. We will consider the case of similarity queries in which we are computing the expected number of dimensions which lie within the thresholds $t_1 \ldots t_d$. We note that this query is exactly similar to the case when we are using automated thresholds $s^1 \ldots s^d$. The d dimensions of $\overline{Y}$ are denoted by $(y^1 \ldots y^d)$. In many applications, the target record may be deterministic. In such cases, we can assume that the uncertainty function is deterministic with zero span. We also assume that the midpoints of the corresponding probability density functions (between the left and right boundary) are denoted by $(z^1 \ldots z^d)$. We note that the midpoint may be different from the corresponding mean.

The first step in resolving the similarity query is to determine the span of the target record. Let the span along the kth dimension be denoted by θ^k. Therefore, the probability density function for the kth dimension in the target record $\overline{Y}$ is encapsulated by $z^k - \theta^k$ and $z^k + \theta^k$. We would like to determine the probability $h(y^k, x_i^k, t_k)$ that the kth dimension for the record $\overline{Y}$ lies within the distance t_k of the kth dimension for record $\overline{X_i}$. We note that in order for this probability to be non-zero, the probability density function of the record $\overline{X_i}$ must intersect with the range $[z^k - \theta^k - t_k, z^k + \theta^k + t_k]$.

Since the ranges in the inverted lists are organized by span, it is possible to truncate those lists for which the span does not intersect with the range $[z^k - \theta^k - t_k, z^k + \theta^k + t_k]$. Then, for all values of r and t, we truncate those lists for which the range $[ls(k, r, t), us(k, r, t)]$ does not intersect with the range $[z^k - \theta^k - t_k, z^k + \theta^k + t_k]$. We note that this intersection must occur for the corresponding dimension to be relevant to the similarity computation of a particular record with respect to the target $\overline{Y}$. If this is not the case, then the computation of Equation 2.3 will return a null value.

For the inverted lists which do intersect with the range $[z^k - \theta^k - t_k, z^k + \theta^k + t_k]$, we further determine the records in these inverted lists which do intersect with the range $[z^k - \theta^k - t_k, z^k + \theta^k + t_k]$. For these records, we compute the corresponding probability of intersection by using Equation 2.3. We keep track of this value over the different records which lie in these inverted lists. At the end, we report the largest similarity value. We note that in order to keep track of the similarity values over the different records, we need to maintain a hash table of record identifiers. Whenever the similarity contribution of a record identifier from an inverted list is computed with the use of Equation 2.3, it is added to the corresponding entry in the hash table. The overall process of performing the similarity computation is illustrated in Figure 3. We note that when only a small subset of dimensions are specified, we can completely omit counting along any dimension k for which $t_k = \infty$. This is because there is no *differential* counting along those dimensions. This can improve the efficiency of the counting process.

4.2 Range Queries The UniGrid structure can also be used for the case of range queries in a fairly straightforward way. In this case, we are using only a small subset of the dimensions for range queries. For the high dimensional case, only a small subset of the dimensions is typically used in practical applications, since a user is unlikely to query on a large number of dimensions simultaneously. Such queries are extremely difficult to resolve with the use of traditional indices. The subset of dimensions for the range query is denoted by S, and the corresponding set of ranges are denoted by $R(S) = [a_1 b_1] \ldots [a_v b_v]$. Without loss of generality, we can assume that the first v dimensions are picked for querying, since the same argument applies after reordering the dimensions. We would like to determine the set of points which lie in $R(S)$ with probability at least δ.

In order to resolve such a query, we first determine the set of inverted grid lists which intersect with these ranges. In order for the range along the kth dimension to intersect with the inverted list $[ls(k, r, t), us(k, r, t)]$, the two ranges $[a_k, b_k]$ and $[ls(k, r, t), us(k, r, t)]$ must intersect. Once the intersecting ranges have been determined, we process the inverted lists in order of dimension. First, we determine all record identifiers which have non-zero probability of intersection with $[a_1, b_1]$ along the first dimension, and record their

Algorithm RangeQuery(Database: $\mathcal{D}$, Dimensions: $\{1 \ldots v\}$, Ranges: $[a_1, b_1] \ldots [a_v, b_v]$ Threshold: δ);
begin
Determine all inverted lists along the first dimension for which $[ls(1, r, t), us(1, r, t)]$ intersects with the range $[a_1, b_1]$;
Compute probability of intersection of corresponding data points in inverted lists with range $[a_1, b_1]$ using the uncertainty function of corresponding data points and denote list by L_1;
Remove data points in list L_1 with probability less than δ;
for $dim = 2$ to v **do**
begin
Determine all inverted lists along the first dimension for which $[ls(dim, r, t), us(dim, r, t)]$ intersects with the range $[a_{dim}, b_{dim}]$;
Compute probability of intersection of corresponding data points in inverted lists with range $[a_{dim}, b_{dim}]$ using the uncertainty function of corresponding data points and denote list by L_2;
Multiply the probabilities of list L_1 with the probabilities in list L_2;
{ It is assumed that the absent elements have probability of zero }
Remove data points in list L_1 with probability less than δ;
end
return(L_1);
end

Figure 4: Range Query Processing

probability of intersection. This is done by examining all record identifiers which lie in the inverted lists which intersect with $[a_1, b_1]$ along dimension 1. We do this for all intersecting lists along dimension 1, and take the union of the corresponding lists. We prune all those dimensions for which the threshold is less than δ. We repeat the same process for the second dimension and the corresponding range $[a_2, b_2]$. We retain only those record identifiers which are common between the two lists. For these record identifiers, we multiply the probability values in the two lists in order to compute the probability that the corresponding record identifier lies within the pre-specified range along both dimensions. As in the previous case, we remove those identifiers for which the probability value is less than δ. We repeat the process iteratively for each dimension, until all the ranges have been processed. The overall process is illustrated in Figure 4.

This approach to projected range queries has some advantages in terms of access costs. We need to process only those dimensions which are selected by the user. For example, if v dimensions are selected in the projected range query, and a fraction α of the ranges along each dimension are selected, then the fraction of the inverted index processed is given by $\alpha \cdot v/d$. Consider the case when we are performing a projected range query on a subset of 2 dimensions in 100 dimensional data. Let us also assume that the average range length along each dimension includes a fraction 0.1 of the inverted lists along each dimension. Then, the above relationship implies that we are processing only a fraction 0.2% of the inverted index. Thus, for a small number of projected dimensions, only a small fraction of the inverted index needs to be accessed.

5 Experimental Results

In this section, we will explore the effectiveness and efficiency of the proposed indexing technique. We note that most indexing structures are designed to handle full dimensional indexing queries, and cannot be used for the kind of indexing techniques studied in this paper. Furthermore, we have designed specialized distance functions for the case of high dimensional uncertain data. Such distance functions cannot be effectively indexed with the use of index structures such as those discussed in [8, 9]. Therefore, a direct comparison between the methods is not meaningful. Nevertheless, our approach can be tested using a variety of independent measures:

- Effectiveness of our designed distance function with increasing uncertainty. We will specifically compare our method with respect to traditional ap-

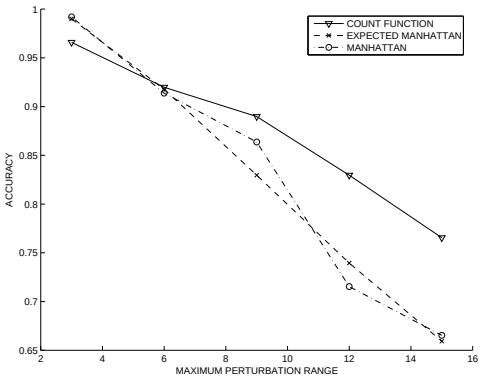

Figure 5: Effectiveness of different distance functions on $Network$.P(u)

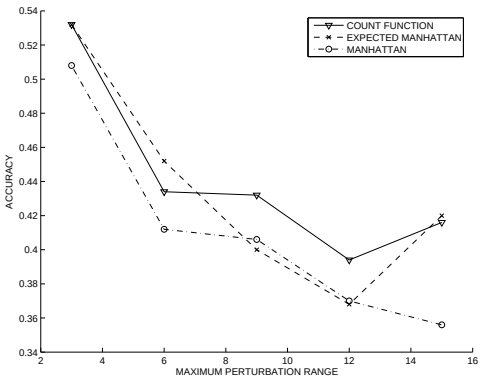

Figure 6: Effectiveness of different distance functions on $Forest$.P(u)

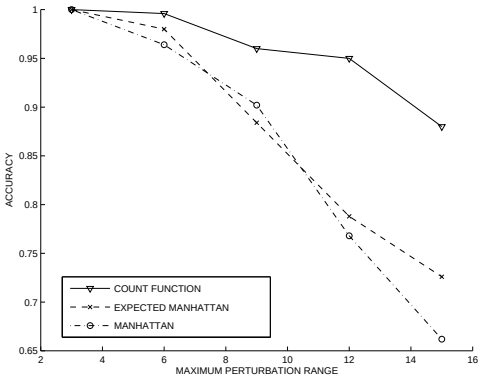

Figure 7: Effectiveness of different distance functions on Syn20.D20K.P(u)

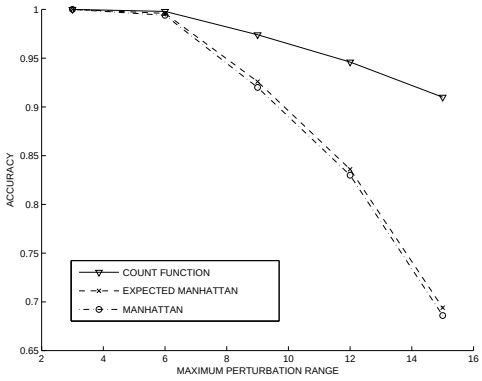

Figure 8: Effectiveness of different distance functions on Syn25.D20K.P(u)

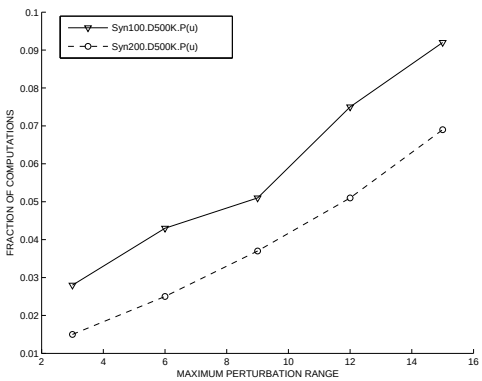

Figure 9: Efficiency of similarity search with increasing uncertainty

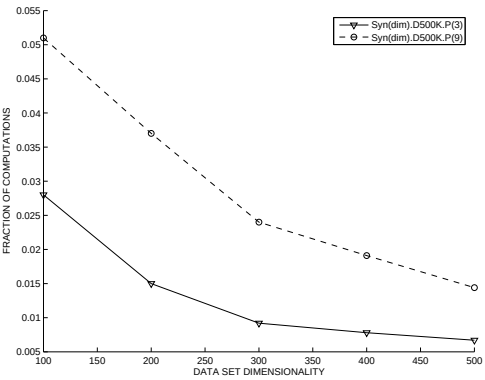

Figure 10: Efficiency of similarity search with increasing dimensionality

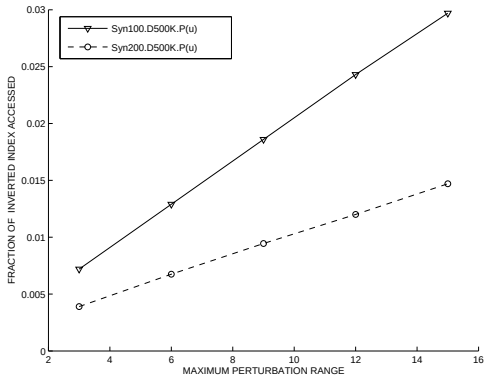

Figure 11: Efficiency of projected range query with increasing uncertainty

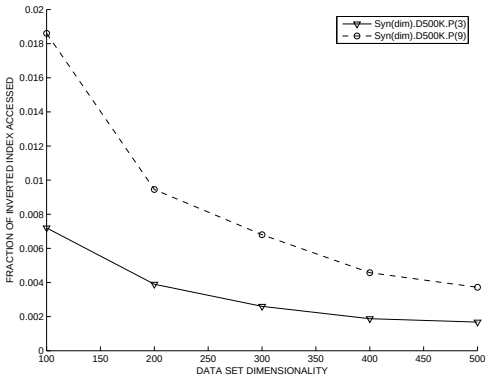

Figure 12: Efficiency of projected range query with increasing data dimensionality

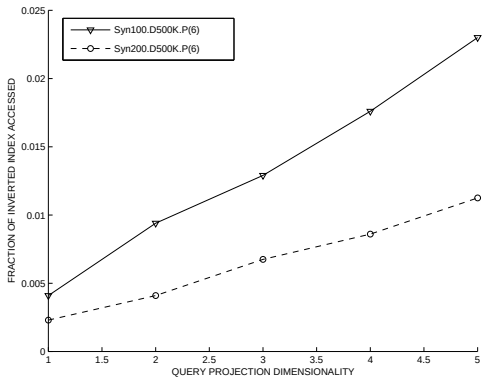

Figure 13: Efficiency of projected range query with increasing projection dimensionality

proaches such as the L_p-norm.

- Efficiency of the proposed similarity search technique with increasing uncertainty and dimensionality.
- Efficiency of projected range queries with increasing uncertainty, projection dimensionality, and full dimensionality.

For the purpose of comparison, we used data sets from the UCI machine learning repository. We introduced additional uncertainty into the data sets by adding noise which was drawn from a uniform probability distribution. For dimension j of record i, the uniform distribution had a range γ_{ij} which was drawn from $[0, u]$. The noise added to the jth dimension of record i was drawn from the uniform distribution in the range $[-\gamma_{ij}/2, \gamma_{ij}/2]$. By varying the value of u, it is possible to increase or decrease the level of uncertainty in each record. We also store the pdf of the record in the form of the range of the corresponding uniform distribution. We note that we do not know the original value of the record, and therefore, we do not know where the pdf is centered. Therefore, we assume that the pdf for each record is centered at its perturbed value rather than its original value. The real data set with name $< RName >$ and perturbation level u was denoted by $< RName >$.P(u).

We also generated a number of synthetic data sets by constructing gaussian clusters with means which are randomly drawn from the unit cube. A total of N points were generated in d dimensions. The number of data points in each cube were drawn in proportion to a uniform distribution in $[0, 1]$. We generated $q = 4$ clusters, and the distance of a point in the cluster from the mean along each dimension was drawn from a normal distribution with zero mean and unit standard deviation. Each point was labeled with the identity of the cluster that it belonged to. We note that unlike the case of the real data sets, we know the exact relationship between the data features and the cluster label. A data set with d dimensions, N records was denoted by Syn(d)D(N). For example, two of our generated data sets with $20,000$ data points and 20 and 25 dimensions are denoted by Syn20.D20K and Syn25.D20K respectively. We used the same approach as the real data sets in order to add uncertainty to the base data. Thus, the corresponding data sets with perturbation level u were denoted by Syn20.D20K.P(u) and Syn25.D20K.P(u) respectively

We measured the effectiveness of our distance function with increasing uncertainty level. In order to measure the effectiveness of our distance function, we measured the accuracy of a nearest neighbor classifier with

increasing level of uncertainty. This uncertainty is controlled with the use of the parameter u discussed earlier. We note that for the case of real data sets, we are using the class label as a proxy for the effectiveness of the distance function. While the exact relationship of the feature variables to the class label is not known, the accuracy of the distance function for a classification application provides strong evidence for the effectiveness of the approach. In the case of synthetic data sets, we can control the intensity of the relationship between the feature variables and class label (cluster id) effectively by increasing the uncertainty level in the data. Therefore, in this case we can measure the effectiveness of the distance function by varying the level of relationship between the feature variables and class variable.

In Figure 5, we have illustrated the effectiveness of the nearest neighbor classifier on the Network Intrusion data set, which we denote by $Network$.P(u). On the X-axis, we varied the uncertainty level u, and on the Y-axis, we have illustrated the accuracy of a nearest neighbor classifier. It is clear that the accuracy of the count function was higher than the accuracy of either the Manhattan or the expected Manhattan distance function in most cases. The expected count function was less effective only for the case of very low uncertainty levels. For higher uncertainty levels, the expected count function was superior, and the difference between the expected count function and the other functions increased with uncertainty level. One interesting observation is that the relative accuracy of the raw Manhattan function and the expected Manhattan function varies considerably. However, the expected count function is able to perform relatively robustly in all cases.

In Figure 6, we have illustrated the effectiveness of the nearest neighbor classifier on the Forest Cover data set, which we denote by $Forest$.P(u). As in the previous cases, the expected count function performs better than the other functions in most cases. For lower uncertainty levels, the expected count function performs either evenly or slightly worse than the other functions. However, with increasing uncertainty, the expected count function significantly outperforms the other two functions. As in the previous case, there is very little difference between the Manhattan and the expected Manhattan functions, and the relative quality of the two functions vary over different uncertainty levels. A further observation about this data set is that there is a lot of noise in the behavior of the distance functions. This is because the relationship of the feature variables to the class variable is not perfectly captured by distance functions. Nevertheless, the overall trend provides an idea of the behavior of the distance functions. In order to get a better idea of how locality based functions are affected by the uncertainty, we will examine the behavior of the synthetic data sets.

In Figures 7 and 8, we have illustrated the effectiveness of the approach on the 20- and 25-dimensional synthetic data sets with increasing uncertainty level. We note that the synthetic data set was intentionally designed in such a way that the classification accuracy was very high for low uncertainty levels, and then gradually reduced with addition of noise. Furthermore, this data set had a high level of correlation between locality and classification accuracy, since the labels were defined by cluster behavior. Therefore, the effectiveness of the approach on this data set is a direct indicator of the effectiveness of the different distance functions with increasing uncertainty. In all cases, the expected count function was significantly superior to the other two functions. The difference between the expected count function and the other functions increased with uncertainty level. We also note that there is much less noise in the data set because the class labels (cluster ids) are well correlated with the distance behavior in the base data set, which was used to create the uncertain representations of the data.

In Figure 9, we have illustrated the efficiency of the similarity search technique with increasing uncertainty in the underlying data. On the X-axis, we have illustrated the perturbation range u, and on the Y-axis, we have illustrated the fraction of distance computations that would be otherwise performed with the use of a sequential scan. We have illustrated the results for the two synthetic data sets corresponding to Syn100.D500K.P(u) and Syn200.D500K.P(u). We note that the fraction of distance computations increased with uncertainty level, since a larger fraction of the records were relevant to the search process. We note that these data sets are extremely high dimensional, and even deterministic data sets cannot be effectively indexed in such cases. However, we are able to construct an efficient index, since our choice of distance function is not only index-friendly, but is also more effective than the standard L_1 function in the case of uncertain high dimensional data. One interesting observation is that the efficiency of the technique was better for the higher dimensional data sets. This is because the similarity thresholds for our counting function are defined in a dimensionality-dependent way. With increasing dimensionality, the automated thresholds are smaller. Therefore, a smaller fraction of record identifiers are relevant from the inverted index. We have illustrated the trend with increasing data dimensionality for two different levels of uncertainty in Figure 10. In each case, the (fractional) efficiency of the scheme improves with increasing dimensionality. This advan-

tage is obtained by designing a distance function which works effectively for high dimensional data.

In Figure 11, we have illustrated the efficiency of projected range queries with increasing uncertainty level. In each case, we chose a range along each axis, which was between 10% to 20% of the entire range, and was randomly picked along the corresponding axis. On the X-axis, we have illustrated the uncertainty level, and on the Y-axis, we have illustrated the fraction of the UniGrid index which was accessed. We have illustrated the results for 3-dimensional range queries in 100- and 200-dimensional data sets respectively in Figure 11. As in the case of similarity queries, a greater fraction of the UniGrid index needs to be accessed with increasing uncertainty. We note that the efficiency of the range query is better for the higher dimensional range query. We have illustrated this trend explicitly in Figure 12. In this case, we tested the results for two data sets with uncertainty level $u = 3$ and $u = 9$ respectively. The corresponding data sets are denoted by Syn(dim).D500K.P(3) and Syn(dim).D500K.P(9) respectively. On the X-axis, we have illustrated the data dimensionality, and on the Y-axis, we have illustrated the fraction of the UniGrid index accessed. It is clear that the fraction of the index accessed is approximately inversely proportional to the data dimensionality. This is because approximately the same number of lists are accessed, but the base size of the data increases linearly with dimensionality. We have also illustrated the efficiency of the range query with increasing projection dimensionality. We have illustrated the results for 100- and 200-dimensional data sets with $u = 6$. The corresponding data sets are denoted by Syn100.D500K.P(6) and Syn(200).D500K.P6 respectively. On the X-axis, we have illustrated the data dimensionality and on the Y-axis, we have illustrated the fraction of the UniGrid index accessed. It is clear that the fraction of data accessed is linearly proportional to increasing projection dimensionality. Therefore, the approach scales well with increasing projection dimensionality.

6 Conclusions and Summary

In this paper, we presented a method for distance function computation and indexing of high dimensional uncertain data. We designed an effective method for performing the distance function computations in high dimensionality, so that the contrast in the distances is not lost. We explored the unique issues which arise in the context of performing range or similarity searches in a subset of the dimensions. Such queries cannot be easily resolved with the use of traditional index structures. In order to effectively handle these issues, we designed the UniGrid Index which uses a two level inverted representation for querying purposes. We tested the effectiveness of the method on a number of real data sets, and show that it continues to be efficient and effective with increasing dimensionality.

References

[1] C. C. Aggarwal, and P. S. Yu, *The IGrid Index: Reversing the Dimensionality Curse in High Dimensional Space*, ACM KDD Conference, (2000).

[2] C. C. Aggarwal, *On Density Based Transforms for Uncertain Data Mining*, IEEE ICDE Conference, (2007).

[3] C. C. Aggarwal, *On Unifying Privacy and Uncertain Data Models*, IEEE ICDE Conference, (2008).

[4] C. C. Aggarwal, and P. S. Yu, *A Survey of Uncertain Data Algorithms and Applications*, IBM Research Report RC24394, (2007).

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

[6] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, *The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles*, ACM SIGMOD Conference, (1994).

[7] S. Berchtold, D. Keim, and H.-P. Kriegel, *The X-Tree: An Index Structure for High Dimensional Data*, VLDB Conference, (1996).

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

[9] R. Cheng, D. Kalashnikov, and S. Prabhaker, *Evaluating Probabilistic Queries over Imprecise Data*, ACM SIGMOD Conference, (2003).

[10] N. Dalvi, and D. Suciu, *Efficient Query Evaluation on Probabilistic Databases*, VLDB Conference, (2004).

[11] A. Das Sarma, O. Benjelloun, A. Halevy, and J. Widom, *Working Models for Uncertain Data*, IEEE ICDE Conference, (2006).

[12] A. Hinneburg, C. Aggarwal, and D. Keim, *What is the nearest neighbor in high dimensional spaces?*, VLDB Conference, (2000).

[13] H.-P. Kriegel, and M. Pfeifle, *Density-based clustering of uncertain data*, ACM KDD Conference, (2005).

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

[15] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch, *Indexing Uncertain Categorical Data*, IEEE ICDE Conference, (2007).

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