Searching Dimension Incomplete Databases

Wei Cheng, Xiaoming Jin, Jian-Tao Sun, Xuemin Lin, Xiang Zhang, and Wei Wang

Abstract—Similarity query is a fundamental problem in database, data mining and information retrieval research. Recently, querying incomplete data has attracted extensive attention as it poses new challenges to traditional querying techniques. The existing work on querying incomplete data addresses the problem where the data values on certain dimensions are unknown. However, in many real-life applications, such as data collected by a sensor network in a noisy environment, not only the data values but also the dimension information may be missing. In this work, we propose to investigate the problem of similarity search on dimension incomplete data. A probabilistic framework is developed to model this problem so that the users can find objects in the database that are similar to the query with probability guarantee. Missing dimension information poses great computational challenge, since all possible combinations of missing dimensions need to be examined when evaluating the similarity between the query and the data objects. We develop the lower and upper bounds of the probability that a data object is similar to the query. These bounds enable efficient filtering of irrelevant data objects without explicitly examining all missing dimension combinations. A probability triangle inequality is also employed to further prune the search space and speed up the query process. The proposed probabilistic framework and techniques can be applied to both whole and subsequence queries. Extensive experimental results on real-life data sets demonstrate the effectiveness and efficiency of our approach.

Index Terms—Dimension Incomplete Database, Similarity Search, Whole Sequence Query

1 Introduction

Similarity query in multidimensional database is a fundamental research problem with numerous applications in the areas of database, data mining, and information retrieval. Given a query object, the goal is to find similar objects in the database [1–4]. Recently, querying incomplete data has attracted extensive research efforts [5–7]. In this problem, the data values may be missing due to various practical issues. For example, in sensor networks, the received data may become incomplete when sensors do not work properly or when errors occur during the data transfer process. The data incompleteness problem studied in the existing work usually refers to *the missing value problem*, i.e., the data values on some dimensions are unknown or uncertain. The common assumption of the existing work is that, for each dimension, whether its data value is missing or not is known. However, in real-life applications, we may not know which dimensions or positions have data loss [8, 9]. In these cases, we only have the arrival order of data values without knowing which dimensions the values belong to. When the dimensionality of the collected data is lower than its actual dimensionality, the correspondence relationship between dimensions and their associated values is lost. We refer to such a problem as *the dimension incomplete problem*.

- *W. Cheng is with the Department of Computer Science, University of North Carolina at Chapel Hill, NC 27599-3175 USA. E-mail:weicheng@cs.unc.edu*
- *X. Jin is with School of Software, Tsinghua University, Beijing 100084, China. E-mail: xmjin@tsinghua.edu.cn*
- *J. Sun is with Microsoft Research Asia, China, Beijing 100080, China. E-mail: jtsun@microsoft.com*
- *X. Lin is with School of Computer Science and Engineering, The University of New South Wales,Sydney, NSW 2052, Australia. E-mail: lxue@cse.unsw.edu.au*
- *X. Zhang is with Department of Electrical Engineering and Computer Science, Case Western Reserve University, 44106 USA. E-mail: xiang.zhang@case.edu*
- *W. Wang is with Department of Computer Science, University of California, Los Angeles, CA 90095-1596 USA. E-mail: weiwang@cs.ucla.edu*

Application 1: Data missing when dimension information is not explicitly maintained. Consider the sensor networks. The database usually contains time series data objects, each of which is represented by a sequence of values $(x_1, x_2, \ldots, x_m)$. The dimension information (e.g., time stamp) associated with data values can be implicitly inferred from the data arrival order. This schema of data collection and storage is very common in resource-constrained applications since explicitly maintaining dimension information will cause additional costs. In this problem setting, missing a single data element will destroy the dimension information of the entire data object.

For example, in Fig. 1, the original data object is (3, 1, 2, 5). When data element 1 is missing, the dimension information for the rest of data elements becomes uncertain. For example, 3 can be the first or the second element, and 2 can be the second or the third element. When data element 1 and 5 are missing, then both element 3 and 2 may locate on three different dimensions. In applications where dimension information is explicitly maintained, the dimension indicator itself may be lost. This will also cause the dimension incomplete problem.

Application 2: Time series data with temporal uncertainty due to imprecise timestamps. The imprecise timestamps due to granularity mismatch, or data collection from distributed system that is lack of clock synchronization may also cause dimension incompleteness in time series data [9]. For example, when time series data are collected from distributed environment, due to the lack of clock synchronization, each collected data value is assigned an

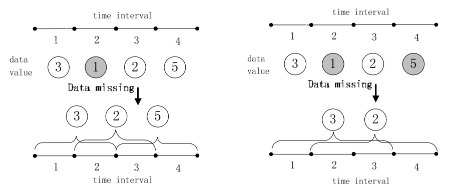


Fig. 1. Dimension incomplete data due to dimension information not being explicitly maintained

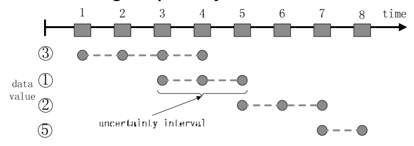


Fig. 2. Dimension incomplete data due to imprecise timestamps

uncertainty interval on the time line. Fig. 2 shows a time series stream of four values $\{3, 1, 2, 5\}$, and their uncertainty intervals on the time line. We may approximately infer the arrival order of the values as (3, 1, 2, 5). But the precise occurrence time of each element is uncertain. In real world applications, we may want to automatically identify patterns(or say subsequences) satisfying given query condition on certain dimension incomplete time series data. This poses the subsequence matching problem on dimension incomplete data.

The dimension incomplete problem brings new challenges to the similarity query task, since the dimension information is essential for the existing uncertain data querying methods [10, 11]. The traditional similarity measurements (or distance functions), which are the bases of any similarity query task, may not be applied when the dimension information is missing. For example, given a query object $Q = (q_1, q_2, \ldots, q_m)$ and a data object $X = (x_1, x_2, \ldots, x_m)$, the ℓ_p-*norms* distance, $\ell_p(Q, X) = (\sum_{i=1}^{m} |q_i - x_i|^p)^{1/p}$, is a widely used distance function in similarity search. However, it can not be calculated if the dimensions of the two objects do not match. Other similarity computation methods, such as dynamic time warping (DTW) [12] and longest common subsequence (LCS) [13], can not be directly applied to dimension incomplete data. They are developed to find the common structures of two sequences, and the exact dimension information is not critical for these methods.

Suppose that the original data dimensionality is m. Given a query object $Q = (q_1, q_2, \ldots, q_m)$ and a dimension incomplete data object $X = (x_1, x_2, \ldots, x_n)$ $(n < m)$, a naive solution to calculate the distance between these two objects is as follows. We examine all possible missing dimension combinations for the data object X. For each combination, we impute the values at the positions with missing data, and then calculate the similarity between Q and the imputed X based on certain distance function such as the ℓ_p-*norms* distance. However this approach is intractable in practice, since there are $\binom{m}{n}$ possible dimension combinations need to be examined. Efficient algorithms are highly desirable.

In this paper, we formulate the problem of similarity query on dimension incomplete data within a probabilistic framework. Using the framework, a user can specify two thresholds: a *threshold of the distance* between the query object and the data object, and a *threshold of the probability* that the retrieved data objects are similar to the query object. An efficient method is developed to compute the lower and upper bounds of the probability that a data object is similar to the query. These bounds can be utilized to effectively prune the search space. Moreover, we develop a probability triangle inequality that can further speed up the query process. Our contributions are summarized as follows.

1) To the best of our knowledge, this is the first work to address the dimension incomplete similarity query problem. This problem has a wide range of applications and poses new technical challenges to traditional query methods. We propose a probabilistic framework to model and manipulate the uncertainty of the data. We also provide theoretical analysis of its computational properties.
2) We develop efficient algorithms to address the challenges in querying dimension incomplete data. The complexity of a naive approach to compute the probability that a data object is similar to the query object is $O(m \cdot \binom{m}{n})$, which is intractable in practice. An efficient method with time complexity $O(n \cdot (m - n)^2)$ is proposed to compute the lower bound and upper bound of the probability. These bounds can be utilized to prune the search space. We further develop a probability triangle inequality, which can be evaluated in $O(m)$ time and used as a filtering tool to further speed up the query process.
3) Our method can be applied to both whole sequence matching and subsequence matching problems on dimension incomplete data. Moreover, the data of interest can be either static data or dynamical data streams.
4) We provide theoretical analysis of the relationship between the probability threshold and the quality of query results. This provides guidance for the users to effectively determine the probability threshold according to their search quality preference.
5) We conduct extensive experiments on real-life data sets. The results demonstrate the effectiveness and efficiency of our method.

The rest of the paper is organized as follows. The related work is reviewed in Section 2. The problem formalization is presented in Section 3. The techniques developed for whole sequence matching are discussed in Section 4. The techniques developed for subsequence matching are discussed in Section 5. The experiment results are reported in Section 7. Section 8 concludes the paper.

2 Related work

Analyzing data with missing values has attracted extensive research interest recently [7, 14–16]. The existing work

TABLE 1
Summary of symbols and their meanings

Symbols	Description
D	Database
X	The underlying complete multidimensional data object
X_o	The observed part of X
X_l	The missing part of X
X_{rv}	The recovery version of X_o
Q	The query
c	The probability threshold
r	The distance threshold
$\Pr[\tau]$	The probability of τ to occur
δ	The distance function
φ	The imputation strategy
$X_o[i]$	The i-th element of X_o
$X_o[i:j]$	The subsequence of X_o, including elements in positions i through j

assumes that the dimensions on which the data values are missing are known. However, in many applications, the dimension information may also be missing.

Analyzing uncertain data is an area that is also related to our work [17, 18]. The goal of these methods is to estimate a probability density function to model the uncertainty in the data. They do not address the problem of dimension incompleteness.

In [8], the authors address the problem where there are missing elements in symbolic sequences. Our problem is more general in the sense that we consider real value data and address the probabilistic query task. In [9], a temporal model is proposed to discover patterns in streams with imprecise timestamps. This work deals with pattern evaluations in event streams where event ids should be exactly matched. Moreover, the data arrival time intervals are needed to construct the temporal uncertainty model. In our work, such information is not available and only the data arrival order is known.

To find common structure of two sequences, dynamic time warping (DTW) [12, 19] and longest common subsequence (LCS) [13, 20] algorithms are proposed. In these problems, the exact dimension information is not critical. These methods can not be directly applied to the similarity query problem on dimension incomplete data.

3 Problem definition

In this section, we formally define the similarity query problem on dimension incomplete data. The main symbols used in the paper are listed in Table 1.

Let D be the database. A data object $X \in D$ is a real-valued vector $(x_1, x_2, \ldots, x_m)$, where x_i $(1 \le i \le m)$ is the data value for the i-th dimension of X. $|X| = m$ denotes the dimensionality of X.

A data object X is *dimension incomplete*, if it satisfies (a) at least one of its data elements is missing; (b) the dimension of the missing data element is unknown. For example, given a complete data object X, if its k data elements are missing, the resulting dimension incomplete data object is of the form $X_o = (x_{o_1}, x_{o_2}, \ldots, x_{o_n})$ where $o_j < o_{j+1}$, $n = |X| - k$.

The traditional range query on a multidimensional database is defined as follows. Given a database D containing N data objects of m dimensions, an m-dimensional query Q, a distance function δ, and a distance threshold r, traditional range queries retrieve all the data objects in D whose distances from Q are less than r. More formally,

$$RangeQuery_{\delta}(D, Q, r) = \{X \in D | \delta(Q, X) < r\} \tag{1}$$

The above problem formulation cannot be directly applied to dimension incomplete data since the distance between the query object and the dimension incomplete data object is not well defined. Intuitively, the distance between the query and the data objects depends on both the dimension alignment and the values estimated for the missing dimensions. In the following, we develop a probabilistic framework to formulate the query problem on dimension incomplete data. The goal is to find all data objects that have high probability to be similar to the query.

For the observed dimension incomplete data X_o, whose underlying complete version is X, there are $\binom{|X|}{|X_o|}$ possible combinations of missing dimensions. For a given combination, the missing dimensions can be modeled as independent random variables following normal distribution. Each random variable can be imputed with certain mean and variance to form a recovery version of X_o, denoted by X_{rv}. Given a distance function δ, e.g., the ℓ_p*-norms* distance, $\delta(Q, X_{rv})$ is also a random variable. We can calculate $\Pr[\delta(Q, X) < r]$ by enumerating all $\binom{|X|}{|X_o|}$ combinations. The problem of querying dimension incomplete data can be formulated as follows.

Definition 3.1 (Probabilistic Similarity Query on Dimension Incomplete Data (PSQ-DID))**.** Given a database D containing dimension incomplete data objects X_o whose underlying complete version is denoted by X, a complete query object Q, an imputation method φ indicating the distribution of missing data values, a distance function δ, a probability threshold c, and a distance threshold r, retrieve all data objects whose distances from Q are less than r with probability greater than c. More formally,

$$\mathrm{PSQ{-}DID}_{\delta,\varphi}(D, Q, r, c) = \{X_o \in D | \Pr[\delta(Q, X) < r] > c\} \tag{2}$$

$\Pr[\delta(Q, X) < r]$ indicates the probability that the underlying complete data object X satisfies the query requirement. It depends on both the imputation strategy φ and the distance function δ.

For a dimension incomplete data object X_o, we can construct a complete recovery version of X_o as follows.

1) Assign a missing dimension combination $\{n_1, \ldots, n_{|X|-|X_o|}\}$, indicating the missing dimensions in X;
2) Impute (determine the mean and variance) for each missing dimension according to φ.

We use X_{rv} and X_l to represent the complete recovery version and the imputed part of X_o respectively. Note that the total number of X_{rv} (and X_l) is $\binom{|X|}{|X_o|}$.

Example 1: Assume that the observed dimension incomplete data object $X_o = (12, 9, 40)$. Its original full dimensionality is $|X| = 5$. There are $\binom{5}{3}$ possible missing dimension combinations. For a combination indicating that

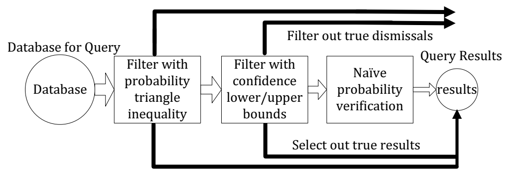

Database for Query
Database
Filter with probability triangle inequality
Filter with confidence lower/upper bounds
Naïve probability verification
Query Results
results
Filter out true dismissals
Select out true results

Fig. 3. The overall query process of the proposed approach

the missing dimensions in X are $\{2, 4\}$, we know that 12, 9, 40 correspond to the first, third, and fifth dimension of X respectively. Then we have that $X_{rv} = (12, x_{i_1}, 9, x_{i_2}, 40)$ and $X_l = (x_{i_1}, x_{i_2})$, where x_{i_1} and x_{i_2} are independent random variables following given distributions. The mean and variance of each random variable can be determined by the imputation strategy adopted by the user.

For each missing dimension combination, we can construct a recovery version X_{rv}. There are in total $\binom{|X|}{|X_o|}$ recovery versions. Assuming the probability of getting each recovery version is equal, we have that

$$\Pr[\delta(Q, X) < r] = \frac{\sum_{X_{rv}} \Pr[\delta(Q, X_{rv}) < r]}{\# X_{rv}}. \tag{3}$$

Next, we discuss the imputation strategy φ and the distance function δ in further detail.

Depending on specific applications, various imputation strategies can be applied within our framework. For simplicity, in this paper, we adopt the following strategy to determine the mean and variance of a random variable. For an underlying complete data object $X = (x_1, x_2, \ldots, x_m)$, if x_i is missing, we impute its mean as $(x_{i-1} + x_{i+1})/2$ (if x_{i-1} or x_{i+1} does not exist, the value of the nearest existing data element is used instead). For example, suppose that the dimensionality of X is 6 and the observed dimension incomplete data object is $X_o = (1, 3, 5)$. For dimension combination whose missing dimensions in X are $\{1, 3, 4\}$, we can construct a recovery version $X_{rv} = (①, 1, ②, ②, 3, 5)$, where the circled values denote the specified mean values of the random variables. For many applications, this imputation strategy is reasonable since the value of missing data element tends to be related to its neighbor elements [21]. The variance of X_o, denoted as σ^2, is used as the variance of the random variables in X_{rv}. Other imputation strategies can also be applied within our framework. For example, we can use the mean value of X_o as the mean value of the random variables. Our framework is *independent* of the imputation strategy applied.

We adopt the widely used Euclidean distance as our distance function δ. Note that the proposed approach can be easily extended to handle other distance functions.

4 Whole Sequence Query on Dimension Incomplete Data

To determine if a dimension incomplete data object X_o is similar to the query Q with high probability, a naive approach is to enumerate all recovery versions of X_o. For each recovery version X_{rv}, we evaluate the probability $\Pr[\delta(Q, X_{rv}) < r]$. Then $\Pr[\delta(Q, X) < r]$ can be computed according to Eq. 3. This approach is not efficient since the total number of recovery versions is $\binom{|X|}{|X_o|}$.

In this section, we present an efficient approach for the PSQ-DID problem introduced in Section 3. The key idea is to evaluate $\Pr[\delta(Q, X) < r]$ without enumerating all possible recovery versions. We propose to utilize a gradual refinement search strategy. Specifically, we develop two strategies to effectively prune the search space: (1) lower and upper bounds of probability $\Pr[\delta(Q, X) < r]$, and (2) a probability triangle inequality. Both the bounds and the inequality can be evaluated very efficiently.

The overall query process is shown in Fig. 3. The probability triangle inequality is first applied to evaluate the data objects. In this step, some data objects are judged as true results and some are filtered out. The lower and upper bounds of the probability are then applied to evaluate the remaining data objects, from which some are determined as true results and some as dismissals. Only those data objects that can not be determined in the former two steps are evaluated by the naive method.

In the following, we will firstly discuss the bounds of the probability and then present the probability triangle inequality.

4.1 Bounds of the Probability

In this section, we develop the lower and upper bounds of the probability $\Pr[\delta(Q, X) < r]$, the proof of their correctness, and an efficient algorithm for calculating them.

Given a query Q and a recovery version X_{rv} of dimension incomplete data X_o, we have that

$$\delta^2(Q, X_{rv}) = \delta^2(Q_o, X_o) + \delta^2(Q_l, X_l), \tag{4}$$

where Q_o and Q_l represent the data values of Q on the same dimensions as those of X_o and X_l respectively. The total number of Q_o's (and Q_l's) is the same as the number of X_{rv}'s, since each X_{rv} corresponds to a Q_o and a Q_l. Note that $r \geq 0$, thus we have that

$$\Pr[\delta(Q, X_{rv}) < r] = \Pr[\delta^2(Q_l, X_l) < r^2 - \delta^2(Q_o, X_o)]. \tag{5}$$

Given X_{rv}, $\delta^2(Q_o, X_o)$ is a real value and $\delta^2(Q_l, X_l)$ is a random variable. We can determine the lower and upper bounds on the distance between X_o and Q as follows.

$$\delta_{LB_o}(Q, X_o) = \min_{|Q_o|=|X_o|} \delta(Q_o, X_o), \tag{6}$$

$$\delta_{UB_o}(Q, X_o) = \max_{|Q_o|=|X_o|} \delta(Q_o, X_o). \tag{7}$$

where the right hand side of Equation 6 (Equation 7) represents the minimum (maximum) distance among all possible combinations of missing dimensions.

Similarly, we can determine the lower bound and upper bound on distance between X_l and Q_l as follows.

$$\begin{aligned}&\delta_{LB_l}(Q, X_l)\\ &= \delta(\underset{Q_l}{\operatorname{argmin}}\{\delta(Q_l, E(X_l)) | \quad |Q_l| = |X_l|\}, X_l),\end{aligned} \tag{8}$$

$$\begin{aligned}&\delta_{UB_l}(Q, X_l)\\ &= \delta(\underset{Q_l}{\operatorname{argmax}}\{\delta(Q_l, E(X_l))| \quad |Q_l| = |X_l|\}, X_l),\end{aligned} \tag{9}$$

where $E(X_l) = (\mu_1, \mu_2, \ldots, \mu_{|X|-|X_o|})$, and μ_k is the mean value assigned by the imputation method on the k-th dimension of X_l. Note that both $\delta_{LB_l}(Q, X_l)$ and $\delta_{UB_l}(Q, X_l)$ are random variables rather than specific values.

Example 2: Consider a dimension incomplete data $X_o = (2, 8, 7)$ and a query Q=$(1, 4, 5, 6, 7)$. The lower bound for the observed data $\delta^2_{LB_o}(Q, X_o) = (2-1)^2 + (8-6)^2 + (7-7)^2 = 5$, corresponding to the recovery version (2,?,?,8,7), where "?" denotes the imputed random variable. The upper bound $\delta^2_{UB_o}(Q, X_o) = (2-1)^2 + (8-4)^2 + (7-5)^2 = 21$, corresponding to the recovery version $(2, 8, 7, ?, ?)$. For the imputed random variables X_l=$\{x_1, x_2\}$, according to our imputation policy, $E(x_1)$ and $E(x_2)$ rely on the dimensions to be imputed. We have that $\delta^2_{LB_l}(Q, X_l)$=$(4$-$x_1)^2$+$(5$-$x_2)^2$ ($E(x_1) = E(x_2) = 5$) corresponding to X_{rv}=$(2, \textcircled{5}, \textcircled{5}, 8, 7)$, and $\delta^2_{UB_l}(Q, X_l) = (5-x_1)^2+(6-x_2)^2$ ($E(x_1) = E(x_2) = 7.5$) corresponding to $X_{rv} = (2, 8, \textcircled{7.5}, \textcircled{7.5}, 7)$.

Based on the distance bounds discussed above, we can determine the lower and upper bounds of $\Pr[\delta(Q, X) < r]$ according to the following theorem.

Theorem 4.1. *Given a query Q, threshold r and c, for an incomplete multidimensional data X_o whose complete form is denoted by X, we have*
(a) $\Pr[\delta(Q, X) < r] \le \Pr[\delta^2_{LB_l}(Q, X_l) + \delta^2_{LB_o}(Q, X_o) < r^2]$;
(b) $\Pr[\delta(Q, X) < r] \ge \Pr[\delta^2_{UB_l}(Q, X_l) + \delta^2_{UB_o}(Q, X_o) < r^2]$.

Proof: (a)For any recovery version X_{rv} of X_o, according to Eq. 5, we have

$$\begin{aligned}&\Pr[\delta^2(Q, X_{rv}) < r^2]\\ &= \Pr[\delta^2(Q_l, X_l) < r^2 - \delta^2(Q_o, X_o)]\end{aligned} \tag{10}$$

According to Eq. 6 we know

$$\delta^2(Q_o, X_o) \ge \delta^2_{LB_o}(Q, X_o) \tag{11}$$

Thus

$$\begin{aligned}&\Pr[\delta^2(Q_l, X_l) < r^2 - \delta^2_{LB_o}(Q, X_o)]\\ &\ge \Pr[\delta^2(Q_l, X_l) < r^2 - \delta^2(Q_o, X_o)]\end{aligned} \tag{12}$$

Here, random variable $\delta^2(Q_l, X_l)/\sigma^2$ obeys a *noncentral chi-square distribution* with noncentrality parameter $\lambda_{X_{rv}}$=$\delta^2(Q_l, E(X_l))/\sigma^2$ and random variable $\delta^2_{LB_l}(Q, X_l)/\sigma^2$ also obeys a noncentral chi-square distribution with noncentrality parameter denoted by λ_{LB_l}. According to Eq. 8, we know $\lambda_{LB_l} \le \lambda_{X_{rv}}$. In addition, these two random variables have the same degree of freedom $|X_l|$. According to the property of noncentral chi-square distribution [22], we know

$$\begin{aligned}&\Pr[\delta^2_{LB_l}(Q, X_l) < r^2 - \delta^2_{LB_o}(Q, X_o)]\\ &\ge \Pr[\delta^2(Q_l, X_l) < r^2 - \delta^2_{LB_o}(Q, X_o)]\end{aligned} \tag{13}$$

Also considering Eq. 12, we have that

$$\begin{aligned}&\Pr[\delta^2_{LB_l}(Q, X_l) + \delta^2_{LB_o}(Q, X_o) < r^2]\\ &\ge \Pr[\delta^2(Q, X_{rv}) < r^2]\end{aligned} \tag{14}$$

Since for any recovery version X_{rv} of X, the Eq. 14 holds, from Eq. 3, we conclude that

$$\begin{aligned}&\Pr[\delta^2_{LB_l}(Q, X_l) + \delta^2_{LB_o}(Q, X_o) < r^2]\\ &\ge \Pr[\delta(Q, X) < r]\end{aligned} \tag{15}$$

(b)The proof is similar to that of (a). □

For notational simplicity, we denote

$$\delta_{LB}(Q, X) = [\delta^2_{LB_l}(Q, X_l) + \delta^2_{LB_o}(Q, X_o)]^{1/2} \tag{16}$$

$$\delta_{UB}(Q, X) = [\delta^2_{UB_l}(Q, X_l) + \delta^2_{UB_o}(Q, X_o)]^{1/2} \tag{17}$$

4.2 Efficient Bound Evaluation

According to Theorem 4.1, $\Pr[\delta_{LB}(Q, X) < r]$ and $\Pr[\delta_{UB}(Q, X) < r]$ are the upper bound and lower bound of $\Pr[\delta(Q, X) < r]$ respectively. These two probability bounds can be used for filtering purpose in the query process. Specifically, we can select data objects with $\Pr[\delta_{UB}(Q, X) < r] > c$ as true results and filter out data objects with $\Pr[\delta_{LB}(Q, X) < r] \le c$ as true dismissals. The correctness of this pruning process is guaranteed by Theorem 4.1.

To utilize this pruning strategy, we need efficient algorithms for (1) calculating the probability $\Pr[\delta_{LB}(Q, X) < r]$ and $\Pr[\delta_{UB}(Q, X) < r]$, and (2) calculating the distance bounds shown in Equations 6 to 9.

The first sub-problem can be solved easily. Since $\delta^2_{LB_l}(Q, X_l)/\sigma^2$ and $\delta^2_{UB_l}(Q, X_l)/\sigma^2$ obey noncentral chi-square distribution, these two probabilities can be calculated using the cumulative distribution function of noncentral chi-square distribution or by a table lookup approach.

Consider the second sub-problem. The naive method to compute any one of the four bounds is extremely time consuming since we have to enumerate all the $\binom{|X|}{|X_o|}$ recovery versions.

Next we introduce a dynamic programming algorithm to compute these four bounds in $O(|X_o| \cdot (|X| - |X_o|)^2)$ time. Algorithm 1 is for calculating $\delta_{LB_o}(Q, X_o)$ and $\delta_{LB_l}(Q, X_l)$. After the algorithm is executed, the minimum element in the $2n$-th column of matrix T is $\delta_{LB_o}(Q, X_o)$, and $\delta_{LB_l}(Q, X_l)$ can be inferred from the assistant array S. In order to calculate $\delta_{UB_o}(Q, X_o)$ and $\delta_{UB_l}(Q, X_l)$, only a small modification is needed: replace function *min* in line 17, 21 by *max* and replace *argmin* in line 11 by *argmax*. The algorithm does not require building the entire table T. Its computational complexity is $O(|X_o| \cdot (|X| - |X_o|)^2)$, which is a significant improvement over the naive method which has to enumerate all $\binom{|X|}{|X_o|}$ missing dimension combinations.

Example 3: For Algorithm 1, consider a query $Q = (3, 7, 1, 6, 5)$, and a data object $X_o = (2, 4, 8)$. We have that $X' = (\textcircled{2}, 2, \textcircled{3}, 4, \textcircled{6}, 8, \textcircled{8})$, where the circled elements are determined by the imputation strategy. The initialized T is shown in Fig. 4(a). The algorithm starts the calculation from the bottom of the first column to top right. In step.1, $T[1][1]$=1 remains unchanged. $T[1][2] = 25$ is replaced with $T[1][1]$+$T[2][1]$= $25 + 1 = 26$. In the second column, we do nothing. In Step.2, we deal with the third column of T. $T[3][3]$=4 is replaced with

Algorithm 1 Calculate $\delta_{LB_o}(Q, X_o)$ and $\delta_{LB_l}(Q, X_l)$

INPUT: Query Q, $|Q|$=m and dimension incomplete data object X_o, $|X_o|$=n $(0 < n < m)$.
OUTPUT: $\delta_{LB_o}(Q, X_o)$ and $\delta_{LB_l}(Q, X_l)$ (inferred from assistant array S).
Initialization Step: Extend X_o to X', ($|X'|$=$2n + 1$), where

$$X'_i = \begin{cases} X_o[1] & i = 1, \\ X_o[n] & i = 2n+1, \\ X_o[i/2] & i \mod 2 = 0, \\ \frac{X_o[(i-1)/2]+X_o[(i+1)/2]}{2} & i \mod 2 = 1, 1 < i < 2n+1 \end{cases}$$

Construct two $m\times(2n+1)$ matrices T and S, where the (i-th,j-th) element of T is initialized to $(Q_i - X'_j)^2$, S is an assistant array initialized with $(0, 0)$ for each element.

```
1: for $j$=1 to $2n+1$ do
2:   if $j$=1 then
3:     for $i$=1 to $m-n$ do
4:       $S[i][j]$←($i$-1,1)
5:       if $i > 1$ then
6:         $T[i][j]$←$T[i][j]$+$T[i-1][j]$
7:       end if
8:     end for
9:   else if $j$>2 and $j$ mod 2=1 then
10:    for $i$=$\frac{(j+1)}{2}$+1 to $\frac{(j+1)}{2}$+$m$-$n$-1 do
11:      $p \leftarrow \underset{1\le k\le \frac{(j+1)}{2}}{\text{argmin}}\ T[i-k][j-2(k-1)]$
12:      $T[i][j]$←$T[i][j]$+$T[i-p][j-2(p-1)]$
13:      $S[i][j]$←$(i-p, j-2(p-1))$
14:    end for
15:  else if $j$>2 and $j$ mod 2=0 then
16:    for $i$=$\frac{j}{2}$ to $\frac{j}{2}$+$m$-$n$ do
17:      $T[i][j]$←$T[i][j]$+$\min_{\frac{(j-2)}{2}\le k\le i-1} T[k][j-2]$
18:    end for
19:  end if
20: end for
21: return $(\min_{n\le k\le m} T[k][2n])^{\frac{1}{2}}$
```

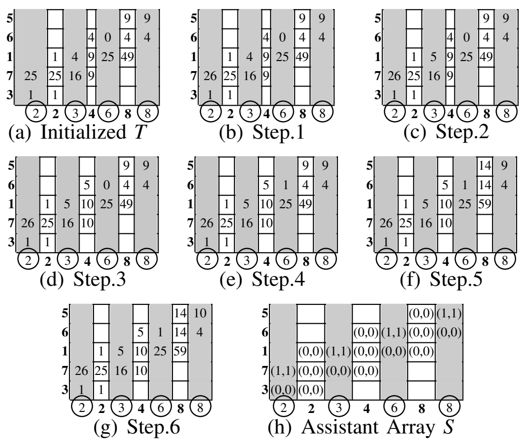


Fig. 4. Example of getting $\delta_{LB_o}(Q, X_o)$ and $\delta_{LB_l}(Q, X_l)$

$T[3][3]$+$min\{T[2][3], T[1][1]\}$=4+$min\{16, 1\}$ = 5. The remaining steps are shown in Fig. 4. Finally, from the sixth column in Fig. 4(g), we find $\delta^2_{LB_o}(Q, X_o)$ is 14, which is the minimal element in the column. In order to find $\delta^2_{LB_l}(Q, X_l)$, we first find the minimal value among those in the top of column 1,3,5,7 of T. That is min{26,5,1,10}. We find $T[4][5]$=1 is the minimal. Thus the imputed variable with mean value 6 will be matched to 6 in Q. Then $S[4][5]$=$<1, 1>$ indicates the previous match is in the first row and first column of T, where the corresponding imputation is with mean value 2 and is matched to 3 in Q. Thus, $\delta^2_{LB_l}(Q, X_l)$=$(3 - x_1)^2$+$(6 - x_2)^2$, where x_1 and x_2 are imputed random variables with mean value 2 and 6 respectively.

4.3 Probability Triangle Inequality

In this section, we present a probability triangle inequality which can also be used to effectively prune the search space.

Theorem 4.2. *Given a query object Q, a dimension complete data object R (i.e., $|R| = |Q|$), and a dimension incomplete data object X_o whose underlying complete version is X, we have that*

(a)$\Pr[\delta(Q, X) < r] \le \Pr[\delta_{LB}(R, X) - \delta(Q, R) < r]$*;*
(b)$\Pr[\delta(Q, X) < r] \ge \Pr[\delta_{UB}(R, X) + \delta(Q, R) < r]$.

Proof: (a) From Theorem 4.1, we have

$$\Pr[\delta_{LB}(R, X) - \delta(Q, R) < r] \ge \Pr[\delta(R, X) < \delta(Q, R) + r] \quad (18)$$

Thus, for any recovery version X_{rv} of X, we have

$$\Pr[\delta(R, X_{rv}) - \delta(Q, R) < r] \le \Pr[\delta_{LB}(R, X) - \delta(Q, R) < r] \quad (19)$$

In a metric space, the triangle inequality holds, thus

$$\delta(R, X_{rv}) - \delta(Q, R) \le \delta(Q, X_{rv}) \quad (20)$$

Thus we have

$$\Pr[\delta(Q, X_{rv}) < r] \le \Pr[\delta(R, X_{rv}) - \delta(Q, R) < r] \quad (21)$$

Here, since X_{rv} is any recovery version of X, recall Eq. 3, thus we have

$$\Pr[\delta(Q, X) < r] \le \Pr[\delta(R, X_{rv}) - \delta(Q, R) < r] \quad (22)$$

Therefore,

$$\Pr[\delta_{LB}(R, X) - \delta(Q, R) < r] \ge \Pr[\delta(Q, X) < r] \quad (23)$$

(b) The proof is similar to that of (a). □

Based on the theorem above, with the help of assistant data object R, some data objects in database can be determined to be true results (when $\Pr[\delta_{UB}(R, X) + \delta(Q, R) < r] \ge c$) or true dismissals (when $\Pr[\delta_{LB}(R, X) - \delta(Q, R) < r] \le c$). Since the required dynamic programming computation can be finished in advance without knowing the query, this evaluating process can be done in $O(|Q|)$ time.

In order to get $\Pr[\delta_{LB}(R, X) - \delta(Q, R) < r]$ and $\Pr[\delta_{UB}(R, X) + \delta(Q, R) < r]$ efficiently, two sets of values need to be stored for each (R, X_o) pair. One is $\delta_{LB_o}(R, X_o)$ and $\delta_{UB_o}(R, X_o)$. The other is the noncentrality parameters of random variables $\delta^2_{LB_l}(R, X_l)/\sigma^2$ and $\delta^2_{UB_l}(R, X_l)/\sigma^2$. In real applications, the assistant data objects can be obtained by sampling the user's query log. The two sets of values can be calculated and pooled in a batch mode for online query processing. The number of assistant data objects controls the tradeoff between the query processing time and the storage. Its effectiveness will be studied in the experiments.

Algorithm 2 Probabilistic Similarity Query

INPUT: the dimension incomplete database D, query Q, distance threshold r, probability threshold c, the set of assistant data objects S_R.
OUTPUT: the results set S_{result}.

```
1:  for all $X$ in $D$ do
2:    for all $R$ in $S_R$ do
3:      if $\Pr[\delta_{LB}(R,X) - \delta(Q,R) < r] \leq c$ then
4:        $X \notin S_{result}$
5:        goto 1 to evaluate next $X$ in $D$
6:      else if $\Pr[\delta_{UB}(R,X) + \delta(Q,R) < r] > c$ then
7:        $X \in S_{result}$
8:        goto 1 to evaluate next $X$ in $D$
9:      end if
10:   end for
11:   if $\Pr[\delta_{LB}(Q,X) < r] \leq c$ then
12:     $X \notin S_{result}$
13:   else if $\Pr[\delta_{UB}(Q,X) < r] > c$ then
14:     $X \in S_{result}$
15:   else if $\Pr[\delta(Q,X) < r] > c$ then
16:     $X \in S_{result}$
17:   else
18:     $X \notin S_{result}$
19:   end if
20: end for
```

4.4 The Overall PSQ-DID Algorithm

The overall PSQ-DID (Probabilistic Similarity Query on Dimension Incomplete Data) algorithm is shown in Algorithm 2. The algorithm utilizes a gradual refinement searching strategy with aforementioned pruning strategies to speed up the query process. Specifically, we first use assistant objects in S_R to examine data objects based on probability triangle inequality in Theorem 4.2. Then lower and upper bounds on the probability shown in Theorem 4.1 are used to further evaluate the remaining candidates. The remaining objects that cannot be determined by the previous pruning strategies will be evaluated by the naive verification algorithm.

The triangle inequality and the bounds of probability can be evaluated efficiently and used to effectively prune the search space. Only a small portion of the data objects need to be evaluated by the naive verification algorithm. To further increase the efficiency, we can avoid the naive verification step and simply treat the remaining candidates as query results (or dismissals, depending on the requirements of query precision and recall). This is reasonable for applications where the two probability bounds are effective for selecting candidates. This simplified strategy will dramatically increase the efficiency of the algorithm without causing significant change of the quality of the results. The experimental results shown in Section 7.2.3 demonstrate the effectiveness of the simplified algorithm.

4.5 Improving the Efficiency of Naive Probability Evaluation

Recall that from Eq. 3 in Section 3, a straightforward way for probability evaluation is to examine all possible recovery versions of the incomplete data. In this section, we provide an optimized enumeration process that prunes some recovery versions safely. From Eq. 4, we know that if $\delta(Q_o, X_o) \geq r$, $\Pr[\delta^2(Q_l, X_l) < r^2 - \delta^2(Q_o, X_o)]$ will be 0. Intuitively, if $\delta_{LB_o}(Q, X_o) \geq r$ and thus $\delta(Q_o, X_o) \geq r$, there is no need to examine X_o. Furthermore, according to Eq. 6, we have the following theorem:

Theorem 4.3. *Given a query Q and a dimension incomplete data object X_o ($|X_o| < |Q|$), if there is a Q' derived from Q (by eliminating some data values of Q) ($|Q'| \geq |X_o|$) that satisfies $\delta_{LB_o}(Q', X_o) \geq r$, then for all Q'' derived from Q' ($|Q''| \geq |X_o|$) we have $\delta_{LB_o}(Q'', X_o) \geq r$.*

Proof: Since $\delta_{LB_o}(Q', X_o) \geq r$, we assume there exists a Q'' derived from Q' ($|Q''| \geq |X_o|$), and $\delta_{LB_o}(Q'', X_o) < r$. According to Eq. 6, there will exist a Q''' derived from Q'' with length $|X_o|$ and $\delta(Q''', X_o) < r$. Since Q''' is also derived from Q', we have $\delta_{LB_o}(Q', X_o) < r$ which leads to a contradiction. Thus the theorem is proved. □

This theorem enables us to evaluate only part of the $\binom{|X|}{|X_o|}$ recovery versions that yield probability larger than 0. Based on the theorem, we can recursively enumerate the recovery versions. For Q' derived from Q ($|Q'| \geq |X_o|$), if $\delta_{LB_o}(Q', X_o) \geq r$, all Q'' derived from Q' do not need to be evaluated. Due to space limitation, we will not discuss the details of this recursive procedure.

5 Subsequence Matching on Dimension Incomplete Data

In this section, we discuss the problem of subsequence matching on dimension incomplete data. The probabilistic definition and framework introduced in Sections 3 and 4 for whole sequence query can be extended to handle this problem.

5.1 Problem Description

Definition 5.1 (Probabilistic Subsequence Matching on Dimension Incomplete Data (PSM-DID))**.** Given a database D containing dimension incomplete sequences of real numbers X_o whose underlying complete version is of the length that is potentially different and unknown, a query sequence Q of length $|Q|$, a distance threshold r, a probability threshold c, an imputation method φ indicating the distribution of missing data values, and a distance function δ,

$$\begin{aligned}&\text{PSM-DID}_{\delta,\varphi}(D, Q, r, c)\\ &= \{X_o[i:j] \mid \Pr\{\delta(Q, X_{rv}[k : k + |Q| - 1]) < r\} \geq c,\\ &\qquad X_o \in D, j \geq i - 1\}\end{aligned} \tag{24}$$

Here, $X_o[i : j] = (X_o[i], ..., X_o[j])$ stands for the subsequence of X_o, including elements in positions i through j, and $X_{rv}[k : k+|Q|-1]$ is the recovery version for $X_o[i : j]$ of length $|Q|$. There is a special case, when $j = i-1$, it indicates that the missing data elements are between $X_o[i-1]$ and $X_o[i]$.

5.2 Algorithm for Subsequence Matching on Dimension Incomplete Data

In this subsection, we will show that our framework on whole sequence queries can be extended to handle subsequence queries. Without taking into account the influence of boundary elements in determining the expected values

Algorithm 3 Subsequence matching on dimension incomplete data

INPUT: the dimension incomplete sequence database D, query Q, distance threshold r, probability threshold c.
OUTPUT: the result set S_{result}.

```
1: for all $X$ in $D$ do
2:     for $k$=1 to $|X|$ do
3:         for $length$ = 0 to $min\{|X| - k + 1, |Q|\}$ do
4:             $X_{sub} \leftarrow X[k : k + length - 1]$
5:             if $\Pr[\delta(Q, X_{sub}) < r] > c$ then
6:                 $X_{sub} \in S_{result}$
7:             else
8:                 $X_{sub} \notin S_{result}$
9:             end if
10:        end for
11:    end for
12: end for
```

of random variables, we can divide the process of the subsequence query problem as follows.

Step 1:Tackle the case that matches the subsequence from the first element of the target dimension incomplete sequence;

Step 2:Remove the first element of target dimension incomplete sequence and repeat Step 1; Terminate until there is no element left in target sequence.

One straightforward method for Step 1 is to divide the problem into several cases and examine each case separately. For example, given query $Q = (4, 1, 3)$ and dimension incomplete sequence $X_o = (3, 1, 2, \ldots, 5)$, we can divide Step 1 into four cases as illustrated in Fig. 5. The first case assumes that all elements to be matched with the query are missing, and there are three random variables need to be imputed. Next, we consider the case where only one element is chosen to match the query, and the remaining two elements are assumed to be missing. The third and fourth cases are similar to the second case. Obviously, the first and fourth cases can be evaluated easily, and the second and the third cases can be processed with the algorithms discussed in Section 4. The time complexity of this approach is $O(|X_o|\cdot\{\sum_{i=1}^{|Q|-1} i\cdot(|Q|-i)^2\}) = O(|X_o|\cdot|Q|^4)$. Note that in many real-life applications, we have $|X_o| \gg |Q|$.

The detailed algorithm is presented in Algorithm 3. Note that when $length = 0$, X_{sub} is an empty sequence and all imputed random variables are with mean value $X[k]$. In the fifth line of the algorithm, in order to determine whether $\Pr[\delta(Q, X_{sub}) < r] > c$, the aforementioned probability triangle inequality and probability upper and lower bounds can be applied. In practice, we can set constraint on the minimal *length* of the subsequence used to match the query. This constraint reveals the prior knowledge for the missing ratio of dimension incomplete sequences. For example, a smaller value for the minimal *length* is better for larger missing ratio.

To improve the efficiency of Step 1, we develop an algorithm which only needs to calculate table T once rather than $|Q|$ times. Only three lines of Algorithm 1 need to be changed:

(1) change $m - n$ in line 3 to m;

(2) change $\frac{(j+1)}{2}$+m-n-1 in line 10 to m;

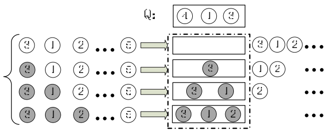


Fig. 5. Subsequence matching on dimension incomplete data

(3) change $\frac{j}{2}$+m-n in line 16 to m.

In other words, all elements in the upper triangle of T and S will be calculated. The calculation of corresponding probability upper and lower bounds can be fulfilled with the help of T and S. In order to do Step 2, we only need to treat the third column of T as the first column and recalculate T and S. Since we only need to calculate the array values once for Step 1, the efficiency can be improved. Without applying the probabilistic triangle inequality, the time complexity is $O(|X_o| \cdot |Q|^3)$. If the inequality is applied, during the query processing, as shown in Fig 5, Step 1 requires execute triangle inequality filtering $|Q| - 1$ times, each of them is with time complexity $O(|Q|)$. For the whole sequence X_o, there are totally $|X_o|$ times of execution of Step 1. Thus the overall complexity becomes $O(|Q|^2 \cdot |X_o|)$.

In addition, in our algorithm for subsequence matching on dimension incomplete data, the data of interest can be either static data or dynamical data streams. A special case of this problem would be monitoring the data streams, where the sequence in database is appended over time, and once the sequence is updated, we examine if the newest part of the sequence is similar to a user specified query. It can be found that our method for subsequence matching can also tackle this problem, with only a little modification required. Specifically, when the sequence is updated, we only have to do the two steps on the new added part of the sequence. If we find the matching patterns that meet the query requirements, the system will output the matched subsequences.

6 Relationship between Probability Threshold and Quality of Query Results

Probability threshold c is an important user-specified parameter for our approach. In this section, we study the relationship between the probability threshold c and the quality of query results (precision and recall).

For conciseness and clearness of analysis, without loss of generality, we first shift the data space so that the query Q is transformed to $Q' = (0, 0, ..., 0)$. In the transformed space, we want to find data objects X' that satisfies

$$\begin{aligned} &\Pr(\|X'\|_2 < r) > c \\ \Leftrightarrow\ &\Pr(\|X'_l\|_2^2 < r^2 - \|X'_o\|_2^2) > c. \end{aligned} \tag{25}$$

Here, X'_l and X'_o are the missing part and the observed part of X', respectively. There are totally $\binom{|Q'|}{|X'_o|}$ possible recovery versions for dimension incomplete data X'_o. Consider the i-th recovery version X'_{rv_i}, the region that contains all the data objects that will be retrieved, denoted by R_i. The

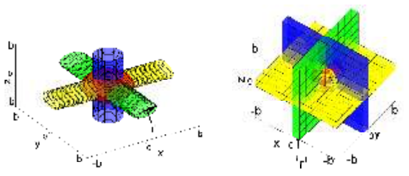

(a) n=2 (b) n=1

Fig. 6. 3-dimensional data objects with n dimensions observed

union of all $\binom{|Q'|}{|X'_o|}$ regions is the region containing all query results outputted by our approach. On the other hand, the region that contains all the true results is a hyper-sphere R_{sphere} with the center in the origin, and radius r. For simplicity, we make the assumption that the data objects are uniformly distributed in the multidimensional space with data element of each dimension ranging from $-b$ to b. So given a region R in the space, the number of data objects within it is proportional to its volume, denoted by $V(R)$. The volume of this region R can be determined by $|Q'|$, $|X'_l|$, c and b. Based on the above notions, the expected precision and recall (denoted by $\mathbb{E}(precision)$ and $\mathbb{E}(recall)$ respectively) are as :$\mathbb{E}(precision) = V(\cup R_i \cap R_{sphere})/V(\cup R_i)$ and $\mathbb{E}(recall) = V(\cup R_i \cap R_{sphere})/V(R_{sphere})$.

Taking three dimensional data objects for instance, for simplicity, we assume the mean and variance are given values, then the shape for the retrieved results region and the real results region are shown in Fig. 6. The red sphere is the real results region, and its volume is $V(R_{sphere})$. Fig. 6(a) presents the case of 3-dimensional data with one dimension missing. The three cylinders are the retrieved results region whose volume is $V(\cup R_i)$ and $\cup R_i \cap R_{sphere}$ will be the region of three cylinders within the sphere. Fig. 6(b) shows the case of 3-dimensional data with two dimensions missing. The three cubes are the region of retrieved results. In Fig. 6, the volume of the retrieved results region is controlled by r' which is determined by probability threshold c, the lager the value of c is, the smaller r' will become.

Given $|Q'|$, $|X'_l|$, r and b, with the increase of c , both $V(\cup R_i)$ and $V(\cup R_i \cap R_{sphere})$ decrease. Thus, $\mathbb{E}(recall)$ decreases with the increase of c. When $r \leq b$, with the increase of c, the decreasing rate of $V(\cup R_i \cap R_{sphere})$ will be smaller than that of $V(\cup R_i)$. Thus, $\mathbb{E}(precision)$ increases with the increase of c. This indicates that c can be chosen to trade off between precision and recall based on the understanding of application domain.

7 Experimental Results

In this section, we present extensive experimental results on evaluating the performance of our approach. We (a) evaluate the robustness of the proposed probabilistic framework and the effectiveness of its algorithmic components; (b) study the influence of the parameters; (c) compare the performance of our approach with other alternative methods.

7.1 Data Sets

Two real data sets are used in the experiments. The first one is the Standard and Poor 500 index historical stock data[1]. This data set contains stock prices of 541 companies collected over one year (from 2008-03-07 to 2009-03-06). The opening stock prices are used as data objects. To increase the sample size, each price vector is truncated into 8 segments each with 30 dimensions. The resulting data set contains 4,328 data objects with 30 dimensions (denoted by S&P500). The second data set is the color histograms of Corel image features (denoted by IMAGE)[2]. It contains 32-dimensional image features extracted from 68,040 images of the Corel image collection. For both data sets, the original data objects are complete. Similarity query results on the complete data are used as the "ground truth" to evaluate the precision and recall of our approach.

We construct the dimension incomplete data sets by randomly removing some dimensions of each complete data object. The number of missing data elements are controlled by *missing ratio* which is the percentage of the missing dimensions. For whole sequence query problem, 100 randomly sampled complete data objects are used as queries. For subsequence matching problem, the first 5 dimensions of these 100 data objects are used as the query objects.

7.2 Results and Analysis

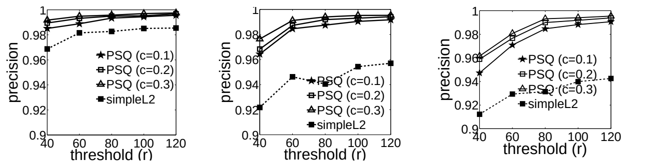
7.2.1 Effectiveness of the Probabilistic Framework

Because we use query results on the complete data as the "ground truth", when the probability threshold used is between 0 to 1, both *false positive* and *false negative* might occur. In order to evaluate how well the proposed probabilistic framework can model the uncertainty caused by the dimension incompleteness, we use two standard measures, precision and recall: $precision = \frac{|tp|}{|S_{result}|}$, $recall = \frac{|tp|}{|S_{true}|}$, where tp stands for true positives, i.e., the retrieved data objects whose complete forms are in the "ground truth", S_{true} stands for the "ground truth" results, S_{result} is the retrieved dimension incomplete data objects.

Figures 7, 8, 9 and 10 show the quality of query results. It can be observed that our method (PSQ) achieves high precision and recall on both data sets. These results demonstrate the effectiveness of the proposed probabilistic framework. We also compare our approach with a simple method ($simpleL2$), which randomly removes elements of the query to construct a new query with the same dimensionality as the dimension incomplete objects in the database. If the distance between the data object and the constructed query object is less than r, $simpleL2$ will report it. From the results, we can see that our method models the underlying problem better hence achieves better query quality.

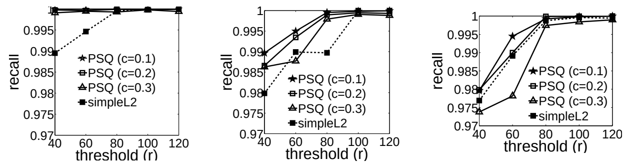
Our method performs better on the S&P500 data set than on the IMAGE data set. This is due to the intrinsic characteristics of these two data sets. S&P500 data set is

1. http://kumo.swcp.com/stocks/
2. http://kdd.ics.uci.edu/databases/CorelFeatures/CorelFeatures.data.html



(a) 5% missing ratio (b) 10% missing ratio (c) 15% missing ratio

Fig. 7. Query precision on S&P500 data set



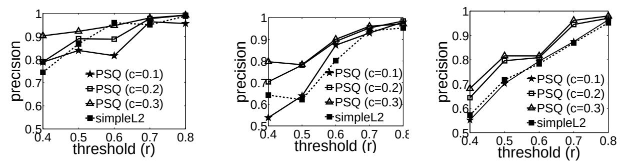
(a) 5% missing ratio (b) 10% missing ratio (c) 15% missing ratio

Fig. 8. Query recall on S&P500 data set

a typical time series data set. An example data object is "55.21 52.87 52 53 50.44 ...". The adopted imputation method (imputing normally distributed random variable with expectation equal to the mean of nearest existing data elements) suits this application well. On the other hand, the image histogram data is not a time series data set. The adopted imputation method cannot accurately assign expected values for the random variables. This indicates the importance of the imputation method. Note that our method is independent of the imputation method applied. The user can decide what imputation method to use depending on the application.

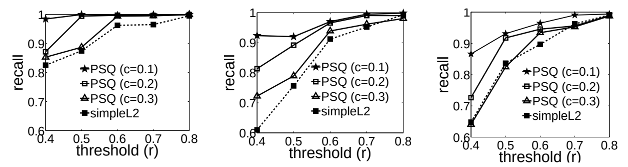
7.2.2 *Effect of the Probability Threshold*

It can be observed from Figures 7, 8, 9 and 10 that recall value decreases and precision value increases with the increase of the probability threshold. Fig. 11 shows the relationship between the probability threshold c and precision/recall (*missing ratio*=0.1, r=60 for S&P500 and r=0.7 for IMAGE). It can be seen that our method is very robust with respect to the probability threshold. This indicates that c can be chosen to trade off between precision and recall based on the understanding of application domain. In the Appendix, we provide theoretical analysis of the relationship between probability threshold and quality of query results formally.



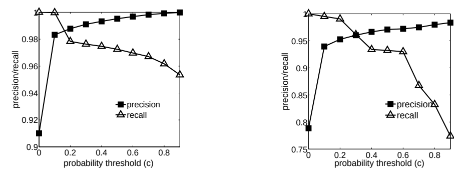
(a) 5% missing ratio (b) 10% missing ratio (c) 15% missing ratio

Fig. 9. Query precision on IMAGE data set



(a) 5% missing ratio (b) 10% missing ratio (c) 15% missing ratio

Fig. 10. Query recall on IMAGE data set



(a) S&P500 data set (b) IMAGE data set

Fig. 11. Probability threshold vs precision/recall

7.2.3 *Effectiveness of Different Pruners*

In this section, we study the effectiveness of the pruners proposed in this paper by examining their pruning power. Their effectiveness is measured by $\frac{N_{pruned}}{N}$ where N is the number of the data objects in the database, and N_{pruned} is the number of data objects in the database judged as dismissals or search results by the pruner.

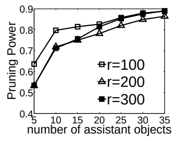
Fig. 12 shows the overall pruning power of probability triangle inequality with various number of assistant data objects ($c = 0.2$). For S&P500 data, when only 10 assistant data objects were used, the pruning power is more than 60%. For IMAGE data, in most cases, the pruning power of the probability triangle inequality is more than 20% with 20 assistant data objects.

The results show that the probability triangle inequality has good pruning power by involving only a few assistant data objects. Moreover, the performance will improve when more assistant data objects are available. After the pruning power reaches a certain level, the increase of assistant data objects has no significant further impact on the pruning power.

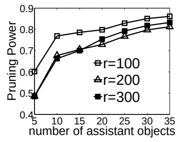
For a more detailed study, we examine the pruning power of the following four pruners: probability triangle inequality (a) in Theorem 4.2 (*pruner*1), probability triangle inequality (b) in Theorem 4.2 (*pruner*2), probability lower bound in Theorem 4.1 (*pruner*3), and probability upper bound in Theorem 4.1 (*pruner*4). Fig. 13 shows the pruning power of each pruner with various r (*missing ratio*=10%, c=0.1, 20 assistant objects). From the figure, we can see the effectiveness of each pruner. We find that for S&P500 data set, about 90% data objects are pruned and only a very small portion of the data objects need to be evaluated by the naive verification process. For IMAGE data set, even in the worst case, there are about 50% data objects can be pruned.

The pruning power of these four pruners is influenced by the threshold r. When specifying a smaller r (i.e., the user wants to get a relatively small amount of query results), more data are pruned by *pruner*1 and *pruner*3. In contrast, a larger distance threshold produces a larger pruning power for *pruner*2 and *pruner*4.

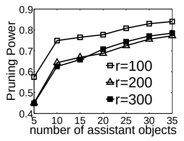
Note that the naive verification step is time consuming. Next, we study whether this step is necessary. We try



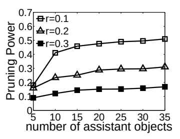
(a) 5% missing ratio (S&P500)



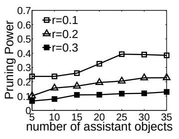
(b) 10% missing ratio (S&P500)



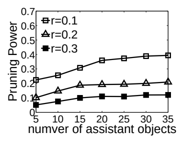
(c) 15% missing ratio (S&P500)



(d) 5% missing ratio (IMAGE)

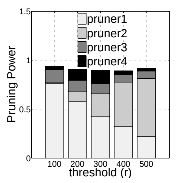


(e) 10% missing ratio (IMAGE)

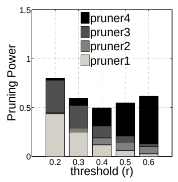


(f) 15% missing ratio (IMAGE)

Fig. 12. Pruning power of probability triangle inequality



(a) S&P500 data set



(b) IMAGE data set

Fig. 13. Pruning power of four pruners

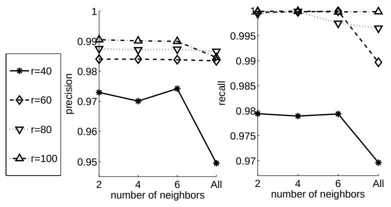
two simplified verification strategies. For data objects that the former four pruners cannot judge, strategy *Pos* simply outputs them as query results. Strategy *Neg*, by contrast, judges them as dismissals. Clearly, *Pos* may result in more false positives, and *Neg* may result in more false negatives. The query quality of these two strategies and the naive verification step (denoted by *DoN*) are shown in Table 2 (c=0.1). From the table, we can see that for S&P500 data set, without naive verification, the precision and recall are very close to those with naive verification. The query quality depends more on the naive verification step for the IMAGE data. We can observe that for the data sets with time series characteristics, such as the S&P500 data set, a good query quality can be achieved even without the naive verification process.

7.2.4 Comparison of Different Imputation Strategies

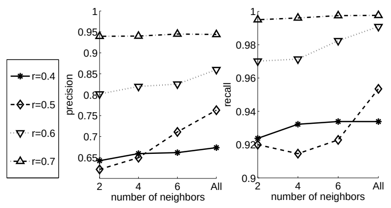
In this section, we study the effect of different imputation strategies. In our algorithm, the expected value of a random variable imputed is the mean value of its nearest two observed neighbors. We refer to this imputation strategy as *imputeNeighbourMean*. This strategy utilizes the local properties of the multidimensional data. For comparison purpose, we apply another imputation strategy that uses the mean value of all observed data values as the expected value of the random variables. We refer to this imputation strategy as *imputeOverallMean*. This strategy utilizes the characteristic of the entire data object. Moreover, we also study the case when the expected value of imputed random variable is the mean of more observed neighbors. Fig. 14 shows the comparison of these imputation strategies. In the figure, we evaluate the precision and recall when varying the number of neighbors used for imputation. The strategy *imputeNeighbourMean* corresponds to the case when "number of neighbors" is 2. Strategy *imputeOverallMean* corresponds to the case "All". Clearly, the more neighbors are used, the more global properties are incorporated. From the figure, we can observe that *imputeNeighbourMean* is more suitable for S&P500 data set. This is reasonable since this strategy captures the local properties of the typical time series data. By contrast, *imputeOverallMean* works better on IMAGE data set. This is because IMAGE data set is not a time series data and lack of continuity and consistency. Thus a global mean is capable of reducing randomness, thus more suitable for IMAGE data. Our framework provides the flexibility to the user to choose the appropriate imputation strategy for specific applications.

TABLE 2
Comparison of query quality

	missing ratio	5%		10%		15%	
		precision	recall	precision	recall	precision	recall
S&P500 (r=40)	*Neg*	0.999	0.948	0.999	0.888	1	0.857
	Pos	0.946	1	0.884	1	0.861	1
	DoN	0.984	0.999	0.964	0.989	0.947	0.979
IMAGE (r=0.4)	*Neg*	1	0.244	0.947	0.053	0.955	0.032
	Pos	0.408	1	0.254	0.998	0.239	1
	DoN	0.787	0.976	0.642	0.923	0.551	0.866

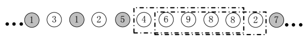


(a) S&P500

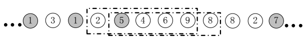


(b) IMAGE

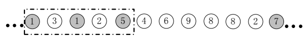
Fig. 14. Search quality comparison with different imputation policies(missing ratio=0.1,c=0.1)



(a)



(b)



(c)

Fig. 15. Subsequence matching evaluation example

7.2.5 Evaluation of Subsequence Matching on Dimension Incomplete Data

In this section, we examine the quality of our framework for subsequence matching on dimension incomplete data. We use the first 5 dimensions of the whole sequence queries as the queries for subsequence matching in S&P500 and IMAGE data sets. The subsequences obtained from the complete data are used as the "ground truth". To make it clear, for instance, in Fig. 15, the original complete sequence is $(...1, 3, 1, 2, 5, 4, 6, 9, 8, 8, 2, 7, ...)$, among which the gray values are observed, and the length of query is 5. As shown in Fig. 15(a), if the algorithm identifies that the subsequence between value 5 and 7 satisfies the query condition, then this is a true positive if and only if $(4, 6, 9, 8, 8)$ or $(6, 9, 8, 8, 2)$ satisfies the distance condition. If the algorithm identifies (5) as the dimension incomplete search result, then this is true if and only if $(2, 5, 4, 6, 9)$ or $(5, 4, 6, 9, 8)$ meets the distance condition. Similarly in Fig. 15(c), if $(1, 1, 5)$ is identified as the search result, it will be a false positive if $(1, 3, 1, 2, 5)$ does not meet the distance condition. Furthermore, if $(5, 7)$ is identified as a search result, then it is definitely a false positive.

Fig. 16 and Fig. 18 show the precision and recall of subsequence matching for S&P500 and IMAGE data sets. Different minimal *length* constraints are used to match the query (c=0.1 for both data sets, r=20 for S&P500, and r=0.4 for IMAGE). This constraint reveals the prior knowledge for the *missing ratio* of dimension incomplete sequences. For larger missing ratio, a smaller value for the constraint of minimal *length* is more appropriate. This constraint also reveals a tradeoff between precision and recall. If the user wants to identify more matched subsequences, a smaller constraint value is better. The results of a naive solution is presented as the baseline. This naive solution neglects the fact of dimension incompleteness, and treats data objects as complete examples. The results demonstrate the superiority of our framework for tackling subsequence matching on dimension incomplete data.

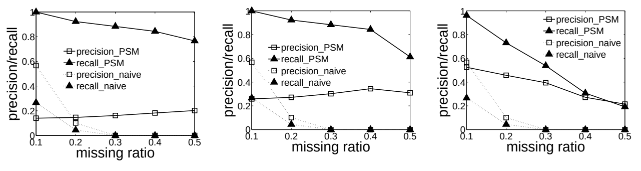
The results also show that, with the increase of minimal *length* constraint, precision and recall are more sensitive to the *missing ratio*. When the minimal *length* constraint is small (e.g., *minimal length*$\leq$ 2), the precision and recall remain stable with the increase of *missing ratio*. Fig. 17(a) and Fig. 17(b) present the evolution of precision and recall, as a function of the minimal *length* constraint (*missing ratio*=0.25 and c=0.1 for both data sets, r=20 for S&P500, and r=0.4 for IMAGE). From the figures, we can see that the constraint value of *length* provides an effective tradeoff between precision and recall. With the increase of minimal *length* constraint, the precision increases, and the recall decreases. When the minimal *length* constraint arrives at 5, the algorithm degenerates into the naive solution in that the query is of dimensionality 5.

7.2.6 Performance for Data with Different Dimensionalities

In this section, we study the effect of dimensionality of the dataset on pruning power and the search quality. The original S&P500 dataset has 250 dimensions. It enables to vary the dimensionality in a wide range. When the dimensionality is large, the naive evaluation step is intractable. We use strategy *Neg* and simply judge them as dismissals. The distance threshold r is set to the value so that half of the data objects satisfy the query condition. The probability threshold c is set to 0.2, and the missing ratio is set to 0.2. Due to using strategy *Neg*, the precision values for different dimensionalities are all 1. The recall values are shown in Fig. 19(a). The pruning power of all four pruners (including probability bounds and probability triangle inequality) are shown in Fig. 19(b). From the figures, we observe that both recall and pruning power decrease as the dimensionality of data increase. This is because for a give missing ratio, with the increase of the dimensionality, the uncertainty of the data object increases. In other words, there are more possible recovery versions for the dimension incomplete data. This causes the probability bounds become looser, together with the decline of the search quality. Nevertheless, we still observed that even when the dimensionality reaches 250, our algorithm achieves more than 0.84 in terms of recall and more than 0.88 in terms of the pruning power. This well illustrates the effectiveness of our algorithm for tackling high-dimensional time series data. Another indicator of the performance is the running time. We will study the running time of the algorithm when varying the dimensionality of the data in Section 7.2.7.

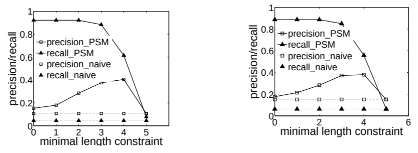
7.2.7 Runtime Evaluation

There are three major steps in our approach for the whole sequence matching: (1) pruning with probability triangle



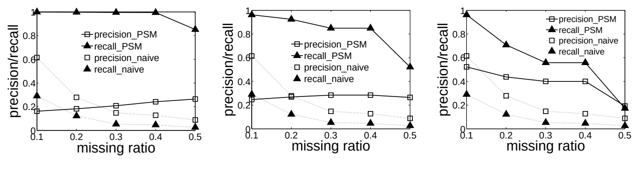
(a) minimal length = 0 (b) minimal length = 2 (c) minimal length = 4

Fig. 16. Missing ratio vs precision/recall on S&P500



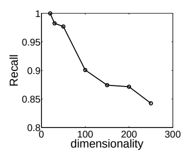
(a) S&P500 (b) IMAGE

Fig. 17. Minimal length constraint vs precision/recall

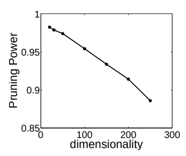


(a) minimal length = 0 (b) minimal length = 2 (c) minimal length = 4

Fig. 18. Missing ratio vs precision/recall on IMAGE

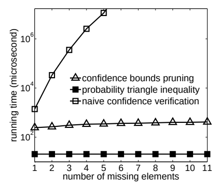


(a) Recall

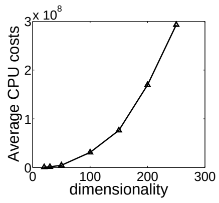


(b) Pruning Power

Fig. 19. Recall/Pruning power vs dimensionality on S&P500

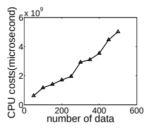


(a) Running time on each step of whole matching

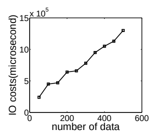


(b) Average time costs vs dimensionality

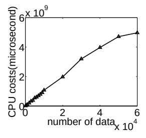
Fig. 20. Time Complexity Evaluation



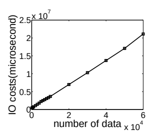
(a) CPU costs for S&P500 data set



(b) IO costs for S&P500 data set

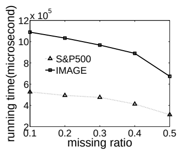


(c) CPU costs for IMAGE data set

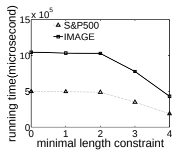


(d) IO costs for IMAGE data set

Fig. 21. CPU/IO costs vs size of dataset



(a) missing ratio



(b) minimal length constraint

Fig. 22. Running time of subsequence matching vs missing ratio/minimal length constraint

inequality; (2) pruning with probability lower and upper bounds; (3) naive probability verification. We examine running time of these steps using the S&P500 data set on a computer with 3.0GHz CPU and 1.0GB RAM. The running time is averaged over all queries. The results are shown in Fig. 20(a). We can see that the first two steps are very efficient compared with the naive probability verification step. Moreover, Table 2 in section 7.2.3 demonstrates that this naive probability verification is not necessary for the data sets with time series characteristics. Thus the efficiency of the overall query process can be dramatically improved without compromising the quality of the results.

To study the scalability of the proposed algorithms, we limit the usage of the maximal memory for the proposed algorithms to 10MB, and compare the CPU and IO costs separately when the size of dataset changes (from 20KB to 100MB). We use the original S&P500 data (containing 500 data objects with dimensionality 250) and IMAGE data. Since when the dimensionality is large, the naive evaluation step is intractable, we use strategy *Neg* and simply judge them as dismissals. When varying the number of data objects, the total running time for randomly selected 20 queries is shown in Fig. 21. From the figure, we observe that both CPU and IO costs on the two datasets grow approximately linearly with the increase of the size of dataset. This is reasonable because our algorithm dose not need to fit all the data objects and queries in the memory. It only needs to fit one query and one dimension incomplete data object to be verified in the memory. Thus, the running time (both CPU and IO costs) for the whole searching process grows linearly when increasing the size of dataset. In addition, due to the fact that the time complexity of our algorithm mainly depends on the dimensionality of data, we also study the average query time cost when varying the dimensionality with a given missing ratio. The results in Fig. 20(b) show that the average time cost for each query processing has an approximately cubic growth with respect to the dimensionality of data. This is due to the fact that the time complexity is $O(|X_o| \cdot (|Q| - |X_o|)^2)$. When fixing the missing ratio, it is identical to $O(|Q|^3)$.

We also measured the running time of subsequence matching using different missing ratios and minimal length constraints. Fig. 22(a) shows that for both data sets, the running time declines with the increase of the missing ratio when the minimal length constraint is fixed (minimal length constraint=0, c=0.1, r=20 for S&P500, r=0.4 for IMAGE). From Section 5, we know that this is because the complexity of subsequence matching is proportional to $|X_o|$. With the increase of the missing ratio, $|X_o|$ decreases. Fig. 22(b) presents the relationship between the running time of subsequence matching and the minimal length constraint when the missing ratio is fixed (missing ratio=0.2). This figure shows that with larger minimal length constraint, running time can be effectively reduced. This is because a larger minimal length constraint enables the algorithm to neglect more data missing cases. Recall that in Section 5, we find that the minimal length constraint provides an effective trade-off among precision, recall and time complexity.

8 Conclusions

This paper addresses the similarity query problem on dimension incomplete data, which is of both practical importance and technical challenge. A probability framework is proposed to model this problem. To solve this problem efficiently, we develop the lower and upper probability bounds and the probability triangle inequality that can be used to dramatically prune the search space. Furthermore, the similarity query framework is extended to tackle subsequence matching in dimension incomplete data. For a query Q and a dimension incomplete data object X_o, the brute force method is of complexity $O(|Q| \cdot \binom{|Q|}{|X_o|})$. Our method achieves a significant improvement: most data objects can be handled in $O(|X_o| \cdot (|Q| - |X_o|)^2)$ or even $O(|Q|)$ time.

We conduct extensive experimental evaluation using real data sets. The results indicate that (1) our approach achieves satisfactory performance in querying dimension incomplete data for both whole sequence matching and subsequence matching; (2) both the probability triangle inequality and the probability bounds have a good pruning power and improve query efficiency significantly;

Our future work will focus on the following directions. Since a probability triangle inequality holds, we plan to develop an index structure that can utilize the inequality to further improve the efficiency of the query process. Further more, we plan to investigate how to extend our query strategy to incorporate a wide range of distance functions.

Acknowledgments

The work was partially supported by National Natural Science Foundation of China (60973103, 90924003), CAS/SAFEA International Partnership Program 70921061, and US National Science Foundation IIS0448392, I-IS0812464.

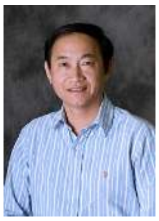
References

[1] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proceedings of SIGMOD'94*. ACM Press, 1994, pp. 419–429.
[2] M. Ankerst, B. Braunmller, H.-P. Kriegel, and T. Seidl, "Improving adaptable similarity query processing by using approximations." in *Proceedings of VLDB'98*, 1998, pp. 206–217.
[3] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient Similarity Search In Sequence Databases," in *Proceedings of FODO'93*, 1993, pp. 69–84.
[4] R. Fagin, R. Kumar, and D. Sivakumar, "Efficient similarity search and classification via rank aggregation." in *Proceedings of SIGMOD'03*, 2003, pp. 301–312.
[5] C. C. Aggarwal and S. Parthasarathy, "Mining massively incomplete data sets by conceptual reconstruction," in *Proceedings ACM SIGKDD'01*, 2001, pp. 227–232.
[6] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan, "Olap over uncertain and imprecise data," in *Proceedings of VLDB'05*, 2005, pp. 970–981.
[7] G. Canahuate, M. Gibas, and H. Ferhatosmanoglu, "Indexing incomplete database," in *Proceedings of EDBT'06*, 2006, pp. 884–901.
[8] J. Gu and X. Jin, "Similarity search over incomplete symbolic sequences," in *Proceedings of DEXA'07*, 2007, pp. 339–348.
[9] H. Zhang, Y. Diao, and N. Immerman, "Recognizing patterns in streams with imprecise timestamps," in *Proceedings of VLDB'10*, vol. 3, 2010, pp. 244–255.
[10] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *Proceedings of ACM SIGMOD'03*, 2003, pp. 551–562.
[11] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: a probabilistic threshold approach," in *Proceedings of SIGMOD'08*, 2008, pp. 673–686.
[12] E. Keogh and M. Pazzani, "Scaling up dynamic time warping to massive datasets," in *Proceedings of ECML/PKDD'99*, 1999, pp. 1–11.
[13] B. Bollobas, G. Das, D. Gunopulos, and H. Mannila, "Time-series similarity problems and well-separated geometric sets," in *Proceedings of SCG'97*, 1997, pp. 454–456.
[14] D. Gu and Y. Gao, "Incremental gradient descent imputation method for missing data in learning classifier systems," in *Proceedings of GECCO'05*, 2005, pp. 72–73.
[15] R. K. Pearson, "The problem of disguised missing data," *ACM SIGKDD Explorations Newsletter*, pp. 83–92, 2006.
[16] I. Wasito and B.Mirkin, "Nearest neighbour approach in the least-squares data imputation algorithms," *Information Sciences: an International Journal*, vol. 169, pp. 1–25, 2005.
[17] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proceedings of VLDB'07*, 2007, pp. 15–26.
[18] J. Pei, M. Hua, Y. Tao, and X. Lin, "Query answering techniques on uncertain and probabilistic data: tutorial summary," in *Proceedings of ACM SIGMOD'08*, 2008, pp. 1357–1364.
[19] E. Keogh, "Exact indexing of dynamic time warping," in *Proceedings of VLDB'02*, 2002, pp. 406–417.
[20] G. Navarro, "A guided tour to approximate string matching," vol. 33, 2001, pp. 31–88.
[21] R. A. Little and D. B. Rubin, "Statistical analysis with missing data," *Wiley Series in Probability and Statistics,1st*, pp. 2–278, 1987.
[22] T. Mathew and K. Nordstrom, "Inequalities for the probability content of a rotated ellipse and related stochastic domination results," *The Annals of Applied Probability*, vol. Vol. 7, No. 4, pp. 1106–1117, 1997.

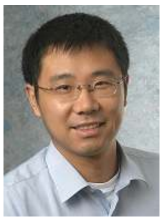
Wei Cheng received the bachelor's degree from Nanjing University in 2006 and the master's degree from Tsinghua University in 2010, both in software engineering. He is currently working toward the PhD degree in computer science at UNC at Chapel Hill. His research interests are data mining, machine learning and bioinformatics.

Xiaoming Jin received the doctor's degree in computer science from Tsinghua University in 2003. He is an associate professor in the School of Software, Tsinghua University. His research interests are data mining and machine learning.

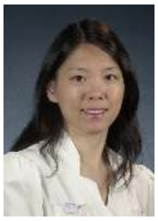
Jian-Tao Sun received the doctor's degree in computer science from Tsinghua University. He is a lead researcher at Machine Learning group, Microsoft Research Asia. His research interests is machine learning, and information retrieval.



Xuemin Lin received the PhD degree in computer science from the University of Queensland in 1992. He is a professor in the School of Computer Science and Engineering at the University of New South Wales (UNSW). His current research interests include data streams, approximate query processing, spatial data analysis, and graph visualization.



Xiang Zhang received the doctor's degree from UNC at Chapel Hill in 2011. He is an assistant professor in the Electrical Engineering and Computer Science Department at Case Western Reserve University. His research interests include data mining, bioinformatics, and databases.



Wei Wang received the doctor's degree in computer science from University of California at Los Angeles in 1999. She is a professor in University of California at Los Angeles. Her research interests include data mining, bioinformatics, and databases.