Top-k Nearest Neighbor Search In Uncertain Data Series

Michele Dallachiesa
University of Trento
dallachiesa@disi.unitn.it

Themis Palpanas
Paris Descartes University
themis@mi.parisdescartes.fr

Ihab F. Ilyas
University of Waterloo
ilyas@uwaterloo.ca

ABSTRACT

Many real applications consume data that is intrinsically uncertain, noisy and error-prone. In this study, we investigate the problem of finding the top-k nearest neighbors in uncertain data series, which occur in several different domains. We formalize the top-k nearest neighbor problem for uncertain data series, and describe a model for uncertain data series that captures both uncertainty and correlation. This distinguishes our approach from prior work that compromises the accuracy of the model by assuming independence of the value distribution at neighboring time-stamps. We introduce the *Holistic-PkNN* algorithm, which uses novel metric bounds for uncertain series and an efficient refinement strategy to reduce the overall number of required probability estimates. We evaluate our proposal under a variety of settings using a combination of synthetic and 45 real datasets from diverse domains. The results demonstrate the significant advantages of the proposed approach.

1. INTRODUCTION

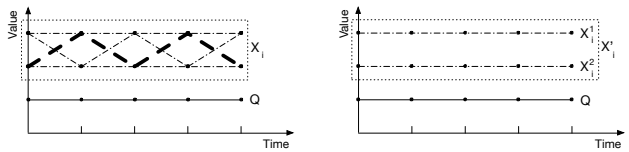
In recent years, the database and data mining community has investigated extensively the problem of modeling and querying uncertain data [3, 15], and several probabilistic database systems have been proposed [16, 6, 28]. Uncertainty can occur for different reasons, including imprecision in the sensor measurements, approximations due to summarization techniques ,privacy-preserving transformations of sensitive records and the limited confidence in the output of predictive models.

The problem of identifying the top-k nearest neighbors has been widely studied in traditional database systems [17], and has been successfully used to implement classifiers, recommendation engines and location-based services.

Similarly, the evaluation of top-k nearest queries has received considerable attention in probabilistic databases [12, 26, 20, 29, 8, 21, 11]. In this study, we consider the problem of finding the top-k nearest neighbors in uncertain data series, i.e., in ordered sequences of values that are uncertain[1]. Some examples are as follows:

[1] *Time series* are a special case of data series, where the values are measured over time, but a series can also be defined over other measures (e.g., mass in Mass Spectroscopy, etc.).

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/. Obtain permission prior to any use beyond those covered by the license. Contact copyright holder by emailing info@vldb.org. Articles from this volume were invited to present their results at the 41st International Conference on Very Large Data Bases, August 31st - September 4th 2015, Kohala Coast, Hawaii.
Proceedings of the VLDB Endowment, Vol. 8, No. 1
Copyright 2014 VLDB Endowment 2150-8097/14/09.



(a) (b)

Figure 1: (a) samples from independent random variables, (b) samples from full-joint distribution.

- The variations of the soil moisture during one year can be modeled by multiple measurements collected from different nearby field locations at a regular time interval. An uncertain data series can be used to model these sensor measurements to represent the uncertainty of the soil moisture within the investigated area. The type of cultivation can be assigned as class label to the uncertain data series. Nearest neighbor searches on a dataset of uncertain data series representing the soil moisture at different sites can then be used to recommend the type of cultivation in new farm lands.
- Trajectories are intrinsically noisy because of imprecisions in the positioning system. In addition, trajectories can be anonymized due to privacy concerns. Uncertain data series can be used to model the unknown original trajectory by enumerating all the feasible modifications consistent with a recovery procedure, i.e., the possible worlds for the original trajectory. One can then identify which are the individuals that most likely went to the pharmacy after visiting the hospital using nearest neighbor searches.

Modeling correlation and uncertainty is critical to produce meaningful results. We illustrate an example in Figure 1, where we show how two different representations of the same uncertain series can lead to different results. Figure 1(a) shows an uncertain series modeled by five independent random variables (X_i). The same uncertain series is represented in Figure 1(b) by a single multivariate random variable (X'_i). In contrast to the independent random variables in X_i, the single multivariate random variable X'_i can model the dependencies among neighboring time-stamps. Two real-valued samples represent the value distribution at each time-stamp in Figure 1(a) and two series samples are drawn from the full-joint distribution of X'_i in Figure 1(b). Despite having the same point values, the two representations of uncertain series X_i and X'_i have a different distance distribution to the query Q. For example, the distance between the highlighted sample (thick dashed line) in X_i and Q has zero occurring probability under the X'_i representation. In summary, proper modeling of correlation leads to more

accurate results, while removing uncertainty using averaged values may lead to erroneous conclusions. For example, the distance between Q and the average series obtained averaging the values of X_i and X_i' is the same even if their distance distributions differ.

The complexity introduced by correlation and uncertainty poses new challenges for the efficient evaluation of top-k nearest neighbor searches in uncertain data series. Prior studies assumed the value distributions at neighboring time-stamps to be independent as a simplifying assumption. However, correlation of neighboring values is an important property of data series, and ignoring it can lead to erroneous results as demonstrated in Figure 1. Moreover, prior approaches base their efficiency on the iterative refinement of the spatial regions that bound the uncertainty. However, this is not an effective strategy with uncertain data series as their bounding regions overlap with high probability, especially for series with normalized values.

Pioneering studies on modeling and querying uncertain data series include DUST [27], PROUD [31] and MUNICH [7], and have been experimentally and analytically compared in [14]. Figure 2 positions these studies and our proposal, the *Holistic-PkNN* algorithm, by their underlying uncertainty model on the x-axis and by their ability to represent correlation on the y-axis. The methods

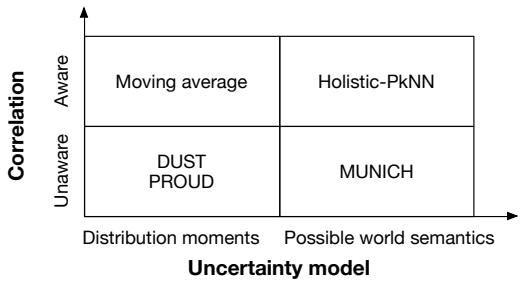


Figure 2: Mapping of relevant studies, clustered along two dimensions: the underlying uncertainty model, and their ability to retain the correlation of the value distributions of neighboring points.

that consider distribution moments as underlying uncertainty model represent uncertainty using statistics of the series distribution such as the mean, variance and skewness. A different approach consists in the "possible world" semantics, where uncertainty is quantified by a set of possible instantiations, i.e., samples drawn from the series distribution. The techniques reported in the bottom quadrants model uncertain data series as independent sequences of random variables. Independence is a simplifying assumption, and can lead to erroneous results. The moving average [14] can be adapted to leverage the correlation across neighboring points (top-left quadrant). However, moving averages may lead to misleading results.

In this study, we introduce the *Holistic-PkNN* algorithm, which represents uncertain data series using full-joint distributions and retains the correlation information. It adopts a "possible world" semantics model, with an uncertain series being represented by a set of sample series drawn from its full-joint distribution, thus enabling the modeling of both uncertainty and correlation. The algorithm starts by estimating the distance bounds between the query and the uncertain series using novel metric bounds, specifically suited to uncertain series. The distance bounds are refined incrementally using a selection strategy that tightens the probability bounds using a small number of iterations and probability estimate computations.

In summary, the contributions of this paper are as follows.

- We formally define the problem of top-k nearest neighbor queries in uncertain data series, and we introduce a representation specific for uncertain data series, which retains the information in the *sequence* of the uncertain series values.
- We propose an iterative algorithm for the evaluation of top-k nearest neighbor queries, which can efficiently refine the candidate result set, leading to a significantly reduced number of probability estimations during the incremental refinement of the probability bounds.
- We further introduce novel metric-space bounds for uncertain series, which are much tighter when compared to traditional solutions. Based on those, we describe an efficient method for the retrieval of the candidate series, using a metric index.
- We perform an extensive experimental evaluation, using 45 real datasets from diverse domains and synthetic datasets. The results demonstrate the effectiveness of the proposed techniques and serve as guidelines for the practitioners.

This paper is organized as follows. In Section 2, we discuss different alternative models of uncertain data series and formally define our problem. In Section 3, we present our proposal. In Section 4, we discuss the experimental results. In Section 5, we survey prior studies and we conclude in Section 6.

2. PROBLEM DEFINITION

In this section, we formalize the problem after introducing some definitions. A dataset D is a set of N uncertain data series. Similarly to prior works [12, 8], we further assume that the uncertain series in D are independent, i.e., the samples drawn from uncertain series X_i are independent of the samples drawn from uncertain series $X_j, \forall i \neq j$.

A data series[2] S is an ordered sequence of n real valued numbers $S = S[t]$, $1 \leq t \leq n$. An uncertain data series X is a data series whose values at each time-stamp are uncertain. We adopt the attribute-uncertainty model under the "possible world" semantics to represent uncertain series.

A possible instantiation represents a sample value drawn from the full joint distribution of the uncertain series. We denote the value at time-stamp t of sample j of uncertain series X_i as $X_i^j[t]$. The index i may be omitted for ease of exposition. We formalize the *uncertain data series* model as follows:

DEFINITION 2.1 (UNCERTAIN DATA SERIES MODEL). *An uncertain series X of length n is represented by m series samples. A series sample X^j is a sample drawn from the full joint distribution of X. Formally, $X = \{X^j : 1 \leq j \leq m\}$.*

An example of uncertain series represented under the *uncertain data series* model is shown in Figure 1(b). As discussed in the introduction, the *uncertain data series* model can represent uncertainty and correlation more accurately than other models that fail to capture the correlation of the value distributions at neighboring time-stamps.

The distance between uncertain series X and query Q is uncertain and its distribution is induced by the distance measure. In particular, the distance samples are obtained by evaluating the distance measure between the series instantiations X^j as defined under the *uncertain data series* model and Q:

DEFINITION 2.2 (SAMPLE DISTANCE DISTRIBUTION). *The sample distance distribution between uncertain series X and query*

[2]In the rest of this paper we use the terms *data series*, *time series*, *series* and *sequence*, interchangeably.

series Q is denoted by $Dist(X, Q)$ and is defined as $Dist(X, Q) = \{dist(X^j, Q) : j \in 1, ..., m\}$, where $dist(X^j, Q)$ is the distance measure between Q and the jth instance of X.

We consider the Euclidean distance as reference implementation of the $dist(\cdot)$ function, however other metric distance measures can be considered as well. Table 1 summarizes the most important symbols used in the rest of the paper.

Now we are ready to formulate the problem of top-k nearest neighbor search in uncertain data series. Let D be a dataset of N uncertain series represented by the *uncertain data series* model with series length n and number of samples m. Given a query series Q, the probability of an uncertain data series X_i to be the NN (Nearest Neighbor), denoted by $P_{NN}(Q, X_i)$, is: $P_{NN}(Q, X_i) =$

$$\int Pr\left(Dist(Q, X_i) = s \wedge \bigwedge_{\forall j \neq i} Dist(X_j, Q) > s\right) ds \quad (1)$$

Let $r(i)$ be a rank function s.t. $r(i) \leq r(j)$ iff $P_{NN}(Q, X_i) \geq P_{NN}(X_j, Q)$. We can now introduce the definition of the top-k probable nearest neighbors:

PROBLEM 2.1 (TOP-k PROBABLE NEAREST NEIGHBORS). *Given a dataset D and query Q, the top-k probable nearest neighbor search Top-k-$P_{NN}(D, Q, k)$ returns the k uncertain series X_i with the largest $P_{NN}(Q, X_i)$ probabilities. Formally, the result set is defined as $\{X_{r(1)}, ..., X_{r(k)}\}$.*

We observe that this problem is computationally bound by the evaluation of the $P_{NN}(Q, X_i)$ probabilities. To this effect, coarse representations of the *sample distance distributions* are used to refine incrementally the probability bounds. The selection of the refinements is crucial, and can affect significantly the time performance [12, 20, 8, 11]. In this study, we propose a novel selection strategy that can tighten the probability bounds using a smaller number of iterations, thus reducing the overall number of evaluations of the probability bounds. The efficient retrieval of the candidates and the initialization of the probability bounds is implemented using novel metric bounds, specifically suited to uncertain series.

3. PROPOSED APPROACH

In this section, we present our proposal for the efficient evaluation of Top-k-$P_{NN}(D, Q, k)$ queries. We present a baseline algorithm and its computational complexity in Section 3.1. An approach that uses coarse representations of the distance samples to obtain estimates of the PNN probability bounds is reported in Section 3.2. We introduce the *Holistic-PkNN* algorithm in Section 3.3. The procedure uses the incremental refinement of the distance distributions to determine the result efficiently. The holistic selection of the refinements is discussed in Section 3.4. In Section 3.5 we present different algorithms to prune the search space and to initialize the distance distributions.

Notation	Description
Q	Query series
k	Result set size
n	Series length
m	Number of uncertain series samples
D	Dataset of uncertain series
N	Number of uncertain series in dataset D
X_i	ith uncertain series
X_i^l	lth sample of ith uncertain series
$X_i^l[t]$	lth sample of X_i at time-stamp t
$B_i = [B_i^{lb}, B_i^{ub}]$	PNN bounds for uncertain series X_i
$S_i = \{S_i^l\}$	distance partition for X_i
S_i^l	lth distance interval in S_i
W_i^l	Weight of lth distance interval in S_i
$I(X)$	1 if X is true, 0 otherwise

Table 1: Notation used in the paper.

3.1 Baseline Algorithm

In this section, we present the baseline algorithm to evaluate top-k probable nearest neighbor queries under the *uncertain data series* model. With the independence assumption among the uncertain series $X_i \in D$, Eq. 1 can be simplified to: $P_{NN}(Q, X_i) =$

$$\int Pr(Dist(Q, X_i) = s) \prod_{\forall j \neq i} Pr(Dist(X_j, Q) > s) ds \quad (2)$$

where the Probability Density Function (PDF), $Pr(Dist(Q, X_i) = s)$, is essentially used to weight the second term and the inverse of the Cumulative Density Function (CDF), $Pr(Dist(X_j, Q) > s)$, is estimated as the ratio of samples matching the inequality condition:

$$Pr(Dist(X_j, Q) > s) = \frac{1}{|\{r \in Dist(X_j, Q) : r > s\}|}$$

For ease of exposition, we introduce the $I(X)$ indicator function: $I(X)$ evaluates to 1 if the condition X holds, 0 otherwise. The evaluation of Eq. 2 can then be reduced as follows: $P_{NN}(Q, X_i) =$

$$\frac{1}{m^N} \sum_{s \in Dist(Q, X_i)} \left[\prod_{\forall j \neq i} \left(\sum_{r \in Dist(X_j, Q)} I(r > s)\right)\right] \quad (3)$$

where $1/(m^N)$ results from further simplifications of the $Dist(Q, X_i) = s$ terms in Eq.2, and $Dist(\cdot)$ is a set of distance samples previously defined in Definition 2.2. The Top-k-$P_{NN}(D, Q, k)$ queries are evaluated by determining the k uncertain series with the largest PNN probabilities, where the $P_{NN}(Q, X_i)$ estimates are determined using Eq. 3 and the selection algorithm is used to identify the top-k uncertain series [19].

The evaluation of Eq. 3 has a CPU cost of $O(Nm^2)$, thus the CPU cost sums up to $O(m^2N^2)$ if we consider all $X_i \in D$. The time complexity of the selection algorithm is linear to the size of the array (on average) and is bounded by $O(N)$. We note that the evaluation of $P_{NN}(Q, X_i)$ dominates the CPU cost.

3.2 Bounding PNN probability estimates

The evaluation of the PNN probabilities is based on the distance measurements between the query series Q and the uncertain series X_i (Eq. 2.2). Once the distance samples have been determined, the raw series X_i are not accessed anymore.

In the following we show how coarse representations of the distance samples $Dist(Q, X_i)$ can be used to determine the bounds for the PNN probability estimates for candidate X_i. Candidates can be either discarded or added to the result set according to the probability bounds without considering the individual distance samples, a rather expensive operation due to the high number of possible sample combinations in Eq. 3.

DEFINITION 3.1 (DISTANCE INTERVAL). *A distance interval S_i^l is a region in the distance space that represents a subset of the distance samples in $Dist(Q, X_i)$. The lower and upper bounds of S_i^l are denoted by $lb(S_i^l)$ and $ub(S_i^l)$, respectively. The associated weight is denoted by W_i^l and is defined as the ratio of samples in*

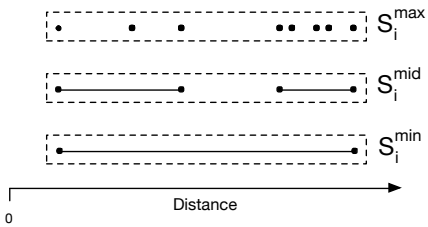


Figure 3: Example of valid distance partition instantiations S_i^{min}, S_i^{mid} and S_i^{max} representing the distance samples in $Dist(Q, X_i)$.

$Dist(Q, X_i)$ falling within the interval bounds. Any distance interval S_i^l represents at least a distance sample, i.e., $W_i^l > 0$. A distance partition S_i is a set of disjoint distance intervals S_i^l that partition the samples in $Dist(Q, X_i)$.

Given a distance partition S_i, these two properties hold: first, distance intervals S_i^l and S_i^k do not overlap, $\forall l \neq k$. Second, the sum of the weights of the distance intervals in the S_i partition equals to one, i.e. $\sum_l W_i^l = 1$.

Distance partitions at different levels of detail can be instantiated to represent the samples in $Dist(Q, X_i)$. The number of distance intervals (denoted by d) in the partitions ranges between 1 and m. We denote with S_i^{min} the partition instantiation composed by a single distance interval (coarsest level, $d = 1$). S_i^{max} denotes the partition instantiation composed by m distinct distance intervals (finest level, $d = m$). An example of S_i^{min}, S_i^{max} and an intermediate valid distance partition instantiation denoted by S_i^{mid} is reported in Figure 3. Distance partitions can be used to estimate lower- and upper-bounds of the PNN probabilities, denoted by $P_{NN}^{lb}(Q, X_i)$ and $P_{NN}^{ub}(Q, X_i)$, respectively.

The lower-bound $P_{NN}^{lb}(Q, X_i)$ is determined by using the upper-bounds $ub(S_i^l)$ as representatives of the distance intervals in partition S_i and the lower-bounds $lb(S_j^l)$ as representatives of the distance intervals in the other partitions S_j. The resulting adaptation of Eq.3 is: $P_{NN}^{lb}(Q, X_i) =$

$$\sum_{S_i^a \in S_i} \left[W_i^a \prod_{\forall j \neq i} \left(\sum_{S_j^b \in S_j} W_j^b \cdot I(ub(S_i^a) < lb(S_j^b)) \right) \right] \quad (4)$$

Similarly, the upper-bound $P_{NN}^{ub}(Q, X_i)$ is determined by using the lower-bounds $lb(S_i^l)$ as representatives of the distance intervals in partition S_i and the upper-bounds $ub(S_j^l)$ as representatives of the distance intervals in the other partitions S_j. The resulting adaptation of Eq.3 is: $P_{NN}^{ub}(Q, X_i) =$

$$\sum_{S_i^a \in S_i} \left[W_i^a \prod_{\forall j \neq i} \left(\sum_{S_j^b \in S_j} W_j^b \cdot I(lb(S_i^a) < ub(S_j^b)) \right) \right] \quad (5)$$

We denote the probability interval identified by the lower- and upper-bounds of the PNN probability estimates by B_i:

$$B_i = [P_{NN}^{lb}(Q, X_i), P_{NN}^{ub}(Q, X_i)]$$

We observe that if we use the finest representation of the distance samples S_i^{max} for all uncertain series $X_i \in D$, then $P_{NN}^{lb}(Q, X_i)$ and $P_{NN}^{ub}(Q, X_i)$ can be reduced to Eq.3 and degenerate to the same value, $P_{NN}(Q, X_i)$.

The CPU cost of determining the PNN probability bounds using Eq.4 and Eq.5 is bounded by the number of distance intervals in each partition. A low number of distance intervals in each partition is to be preferred. However, PNN bounds might not be tight enough to discriminate the answer set and a more fine-grained representation of the distance partitions may be required to improve sufficiently the PNN bounds. In the next section, we introduce the *Holistic-PkNN* algorithm that solves efficiently this problem.

3.3 The Holistic-PkNN algorithm

In this section we present *Holistic-PkNN*, an iterative algorithm to evaluate Top-k-$P_{NN}(D, Q, k)$ queries as defined in Section 2. The *Holistic-PkNN* algorithm uses the PNN probability bounds B_i as defined in Section 3.2 and refines incrementally the distance partitions S_i until convergence at a reduced CPU cost.

After pruning the search space using the coarsest representations of the distance intervals, the procedure refines iteratively the distance partitions of the candidates whose PNN probability bounds overlap with the top-k probability bounds. The algorithm terminates when all candidates are either added to the result set or pruned out.

Let $top_k(V)$ be the kth largest value in a set V of values, not necessarily distinct. Let B^{lb} and B^{ub} be the set of PNN probability lower-bounds and the set of PNN probability upper-bounds, respectively. The critical region $[c, d]$ is defined as follows:

DEFINITION 3.2 (CRITICAL REGION). *Let $c = top_k(B^{lb})$ and $d = top_{k+1}(B^{ub})$. The critical region R is defined as the probability interval $R = [c, d]$. The critical region is empty if $c > d$.*

Uncertain series X_i whose PNN probability upper bound $P_{NN}^{ub}(Q, X_i)$ is lower than the lower bound c of the critical region R have zero probability of being the NN, and can be safely pruned. On the contrary, uncertain series X_i whose PNN probability lower bound $P_{NN}^{lb}(Q, X_i)$ is higher than the upper bound d of the critical region R can be safely appended to the result set. We can now formally introduce the set of the active candidates:

DEFINITION 3.3 (ACTIVE CANDIDATES). *Uncertain series X_i is an active candidate iff its PNN probability interval B_i overlaps with the critical region R. We denote with X^* the set of the uncertain series that qualify as active candidates.*

The set of the active candidates X^* identifies the uncertain series X_i that can be eligible to enter the result set but require tighter PNN probability bounds to make a final decision. Note that, if $c > d$, then the critical region R is empty, and the set of active candidates X^* is empty. This condition is encountered when the PNN probability bounds are sufficiently tight to discriminate the result set without further refinements.

Figure 4(a) shows an example of a critical region (R) for $k = 2$. The uncertain series corresponding to PNN interval B_4 is clearly part of the result set since it dominates all the other PNN bounds. However, PNN bounds B_3 and B_2 overlap and we cannot discriminate between them. Figure 4(b) shows an example of a critical region for $k = 2$, s.t. PNN probability bounds B_4 and B_3 dominate all the other PNN bounds and there is no uncertainty on the memberships of the result set. PNN probability bounds of uncertain series $X_i \in X^*$ are tightened by refining some of the distance partitions. A partition is refined by increasing the number of its distance intervals. We observe that the underlying relationships between different candidates can lead to tighter PNN bounds for candidate X_i after refining a distance interval in partition S_j, where $i \neq j$.

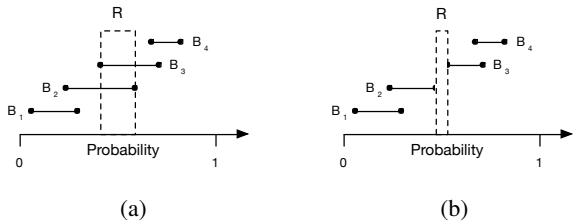


Figure 4: Graph (a) shows an example of critical region R with overlapping PNN probability bounds B_3 and B_2. Graph (b) reports an example of empty critical region R.

Let S^* be the set of the distance partitions $S_1, ..., S_N$. Let B^* be the set of the PNN probability bounds $B_1, ..., B_N$. The overall *Holistic-PkNN* procedure that refines selectively the distance partitions until convergence of the result set is illustrated in Algorithm 1. Line 1 initializes the distance partitions S_i for all $X_i \in D$ to their

Algorithm 1 Holistic-PkNN(D: dataset, Q: query sequence, k: result set size).

1: $S_i \leftarrow$ init-distance-partitions(Q, X_i) for all candidates $X_i \in D$
2: $B_i \leftarrow P_{NN}(Q, X_i)$ bounds using Eq. 4 and Eq. 5 for all candidates $X_i \in D$
3: Update critical region $[c, d]$ using Def. 3.2
4: **while** $c \leq d$ **do**
5: $C \leftarrow$ find-critical$(B^*, [c, d])$
6: $R \leftarrow \bigcup_{X_i \in C}$ find-splits(S_i)
7: **for** $S_i^l \in R$ **do**
8: $S_i \leftarrow$dist-refine(S_i^l)
9: **end for**
10: $B_i \leftarrow P_{NN}(Q, X_i)$ bounds using Eq. 4 and Eq. 5 for all active candidates $X_i \in X^*$
11: Update critical region $[c, d]$ using Def. 3.2
12: **end while**
13: $T \leftarrow \{X_i : P_{NN}^{lb}(Q, X_i) > d\}$

respective S_i^{min} instances. Efficient algorithms to determine the S_i^{min} instances are presented in Section 3.5.1. In Line 2, lower- and upper-bounds B_i for all $X_i \in D$ are initialized using Eq. 4 and Eq. 5. The bounds c, d of the critical region R are updated in Line 3 using Definition 3.2. This concludes the initialization of the data structures before the iterative refinement of the PNN probability bounds. The PNN bounds are then tightened iteratively in Lines 4-12 until convergence of the result set, i.e., the termination condition $c > d$ is met. In each iteration a set of PNN bounds to be tightened is selected (set C) and the distance intervals expected to improve the selected PNN bounds are refined (set R). The PNN probability bounds of the candidates $X_i \in X^*$ and the critical region $[c, d]$ are updated at the end of each iteration. Line 5 identifies the uncertain series $X_i \in C$ whose PNN bounds are selected for improvement. The implementation of the *find-critical* procedure is presented in Section 3.4. In Line 6 the distance intervals S_i^l to be refined are identified, and then refined in Lines 7-9. The efficient evaluation of the refinements is discussed in Section 3.5.2. Line 10 updates the PNN probability bounds B_i of the active candidates $X_i \in X^*$. The bounds c, d of critical region R are then updated in Line 11. Finally, the result set is constructed in Line 13.

We note that the *find-splits* procedure may return the same distance interval S_i^l to tighten the PNN probability bounds of different candidates. However, the distance interval S_i^l gets refined only once by the *dist-refine* function. Multiple PNN bounds can benefit holistically from the same refinement, keeping the global number of refinements as low as possible (thus making the evaluation of the PNN bounds in Line 10 more efficient).

LEMMA 1 (TERMINATION). *Algorithm 1 always terminates after a finite number of partition refinements.*

Proof. Let S^* be the set of distance partitions S_i initialized in Line 1 with their respective S_i^{min} instances. Lines 4-12 ensure that at least one partition S_i is refined at every iteration. In the worst case, all distance partitions $S_i \in S^*$ are refined completely, obtaining their respective S_i^{max} instances. Consequently, the PNN probability bounds B_i converge to the exact probability estimate, i.e. $P_{NN}^{lb}(Q, X_i) = P_{NN}^{ub}(Q, X_i)$. It is then easy to show that $top_{k+1}(B^{ub}) < top_k(B^{lb})$, i.e., $d < c$. We assume that the $P_{NN}(Q, X_i)$ estimates are distinct values, i.e., if $P_{NN}(Q, X_i) = P_{NN}(Q, X_j)$ then it must be that $i = j$.

3.4 Tightening the PNN bounds

The optimal search path to identify the top-k probable nearest neighbors in the *Holistic-PkNN* method (Algorithm 1) minimizes the number of partition refinements (Lines 7-9) and minimizes the number of evaluations of the PNN probability bounds (Line 10). Unfortunately its determination is computationally prohibitive, as it would require an extensive search in the solution space of the possible refinements, altogether with repeated evaluations of the PNN probability bounds. The incurred CPU cost would deny any expected benefit provided by the iterative refinements, and this motivates the introduction of efficient sub-optimal algorithms.

In this section, we discuss efficient implementation of the functions *find-critical* and *find-splits* used in Algorithm 1. While the procedure *find-critical* identifies the PNN bounds B_i to be tightened, procedure *find-splits* finds the refinements in the distance partitions S_i to be applied.

3.4.1 *find-critical*

We identify the candidates whose probability bounds have to be tightened as the set of the active candidates, X^*. In contrast to selection heuristics [26, 8] proposed in prior studies, our experiments show that it is more convenient to consider all the candidates in the critical region at each iteration rather than a small subset.

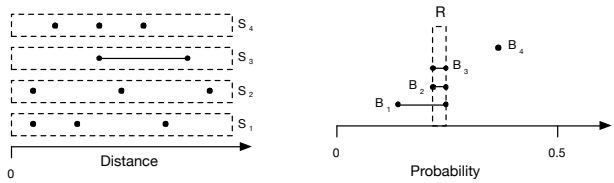
3.4.2 *find-splits*

Given a set or uncertain series R, the *find-splits* procedure identifies the best distance intervals S_i^l of the distance partitions S_i to be splitted, i.e., refined. First, we discuss the importance of the dependencies across different distance partitions in the selection of the refinements to apply.

LEMMA 2 (DEPENDENCIES IN DISTANCE PARTITIONS). *Tightening the PNN probability bounds B_i of candidate $X_i \in X^*$ may require some refinements in the distance partition S_j, $j \neq i$. Uncertain series X_j may not belong to the active set, i.e., $X_j \notin X^*$.*

Demonstration by example. Let D be a dataset with $N = 4$ and $m = 3$. Let S_1, S_2, S_3 and S_4 be the instantiated distance partitions: $S_1 = \{[2, 2] : 0.33, [4, 4] : 0.33, [6, 6] : 0.33\}$, $S_2 = \{[4, 8] : 1\}$, $S_3 = \{[1, 1] : 0.33, [5, 5] : 0.33, [9, 9] : 0.33\}$ and $S_4 = \{[1, 1] : 0.33, [3, 3] : 0.33, [7, 7] : 0.33\}$.

The PNN probability estimates determined using the Eq.4 and Eq.5 result in the following B_i bounds: $B_1 = [0.14, 0.25]$, $B_2 = [0.22, 0.25]$, $B_3 = [0.22, 0.25]$ and $B_4 = [0.37, 0.37]$.



(a) (b)

Figure 5: Graph (a) shows the distance partitions S_i and graph (b) reports their respective PNN bounds B_i.

We want to identify the top-2 most probable nearest neighbors, i.e., $k = 2$. Figure 5(a) and Figure 5(b) show the corresponding distance partitions S_i and the PNN probability bounds B_i, respectively. We observe that the candidate uncertain series X_4 and X_1 can be safely appended to the result set and discarded, respectively. We are unable to discriminate the PNN probabilities of X_2 and X_3. However, distance partitions S_1, S_2 and S_3 have been fully refined to their S_i^{max} instantiations and cannot be refined further. There is no other choice than refining samples in the distance partition S_2, whose respective uncertain series X_2 is not in the set of the active candidates X^* and has zero probability of being part of the result.

We introduce the *Pair-split* method in Algorithm 2 as baseline implementation of the *find-split* procedure. Let $width(S_i^a)$ be the distance width of the distance interval S_i^a. The algorithm iterates

Algorithm 2 Pair-split(S_i: distance partition)

1: $R \leftarrow \emptyset$
2: $score_{best} \leftarrow 0$
3: **for** $\forall\, (S_i^a, S_j^b)$ s.t. $S_i^a \in S_i \wedge S_j^b \in S_j \wedge i \neq j$ **do**
4: $\quad score \leftarrow W_i^a \cdot width(S_i^a) + W_j^b \cdot width(S_j^b)$
5: $\quad$ **if** $score > score_{best} \wedge S_i^a$ overlaps with S_j^b **then**
6: $\quad\quad R \leftarrow \{S_i^a, S_j^b\}$
7: $\quad\quad score_{best} \leftarrow score$
8: $\quad$ **end if**
9: **end for**

over all combinations of the (S_i^a, S_j^b) pairs (Lines 3-9). Line 4 defines a score based on the two interval weights and the respective distance widths. Lines 5-8 select the pair (S_i^a, S_j^b) s.t. they overlap with the largest score value.

Intuitively, splitting the pair of overlapping distance intervals (S_i^a, S_j^b) whose weighted width is the largest is expected to improve the PNN probability bounds of the respective PNN probability estimates B_i and B_j, respectively.

Note that the *pair-split* algorithm considers only pair-wise dependencies between distance intervals. However, there may be a distance interval S_i^a, overlapping with a large number of distance intervals S_j^b, whose score is too low to get selected. In the next section, we propose heuristics that overcome this limitation.

3.4.3 Uncertainty-aware distance refinements

In this section we discuss how to identify a pair of refinements to tighten the PNN bounds B_i by looking explicitly at the candidate refinements in distance partition S_i and in distance partitions S_j, $\neq i$. The proposed heuristics consider the pair-wise dependencies between different distance partitions S_i and S_j, $i \neq j$.

First, we identify the best distance interval S_i^a to tighten the PNN bounds B_i. We observe that refining S_i^a may be beneficial to tighten B_i only if it overlaps with one or more distance intervals S_j^b, $j \neq i$. Among the S_i^a candidate refinements, we select the S_i^a candidate that maximizes the weighted sum of the overlapping distance intervals with all other partitions S_j s.t. $j \neq i$. Function *select-inner* implements this strategy:

$$\text{select-inner}(Q, X_i) = \underset{S_i^a \in S_i}{\text{argmax}} \; W_i^a \prod_{\forall j \neq i} \left(\sum_{S_j^b \in S_j} W_j^b \cdot I(lb(S_i^a) < ub(S_j^b) \wedge ub(S_i^a) > lb(S_j^b)) \right) \tag{6}$$

Second, we identify the best distance interval S_j^b to tighten the PNN bounds B_i s.t. $j \neq i$. Similarly, we observe that refining S_j^b may be beneficial to tighten B_i only if it overlaps with one or more distance intervals in S_i. Among the S_j^b candidate refinements, we select the S_i^b candidate refinement that maximizes the weighted sum of overlapping distance intervals with the distance intervals in partition S_i. Function *select-outer* implements this strategy:

$$\text{select-outer}(Q, X_i) = \underset{S_j^b \in S_j, \forall j \neq i}{\text{argmax}} \; W_j^b \sum_{S_i^a \in S_i} W_i^a \cdot I(ub(S_i^a) > lb(S_j^b) \wedge lb(S_i^a) < ub(S_j^b)) \tag{7}$$

The *find-split* procedure can be implemented by returning the distance intervals identified by the *select-inner* and the *select-outer* heuristics. We observe that the computation of Eq.6 can be combined efficiently with the evaluation of the PNN upper-bound in Eq.5 because of the similarities in the formulation and in the verified inequalities. Eq. 7 cannot be combined in a similar way with the evaluation of the PNN intervals because of the different ordering in the enumeration of the distance interval pairs.

3.5 Distance Partitions

We now discuss an efficient implementation of the procedure *init-distance-partitions* that initializes the distance partitions S_i, and their incremental refinement required in the *dist-refine* function in Algorithm 1.

3.5.1 Initialization

The *init-distance-partitions* procedure initializes the distance partitions S_i for all candidates $X_i \in D$. We consider two different implementations that can be used to construct the distance partitions using their S_i^{min} instantiations.

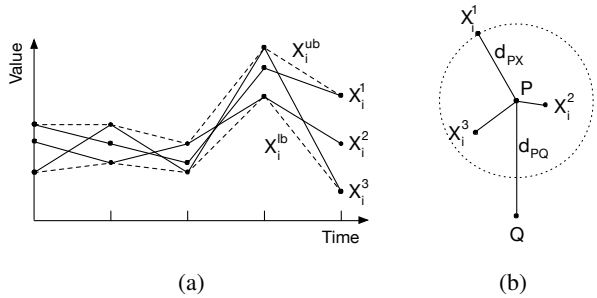
First, we present distance bounds inspired by the spatial properties of the uncertain series. The value of an uncertain series X_i can be bounded by the minimum and maximum values at each time-stamp across all its instantiations X_i^l, where $1 \leq l \leq m$. The lower-bound series of uncertain series X_i at time-stamp t (denoted by $X_i^{lb}[t]$) is defined as:

$$X_i^{lb}[t] = X_i^l[t] : X_i^l[t] \leq X_i^k[t] \; \forall\, k \in \{1, ..., m\}$$

Similarly, the upper-bound series of uncertain series X_i at time-stamp t (denoted by (denoted by $X_i^{ub}[t]$) is defined as:

$$X_i^{ub}[t] = X_i^l[t] : X_i^l[t] \geq X_i^k[t] \; \forall\, k \in \{1, ..., m\}$$

DEFINITION 3.4 (UNCERTAIN SERIES ENVELOPE). *Let X_i^{lb} and X_i^{ub} be the lower-bound and upper-bound series of uncertain series X_i, respectively. The uncertain series envelope of uncertain series X_i is defined as the pair of series $E_i = (X_i^{lb}, X_i^{ub})$.*



(a) (b)

Figure 6: Graph (a) shows an example of uncertain series envelope. Graph (b) illustrates an example of metric distance bounds.

Figure 6(a) shows an example of an uncertain series envelope. We observe that the uncertain series envelope E_i identifies the smallest n-dimensional hyper-rectangle enclosing all instantiations of uncertain series X_i. Note that Uncertain series envelopes are equivalent to the concept of Minimum Bounding Rectangles (MBRs), a popular representation of bounding regions in spatial indexes. Envelopes can be pre-computed, since they don't depend on the query series. Given a query Q, an uncertain series envelope E_i can be used to determine the lower-bound of the distance samples in $Dist(Q, X_i)$ as follows: $spatial_{lb}(Q, X_i) =$

$$\sqrt{\sum_{1 \le t \le n} \begin{cases} 0 & \text{if } X_i^{lb}[t] \le Q[t] \le X_i^{ub}[t] \\ min((Q[t] - X_i^{lb}[t])^2, & \\ (Q[t] - X_i^{ub}[t])^2) & otherwise \end{cases}} \quad (8)$$

Similarly, the upper-bound (denoted by $spatial_{ub}(Q, X_i)$) of the distance samples in $Dist(Q, X_i)$ is defined as: $spatial_{ub}(Q, X_i) =$

$$\sqrt{\sum_{1 \le t \le n} max((Q[t] - X_i^{lb}[t])^2, (Q[t] - X_i^{ub}[t])^2)} \quad (9)$$

The distance bounds between uncertain series X_i and query Q are determined by measuring the minimum and maximum distances between the uncertain series envelope E_i and query Q. An equivalent formulation of the distance lower-bound can be found in [30]. On the contrary, the upper-bound $spatial_{ub}(Q, X_i)$ is novel.

We introduce now a different formulation of distance bounds, inspired by the metric properties of the distance function. Let P be a series serving as *pivot* in the metric space. A *pivot* is a series that has been selected as representative series during the distance computations, and its distance to an uncertain series X_i can be used to bound the distance between X_i and the query Q thanks to the triangle inequality property. We note that the triangular inequality holds only in metric spaces, i.e. distance spaces induced by metric distance measures.

Let d_{PX} be the maximum distance between series P and all X^l instantiations, i.e. $d_{PQ} = max(Dist(X_i, P))$. Let d_{PQ} be the distance between P and Q, i.e., $d_{PQ} = dist(P, Q)$. d_{PX} can be pre-computed, since it does not depend on the query Q. Figure 6(b) shows an example (only distances are relevant, the example is projected in two dimensions for ease of exposition).

Given a query Q, a pivot P and uncertain series X, the distance between X and Q is bounded by the following lower-bound: $max(0, d_{PQ} - d_{PX})$. Similarly, one can define an upper-bound for the distance between X and Q: $D_{PQ} + D_{PX}$. These bounds have been widely used in index structures for metric spaces [24, 10]. Multiple pivot series can be combined to increase to improve the distance bounds as follows. Let P^* be a set of pivot series. The best lower-bound (denoted by $metric_{lb}(Q, X_i)$) is determined using the pivot P s.t. it maximizes $max(0, d_{PQ} - d_{PX})$ and the best upper-bound (denoted by $metric_{ub}(Q, X_i)$) is determined using P' s.t. it minimizes $d_{P'Q} + d_{P'X}$:

$$metric_{lb}(Q, X_i) = \max_{P \in P^*} max(0, d_{PQ} - d_{PX})$$

$$metric_{ub}(Q, X_i) = \min_{P \in P^*} (d_{PQ} + d_{PX})$$

The tightness of the metric bounds depends on the quality of the selected pivot series. In the experimental evaluation we consider different strategies that have been proposed in prior works.

The distance partitions S_i for all uncertain series X_i are initialized using their spatial or metric bounds in the *Holistic-PkNN* algorithm using a linear scan on the dataset D. We observe that in the worst case all distance bounds obtained using metric or spatial bounds overlap. If the initial distance bounds overlap then further refinements are needed to discriminate the respective PNN probability bounds. In the following, we discuss how distance partitions are refined incrementally.

3.5.2 Refinement

Let S_i be a distance partition to be refined and let S_i^l be the distance interval that has been previously selected for refinement by the *find-splits* procedure as presented in Section 3.4.2.

If the bounds of the distance interval S_i^l have been determined using metric or spatial distance bounds (Section 3.5.1), then $lb(S_i^l)$ and $ub(S_i^l)$ are lower- and upper-estimates of $min(Dist(Q, X_i))$ and $max(Dist(Q, X_i))$, respectively. We substitute S_i^l with a new distance interval S_i^k s.t. $lb(S_i^k) = min(Dist(Q, X_i))$ and $ub(S_i^k) = max(Dist(Q, X_i))$. Please note that, after the substitution, the distance partition S_i is a new instantiation of the distance partition S_i^{min}. Distance samples $Dist(Q, X_i)$ are maintained in a sorted array. The optimal distance bounds $min(Dist(Q, X_i))$ and $max(Dist(Q, X_i))$ of the new instantiation of S_i^{min} serve as first refinement for S_i.

In case of subsequent refinements, the refinement of the distance interval S_i^l is performed by partitioning the region covered by S_i^l into two new distance intervals. Let p_{lb}, p_{ub} be a pair of pointers to the sub-array in the $Dist(Q, X_i)$ sorted array that identifies the lowest and highest distance sample in S_i^l, and let $dist_{mid}$ be the value that partitions the region defined by S_i^l into two equi-width regions. Then, the closest sample to $dist_{mid}$ in the sub-array identified by pointers p_{lb}, p_{ub} is used as largest sample in the new left partition, and its immediate next sample in the sorted array is used as lowest sample in the new right partition.

While the first refinement is very expensive, since it requires the evaluation of the distance samples in $Dist(Q, X_i)$, the CPU cost of subsequent refinements is negligible: it is bounded by the cost of evaluating a binary search on a subset of the distance samples, i.e., $O(log(m))$. Recall that m is the number of X_i instantiations.

4. EXPERIMENTAL RESULTS

In this section we empirically evaluate our proposal under a variety of settings, assessing time performance and accuracy. We implemented all techniques in C++, and ran the experiments on a Linux machine with Intel Xeon 1.80GHz processors and 64GB of RAM. For all results, we report the averages of the measurements obtained from 10 independent runs, as well as the 95% confidence intervals. Confidence intervals are reported when the y-axis is not in log scale for clearness.

Datasets: We consider real and synthetic datasets. Time performance is assessed on synthetic datasets constructed as follows. Let $walk_i$ be a random walk of length n where the value difference between neighboring time-stamps is a normally distributed random variable with zero mean and unit standard deviation. The series $walk_i$ is then normalized [25], obtaining a new series denoted by $seed_i$. The l-th sample of uncertain series X_i, X_i^l, is obtained by adding samples drawn from a normally distributed random variable $N(0, \sigma)$ to the $seed_i$ values. Finally, the samples are normalized again. The procedure is repeated independently for each uncertain series X_i. Please note that the seed series $seed_i$ used to generate the samples of uncertain series X_i is different from the other seed series $seed_j$, $j \neq i$. Random queries are generated similarly (and independently) to the X_i series samples.

Accuracy is assessed on a second different synthetic dataset that allows us to assign class labels to the queries according to the generation model. Uncertain series are derived for a single $seed$ series. The samples of uncertain series X_i are obtained by perturbing the $seed$ series with standard deviation $\sigma = \alpha + i\beta$. α ensures a minimum distance to the $seed$ series and β imposes a total order on the distance of the $seed$ series and the uncertain series that we use as ground truth. We observe that series X_1 is the NN to the $seed$ series according to this perturbation model, and its $seed$ is used as query. We additionally consider 45 real datasets from the UCR time series collection (described in detail in [1]) with class labels. The datasets are divided into two sets, *train* and *test*. We assume that these raw series are samples drawn from unknown distributions that may be noisy and error-prone. Similarly to prior works [8, 11, 12, 21, 31, 7, 27, 14], uncertain series are constructed from the series in the *train* set: first, normalized real series are used as seed series $seed_i$. Second, the perturbation standard deviation σ is used to generate positive samples of the normal distribution, then used independently as perturbation standard deviation σ' for each sample. The resulting uncertain series is composed by a set of samples whose perturbation standard deviation varies and is controlled by σ.

Evaluation measures: We now describe how accuracy and time performance have been assessed. Given a random query series Q drawn from the *test* set, the class label associated with the uncertain series identified as the NN in the *train* set is assigned as label. For each method we compute the confusion matrix M (M_{ij} is the count of query series assigned to class i instead of j). Accuracy is defined as the ratio of true positives for all class labels:

$$Accuracy = \frac{1}{|test|} \sum M_{ii}, \tag{10}$$

where $|test|$ is the size of the *test* set. If all queries are labeled correctly, M is a diagonal matrix. The experiment is repeated several times on each dataset (*train* and *test* pair sets) to get statistically significant results. We note that the very same results can be obtained using different algorithms that implement the top-k probable nearest neighbor queries as formulated in Section 2.

The time performance is measured as the time required to evaluate the specified queries. In the first set of experiments, we also evaluate the quality of diverse strategies to prune the search space.

Algorithms: The time performance of our algorithm is compared to the *Baseline* algorithm (presented in Section 3.1), and an adaptation of the *Find-TopK-PNN* method [8]. The algorithm *Find-TopK-PNN* has been designed to minimize the number of I/O operations: uncertain series are retrieved in min-distance order to the query. The PNN probability upper-bound of a virtual object is used to bound the PNN upper-bound probabilities of all non-retrieved objects, and it is used to control the retrieval of new uncertain series until convergence is reached. Our adaptation, named *Holistic-PkNN-Virtual*, considers our best-performing combination of the procedures for the initialization and the refinement of the distance partitions, and for the evaluation of the PNN probability bounds. We also evaluate four distinct versions of our algorithm, *Holistic-PkNN*, which correspond to different combinations of the *find-critical* and *find-splits* implementations: the *find-critical* function can be implemented using the *multi-simulation* procedure [26] (denoted by the suffix *M*), or by enumerating all candidates whose PNN probability bounds overlap with the critical region (denoted by the suffix *H*); the *find-splits* function can be implemented using the *pair-split* method (Algorithm 2, denoted by the suffix *P*), or using the *select-inner* and *select-outer* procedures (Eq. 6 and Eq. 7, denoted by the suffix *S*). From our experiments, the optimal configuration is *Holistic-PkNN-HS*, also denoted in short by *Holistic-PkNN*. Similarly to [5], we use the average of the series samples as unique pivot for each uncertain series. We note that the algorithms *Baseline*, *Holistic-PkNN-Virtual* and *Holistic-PkNN* are different algorithms that return the very same result set as defined by the formulation of top-k probable nearest neighbor queries defined in Section 2.

Parameter	Range
No. of samples (m)	[100, ..., **500**, ..., 1000]
No. of uncertain series (N)	[100, ..., **1000**, ..., 100000]
Standard deviation (σ)	[0.1, ..., **0.5**, ..., 1]
Series length (n)	[100, ..., **256**, ..., 1000]
Result set size (k)	[**1**, ..., 16]

Table 2: Experiment parameter configuration ranges. Default values are indicated in bold.

We further evaluate an adaptation of M-trees to prune the search space. An M-tree is a tree whose nodes represent regions in a metric space and can be used to index objects in metric spaces [24]. In contrast to the original M-tree where a leaf node is used to represent a set of indexed objects stored on disk, each leaf node indexes an uncertain series.

The accuracy of our proposal is compared to the algorithms DUST [27] and *Euclidean-AVG*. The DUST algorithm models the uncertain series as a series of independent random variable, and the distance between the query and the uncertain series is based on the probability of their distance to be zero. The *Euclidean-AVG* algorithm averages all samples to obtain their average series, that is then used as representative to determine a distance measure between the uncertain series and the query.

The parameters considered in the experiments are summarized in Table 2. When not explicitly stated, we use the default configuration value (highlighted in bold).

4.1 Effectiveness of Pruning Strategies

The tightness of the distance bounds during the initialization of the distance partitions as detailed in Section 3.5.1 can greatly reduce the processing time, thanks to the early pruning of a large fraction of the candidate uncertain series. In the first set of experiments, we measure the ratio of candidates that cannot be pruned immediately after the initialization of their distance partitions for the *spatial*, *metric* and *exact* distance bounds on synthetic datasets constructed to evaluate the time performance. Recall that the *spatial* and *metric* distance bounds can be estimated efficiently (Section 3.5), while *exact* distance bounds require the evaluation of the exact distance samples in $Dist(Q, X_i)$. We stress that the *spatial* and *metric* distance bounds lower- and upper-bound the *exact* distance bounds and the final produced result is exactly the same.

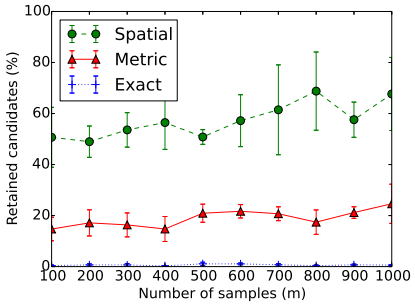


Figure 7: Ratio of retained candidates when varying the number series samples m.

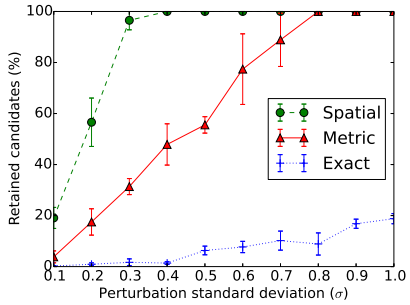


Figure 8: Ratio of retained candidates when varying the perturbation standard deviation σ.

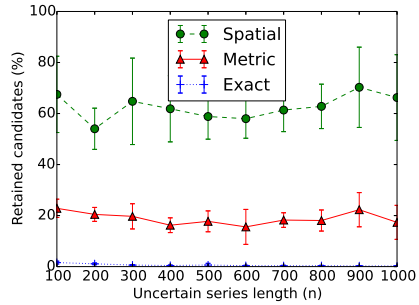


Figure 9: Ratio of retained candidates when varying the uncertain series length n.

In Figure 7 we report the ratio of the retained uncertain series after pruning the candidates based on the distance partitions initialized using the *exact*, *metric* and *spatial* distance bounds with perturbation standard deviation $\sigma = 0.2$ when varying the number of samples, m. The *exact* distance bounds are the most accurate, followed by the *metric* and then the *spatial* distance bounds. The average ratio of the retained candidates is 0.7%, 19% and 57% when the distance partitions are initialized using the *exact*, *metric* and *spatial* distance bounds, respectively. We observe a slight increase in the ratio of retained candidates as the number of samples m increases. Recall that the m samples are drawn from a normal distribution. As the number of samples m increases, the probability that at least a few samples deviate significantly from the mean value increases. Although the distance bounds are affected by outlier samples, the overall sample distribution is stable and does not depend on the number of samples.

In the next experiment, depicted in Figure 8, we vary the perturbation standard deviation σ, and report the ratio of retained uncertain series after pruning the candidates based on their initialization of the distance partitions for the *spatial*, *metric* and *exact* distance bounds. As the perturbation standard deviation σ increases, the pruning power of the different initializations for the distance partition is reduced. With *exact* bounds, 20% of the candidates is retained when σ approaches 1.0. On the contrary, 100% of the candidates are retained when σ approaches 0.4 and 0.8 for the *spatial* and *metric* bounds, respectively.

In Figure 9 we report the ratio of retained uncertain series after pruning the candidates based on their initialization of the distance partitions for the *spatial*, *metric* and *exact* distance bounds with perturbation standard deviation $\sigma = 0.2$ when varying the length of the uncertain series, n. Similarly to the previous experiments, *exact* distance bounds are the most accurate, followed by the *metric* and then the *spatial* distance bounds. The series length n does not affect significantly the performance. We note that normally distributed samples along the candidate series result in a normally distributed distance between the candidate and the query. The same applies for other distributions, independently from the series length n that does not affect the effectiveness of the pruning strategies. The above experiments show that the *metric* distance bounds outperform consistently the *spatial* distance bounds in terms of ratio of pruned candidates.

4.2 Time performance

In the next set of experiments, we compare the time performance using the different methods to initialize the distance partitions S_i to evaluate Top-k-$P_{NN}(D, Q, k)$ queries. Figure 10 reports the time performance when varying the number of samples m for the *spatial*, *metric* and *exact* distance bounds. The *metric* distance bounds are the best performing, followed by the *spatial* and then the *exact* distance bounds. We observe that the evaluation of Top-k-$P_{NN}(D, Q, k)$ queries using the *metric* distance bounds can be up to 18% faster than using *spatial* distance bounds. This is due to the tighter distance bounds when using the *spatial* bounds. It is worth noting that the *exact* distance bounds are even tighter, but the high incurred CPU cost for the evaluation of the exact distance samples leads to inferior overall performance.

We report the time performance when varying the perturbation standard deviation σ with distance partitions initialized using *spatial*, *metric* and *exact* distance bounds in Figure 11. then the *exact* distance bounds. As the perturbation standard deviation σ increases, all methods perform nearly the same. This is due to the increasing probability of overlaps between the distance distributions. As this probability increases, the CPU cost is mainly driven by the repeated evaluation of the PNN bounds and the iterative refinements of the distance partitions.

The time performance with varying number of uncertain series N is illustrated in Figure 12. The distance partitions are initialized using the *spatial*, *metric* and *exact* distance bounds. Similarly to the previous experiments, the *metric* distance bounds are the best performing, followed by the *spatial* and then the *exact* distance bounds. As expected, the CPU cost increases linearly to the dataset size, N.

Finally, we evaluated the time performance when pruning the candidates with the *metric* distance bounds using a linear scan over all candidates and our adaptation of the M-tree index. The M-tree index proved to be competitive to a linear scan only when the perturbation standard deviation σ is low, i.e., $\sigma = 0.2$. In conclusion, the distance partitions initialized using the *metric* distance bounds using one pivot series are the best performing. In the rest of this study we initialize the distance partitions S_i using the *metric* distance bounds, initialized through a linear scan over the dataset.

4.2.1 Comparison to Prior Approaches

In the next series of experiments, we compare the time performance of the algorithms *Baseline* and the different versions of the *Holistic-PkNN* algorithm. We may omit the prefix *Holistic* for ease of presentation in some of the figures. Recall that the *Holistic-PkNN-Virtual* technique is our adaptation of the *Find-TopK-PNN* algorithm [8].

We report the time performance when varying the number of samples m for the *Baseline* and *Holistic-PkNN* algorithms in Figure 13. The graph shows that *Holistic-PkNN-HS* is the best per-

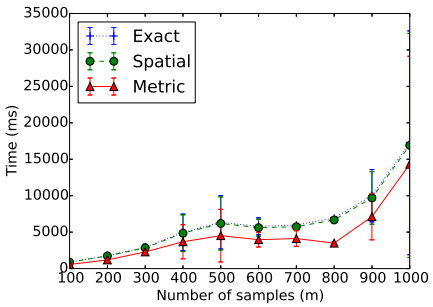


Figure 10: Time performance when varying the number of samples m.

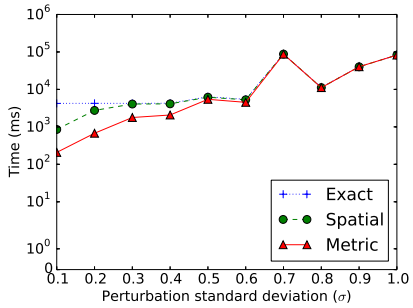


Figure 11: Time performance when varying the perturbation standard deviation σ.

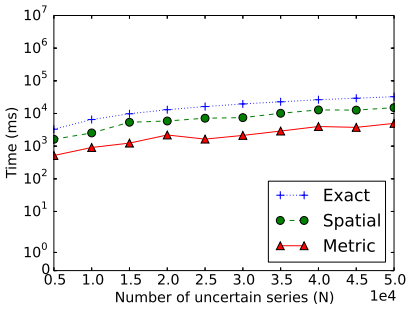


Figure 12: Time performance when varying the number of uncertain series N.

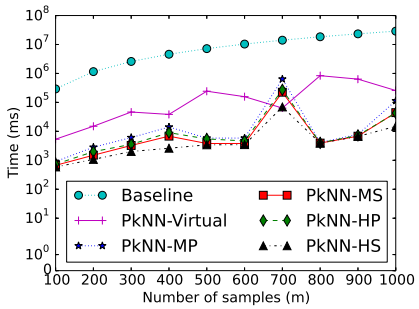


Figure 13: Time performance when varying number of samples m for *Baseline* and *Holistic-PkNN* algorithms.

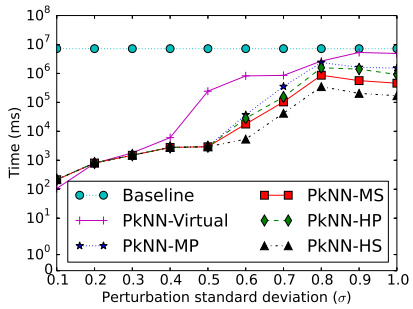


Figure 14: Time performance when varying the perturbation standard deviation σ for *Baseline* and *Holistic-PkNN* algorithms.

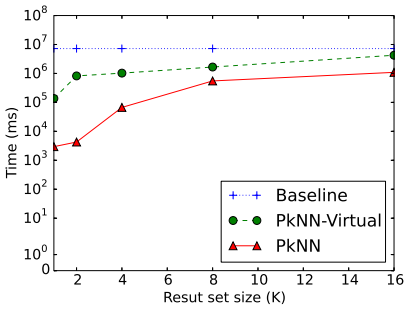


Figure 15: Time performance when varying k for the *Baseline*, *Holistic-PkNN* and *Holistic-PkNN-Virtual* algorithms.

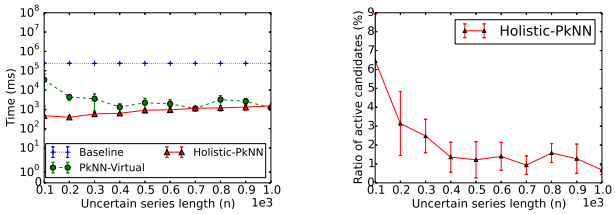
forming, while the *Baseline* approach performs the worst. We note that *Holistic-PkNN-HS* is up to two orders of magnitude faster then the *Holistic-PkNN-Virtual* algorithm. This is due to the larger number of iterations of the *Holistic-PkNN-Virtual* algorithm, that continuously increases the number of candidates until convergence. The larger number of iterations is associated with a larger, very CPU intensive, number of evaluations of the PNN probability bounds. The *Baseline* approach does not rely on pruning strategies to reduce the number of candidates. The distinct distance samples in $Dist(Q, X_i)$ are used in the pairwise comparisons in the baseline algorithm. The incurred CPU cost is up to three orders of magnitude higher than that of the algorithms in the *Holistic-PkNN* family.

In the next experiment, we assess the time performance when varying the perturbation standard deviation σ for the *Baseline* and *Holistic-PkNN* algorithms. The results are reported in Figure 14 and show that *Holistic-PkNN-HS* is in general the best performing. The *Baseline* approach exhibits again the worst performance. The CPU cost of the *Baseline* algorithm is not affected by the properties of the perturbation, and is constant. We observe that the *Holistic-PkNN-Virtual* algorithm is the best performing when the perturbation standard deviation σ is lower than 0.2. An in-depth analysis of the execution revealed that a lower number of candidates is retrieved by the *Holistic-PkNN-Virtual* when the perturbation is sufficiently low. The reduced number of candidates is limiting the number of evaluations of the PNN probability bounds across all iterations in contrast to the *Holistic-PkNN-HS* algorithm.

We report our results on performance when varying the number k of retrieved uncertain series in Figure 15. *Holistic-PkNN* is the best performing. We observe that the *Baseline* approach determines the exact PNN probability estimates for all candidates and the CPU cost incurred in the identification of the top-k result set is negligible if compared to the computational cost of evaluating the PNN estimates. The *Holistic-PkNN* algorithm is more than one order of magnitude faster than the *Holistic-PkNN-Virtual* method when k is low.

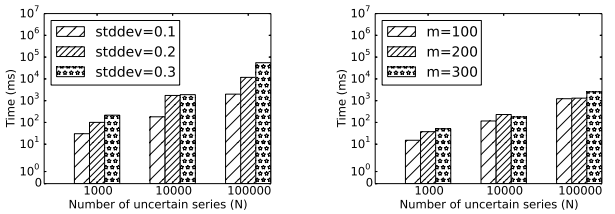
We conclude by reporting the time performance when varying the uncertain series length n. The results in Figure 16(a) show that *Holistic-PkNN* is the winner across the entire range of values, with a larger margin for the low range of series lengths. We observe that the incurred CPU time cost does not increase steadily as the series length n increases. An in-depth analysis of the execution traces revealed that, as n increases, the difference between the closest and the farthest distance samples increases. In other words, as the dimensionality (n) increases, the distance samples are spread in a wider region. This results in a lower probability of overlaps between the distance partitions, eventually resulting in less active candidates in X^* as reported in Figure 16(b). Fewer active candidates in X^* translate to fewer PNN probability bound estimates and fewer refinements of the distance partitions. We further note that the time required to determine the distance samples increases linearly to n. The resulting time performance is a combination of these two characteristics of the distance distributions.

Following the results presented in Section 4.1, we conclude that the *Holistic-PkNN-Virtual* algorithm combined with M-trees is expected to be the best performing only when the perturbation standard deviation is low. In general though, the *Holistic-PkNN-HS* algorithm using a linear scan to initialize the distance partitions S_i with *metric* distance bounds is to be preferred.



(a) (b)

Figure 16: Graph (a) shows the time performance when varying the uncertain series length n for the *Baseline*, *Holistic-PkNN* and *Holistic-PkNN-Virtual* algorithms. Graph (b) reports the ratio of candidates in the active set X^* when varying the uncertain series length n for the *Holistic-PkNN* algorithm.



(a) (b)

Figure 17: *Holistic-PkNN* time performance when varying the number of uncertain series N for (a) different levels of perturbation standard deviation σ, and (b) number of uncertain series samples m.

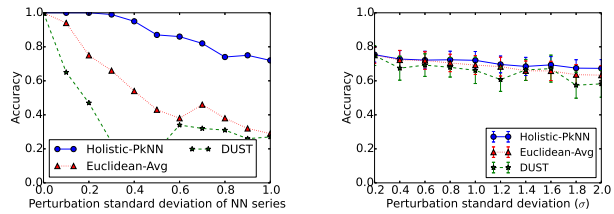
4.2.2 Scalability

Finally, we report our results on scalability. Figure 17(a) shows the time performance for different levels of perturbation standard deviation σ, when varying the number of uncertain series N using the *Holistic-PkNN* algorithm. We observe that the incurred CPU cost for $N = 10,000$ and perturbation standard deviation $\sigma = 0.3$ is similar to the time performance for $N = 100,000$ and perturbation standard deviation $\sigma = 0.1$. The size of the dataset N and the perturbation standard deviation σ are two parameters that affect significantly the time performance of the algorithm, and even relatively small datasets are challenging when a large fraction of the distance distributions overlap (i.e., when σ is large). Figure 17(b) shows the time performance of *Holistic-PkNN* for different configurations of the number of uncertain series N and samples m. The incurred time increases as N and m increase, with N being the most critical parameter.

These experiments show that *Holistic-PkNN-HS* can be used in practice for large datasets, even with a high perturbation standard deviation σ and number of samples m.

4.3 Quality results

In this section, we report our results on the classification accuracy defined in Eq. 10 on synthetic and real datasets constructed to evaluate the accuracy. Figure 18(a) reports the accuracy for methods DUST, *Euclidean-Avg* and Top-k-$P_{NN}(D, Q, k)$ on a synthetic dataset, where the uncertain series are obtained by perturbing the query series with an increasing standard deviation σ'. As the perturbation increases, the accuracy for DUST and *Euclidean-Avg* deteriorates, while Top-k-$P_{NN}(D, Q, k)$ manages to maintain a high accuracy. In Figure 18(b), we present our results on 45 real datasets.



(a) (b)

Figure 18: Accuracy when (a) varying the NN perturbation standard deviation σ', and (b) varying the perturbation standard deviation σ.

The experiment shows the accuracy by varying the perturbation standard deviation σ for the NN classifier defined using the algorithms DUST, *Euclidean-Avg* and Top-k-$P_{NN}(D, Q, k)$. We observe that, as the perturbation standard deviation increases, the top-1 probable nearest neighbor formulation is more accurate than the DUST and *Euclidean-Avg* algorithms. The DUST algorithm requires to know the exact value distribution at each time-stamp. In our experiments we assume the normal distribution whose parameters are estimated from the samples. Moreover, DUST uses lookup tables of predetermined probability estimates to make the algorithm tractable. The errors due to these approximations cause the DUST algorithm to perform worse than the *Euclidean-Avg* method.

5. RELATED WORK

The problem of modeling, querying and mining uncertain data has been investigated extensively in recent years [4, 13]. At the same time we have witnessed an increased interest in data series management and processing [32, 23, 22, 9], related to data produced by sensors, or scientific experiments.

The "possible worlds" model [2] formalizes uncertainty by defining the space of the possible instantiations of the database. Instantiations must be consistent with the semantics of the data. For example, in a spatio-temporal database there may be two distinct possible trajectories representing the uncertain trajectory of a moving object, but an object cannot be in two different locations at the same time. In the context of time series databases, a series can be formalized as a point in a high-dimensional space with correlated dimensions. An uncertain series can then be represented by enumerating its possible instantiations under the "possible world" semantics. Prior works on uncertain time series [27, 31, 7] introduce the additional assumption of independence across different timestamps. Nevertheless, temporal correlation is a a well known property of time series data and ignoring it may lead to erroneous results. The interested reader can find an analytical and experimental comparison of the aforementioned methods in [14].

The evaluation of top-k probable nearest neighbor queries on uncertain data is a well recognized problem, that can be tracked back to the seminal work of Cheng et al. [12]. Subsequently many different formulations of "nearest neighbor" in uncertain data have been proposed [20, 29, 21]. A detailed review of the state of the art can be found in [18]. In [12], the authors dissect the processing of NN queries in four steps: projection, pruning, bounding and evaluation. The projection phase returns the regions bounding the object uncertainties, the pruning phase removes from the list of candidate objects with zero probability of being the NN, and finally the bounding and evaluation phases refine the probability bounds until the NN object is identified. The traits of this four-step approach

can be found in nearly all the subsequent studies tackling the same problems.

In [8] Beskales et al. proposed a method where the non-retrieved objects are modeled by means of a special "virtual" object that represents them all. When the "virtual" object is considered for insertion in the result set, a new real object is retrieved in minimum-distance order to be processed. The retrieved objects are represented by spatial regions that bound their uncertain location. When the bounding regions overlap, they are partitioned to obtain a more fine-grained representation of the object uncertainties. The algorithm terminates when the "virtual" object (and thus all non-retrieved objects) can be safely pruned and the bounding regions of the objects have been sufficiently refined to discriminate their NN probabilities. In [26] Re et al. proposed the Multi Simulation (MS) algorithm to discern the top-k most probable NN objects by running in parallel several Monte-Carlo simulations. The objects that cannot be safely added to the result set of that cannot be discarded are identified by using the notion of critical region. The critical region is a region in the probability space. Each object is represented by its probability interval of being the NN. The objects having their probability intervals overlapping with the critical region are selected for another simulation step until convergence (i.e., the critical region is empty). On the contrary to our approach, the NN probability bounds for different objects are not correlated.

6. CONCLUSIONS

Uncertain data series can be used to represent data series whose values are imprecise or inherently uncertain. In this work, we formalize the top-k nearest neighbor problem in uncertain data series, and propose a comprehensive model for uncertain series. Prior studies assume the independence of the value distribution at neighboring time-stamps as a simplifying assumption, compromising the accuracy of the model in order to reduce the problem complexity. In this study, we introduce the *Holistic-PkNN* algorithm, a method that captures both uncertainty and correlation, using novel metric bounds that are specifically suited to uncertain series. We further introduce a novel selection strategy that tightens the probability bounds using a small number of iterations and probability estimate computations. The experimental evaluation with real and synthetic datasets demonstrates the efficiency and effectiveness of the proposed approach.

7. REFERENCES

[1] Keogh, E., Xi, X., Wei, L. & Ratanamahatana, C. A. (2006). The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/∼eamonn/time_series_data.

[2] S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD*, 1987.

[3] C. C. Aggarwal. *Managing and mining uncertain data*. 2009.

[4] C. C. Aggarwal and P. S. Yu. A survey of uncertain data algorithms and applications. *TKDE*, 2009.

[5] F. Angiulli and F. Fassetti. Indexing uncertain data in general metric spaces. *TKDE*, 2012.

[6] L. Antova, C. Koch, and D. Olteanu. Query language support for incomplete information in the maybms system. In *VLDB*, 2007.

[7] J. Aßfalg, H.-P. Kriegel, P. Kröger, and M. Renz. Probabilistic similarity search for uncertain time series. In *SSDBM*, 2009.

[8] G. Beskales, M. A. Soliman, and I. F. Ilyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *PVLDB*, 2008.

[9] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. J. Keogh. Beyond one billion time series: indexing and mining very large time series collections with i sax2+. *Knowl. Inf. Syst.*, 2014.

[10] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *CSUR*, 2001.

[11] R. Cheng, J. Chen, M. F. Mokbel, and C.-Y. Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, 2008.

[12] R. Cheng, D. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *TKDE*, 2004.

[13] M. Dallachiesa, C. Aggarwal, and T. Palpanas. Node classification in uncertain graphs. In *SSDBM*, 2014.

[14] M. Dallachiesa, B. Nushi, K. Mirylenka, and T. Palpanas. Uncertain time-series similarity: Return to the basics. *PVLDB*, 2012.

[15] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, 2007.

[16] A. Fuxman, E. Fazli, and R. J. Miller. Conquer: Efficient management of inconsistent databases. In *SIGMOD*, 2005.

[17] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *CSUR*, 2008.

[18] J. Jestes, G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data. *TKDE*, 2011.

[19] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.

[20] H.-P. Kriegel, P. Kunath, and M. Renz. Probabilistic nearest-neighbor query on uncertain objects. In *DASFAA*, 2007.

[21] X. Lian and L. Chen. Probabilistic ranked queries in uncertain databases. In *EDBT*, 2008.

[22] A. Marascu, S. A. Khan, and T. Palpanas. Scalable similarity matching in streaming time series. In *PAKDD*, 2012.

[23] T. Palpanas. Real-time data analytics in sensor networks. In *Managing and Mining Sensor Data*, pages 173–210. 2013.

[24] M. P. Paolo Ciaccia and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, 1997.

[25] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *SIGKDD*, 2012.

[26] C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[27] S. Sarangi and K. Murthy. DUST: a generalized notion of similarity between uncertain time series. In *SIGKDD*, 2010.

[28] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah. Orion 2.0: native support for uncertain data. In *SIGMOD*, 2008.

[29] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.

[30] L. Wei, E. Keogh, H. Van Herle, and A. Mafra-Neto. Atomic wedgie: efficient query filtering for streaming time series. In *ICDM*, 2005.

[31] M. Yeh, K. Wu, P. Yu, and M. Chen. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In *EDBT*, 2009.

[32] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *SIGMOD*, 2014.