Accuracy-Aware Uncertain Stream Databases

Tingjian Ge, Fujun Liu

Department of Computer Science
University of Kentucky
ge@cs.uky.edu, fujun.liu@uky.edu

***Abstract*— Previous work has introduced probability distributions as first-class components in uncertain stream database systems. A lacking element is the fact of how accurate these probability distributions are. This indeed has a profound impact on the accuracy of query results presented to end users. While there is some previous work that studies unreliable *intermediate* query results in the tuple uncertainty model, to the best of our knowledge, we are the first to consider an uncertain stream database in which accuracy is taken into consideration all the way from the learned distributions based on raw data samples to the query results. We perform an initial study of various components in an accuracy-aware uncertain stream database system, including the representation of accuracy information and how to obtain query results' accuracy. In addition, we propose novel predicates based on hypothesis testing for decision-making using data with limited accuracy. We augment our study with a comprehensive set of experimental evaluations.**

I. Introduction

Recent research has extended stream databases to handle uncertain data in order to meet the requirements from ever-increasing applications in sensor networks and ubiquitous computing (e.g., [19]). Handling probabilistic and uncertain data has been identified as one of the most important research directions in this research field [1]. Most previous work in probabilistic databases assumes that probability distributions are *somehow* obtained, and that we have "good" knowledge about the distributions. However, in reality, this is often not the case.

Where do we obtain the probabilities in the first place? In many applications, probability distributions are *learned from observations and measurements, a.k.a. samples*. Such applications include sensor networks, ubiquitous computing, and scientific databases. Let us look at an example.

Example 1 (accuracy of learned probability distributions). *A few projects in both academia and industry* (*e.g., the CarTel project at MIT* [24] *and a product at INRIX* [26]) *provide traffic-aware routing and traffic mitigation. The basic idea is to use sensors* (*including GPS, WiFi*) *installed in traveling vehicles to measure real-time traffic delays on various roads in order to provide dynamic, accurate, and real-time traffic routing for travelers. Such a system uses the travel delays reported in a recent time window to infer the probability distribution of current delay at a road. Due to random factors, the best we can get is a distribution. In general, the more reports the system receives for a road, the more accurate the resulting distribution is. However, this number is constrained by how many test vehicles in the network are traveling on that particular road within the current time window. Figure* 1 *shows a snippet of the raw data that contains three observations for road* 19 *and fifty observations for road* 20.

Road ID	Length	Date	Time	Delay	Speed limit
19	200	2010-06-25	8:50	56	25
19	200	2010-06-25	8:51	38	25
19	200	2010-06-25	8:51	97	25
20	150	2010-06-25	8:49	72	30
⋮					
20	150	2010-06-25	8:51	59	30
⋮					

50 observations

Fig. 1 Original raw data samples from which probability distributions are learned and used in query processing for uncertain streams

For each of the two roads in Example 1, the database system can learn the distributions of *Delay* attribute (around time 8:50) using machine learning techniques, ranging from simple ones such as *histograms* to complex ones such as *kernel methods*, *maximum likelihood*, and *kNN* [2]. By doing this, a stream database system transforms the three (fifty, respectively) raw records of road 19 (20) into a single record with a distribution in the *Delay* field. Assuming that these fifty-three observations are equally trustworthy, clearly we can obtain a more accurate and reliable distribution for road 20 than for road 19.

This issue of variable accuracy of probability distributions is a prominent issue in *stream* databases, where a timely decision based on query processing must be made. To the best of our knowledge, previous work in this area does not try to differentiate two random variables (r.v.) in a stream database whose probability distributions have *distinct levels of accuracy*; simply modeling an r.v. with a probability distribution lacks this accuracy information.

In previous work, once a distribution is learned, its accuracy information is lost. The consequence of being *accuracy-oblivious* is that, in the end, the query results would also be accuracy-oblivious. Thus, the end user has no clue about how accurate or reliable the query results that she gets. For example, based on the raw data in Figure 1, we generate one record for road 19 that contains a histogram distribution for the *Delay* attribute, and likewise one record for road 20. Consider a query "SELECT *Road_ID* FROM *t* WHERE *Delay* $>_{2/3}$ 50",

in which the predicate denotes that, with probability at least 2/3, *Delay* is greater than 50. Based on their histograms, the system may determine that both roads have probability 2/3 being greater than 50; thus both satisfy the predicate. However, it is clear that the distribution of road 19 is more likely to be less accurate, and consequently, the decision on road 19 is less reliable than that on road 20. Hence, query results can easily have false positives or false negatives without the awareness of users.

In general, there are mainly two possible reasons for which we get inaccurate distributions: (1) there are *time constraints*, e.g., the observations of an r.v. (i.e., samples) from which we learn a distribution are not produced fast enough for the queries to be answered in time; or (2) the observations are *expensive* to get (e.g., some experimental data in scientific applications).

Reason (1) applies to Example 1. The time constraints are that we have to get fresh data (as traffic condition is highly variable) and that we have to provide a timely answer for the query. This issue is common for sensor network and ubiquitous computing applications because of the limitations of the devices (e.g., power or location), because of the communication costs, and because of the time requirements on query results.

Our Contributions. We propose that an uncertain stream database should be accuracy-aware. Specifically, when a random variable (i.e., a distribution) appears in a query result (e.g., a probabilistic field in the SELECT list), the system also returns its accuracy information in the form of *confidence intervals* [11] of selected *parameters* of the distribution. In particular, for a histogram distribution, the *probability of each bin* becomes a confidence interval, instead of a fixed value. For other distributions such as Gaussian, the accuracy is embodied as a confidence interval on the *mean* value and a confidence interval on the *variance*. This is illustrated in Figure 2, where each interval indicated by a double arrow is a 95% confidence interval of the respective parameter.

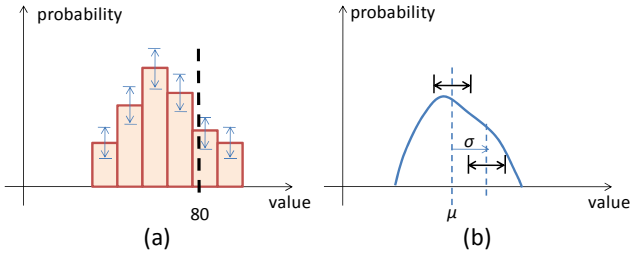


Fig. 2 Confidence intervals for (a) bin probabilities of histograms, and (b) $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ parameters of an arbitrary form of PDF

Clearly, the smaller an interval is, the more accurate the query result is. This gives a user a certain degree of confidence on the results. A user may use such accuracy information in a number of ways. For example, suppose the histogram distribution in Figure 2(a) is a *temperature* field in the result. Based on this result, the user can estimate the probability interval that the temperature is greater than 80 degrees. Similarly, she may also estimate the probability that the expectation is greater than 70 based on the interval of μ, etc. Clearly, such query results give a more complete picture of the state of the acquired data than the distributions alone. In fact, this enables *online computation*. When the intervals are sufficiently narrow to make a decision with enough confidence, we can stop acquiring raw data/samples, which is a slow or expensive process, as in Example 1.

We provide two ways to obtain the accuracy information: the *analytical method* and the *bootstrap method* from statistics [7, 5]. An analytical method has the advantage of little computational overhead, while a bootstrap method has the advantage of better accuracy for skewed distributions. For each of these two methods, we describe how to get query results' accuracy information, in which the confidence intervals can be either for random variable fields in a result tuple or for a result tuple's probability of being in the result set.

Finally, to aid decision making when the data is probability distributions with limited accuracy, we propose a new type of predicate – called a *significance predicate* – that is based on the concept of *hypothesis testing* in statistics [15]. In addition, we devise an algorithm called COUPLED-TESTS to control both false positive and false negative error rates in making decisions based on significance predicates. In summary, our contributions include:

- Defining and developing the concept of accuracy-aware uncertain stream databases.
- Proposing analytical methods to get accuracy information for query results.
- Devising bootstrap methods to obtain accuracy information.
- Proposing a new type of predicate, called significance predicate, for decision making and a COUPLED-TESTS algorithm to control error rates.
- Evaluating our work with a comprehensive set of experiments on both real and synthetic datasets.

The remainder of the paper is organized as follows. In Section II, we present the form of the accuracy information that we propose, followed by the analytical methods to obtain the accuracy information. We then devise an alternative method based on bootstraps to get accuracy information in Section III. In Section IV, we show the novel predicate for decision making using probability distributions with limited accuracy. We then perform extensive experiments in Section V. Finally, we discuss related work in Section VI, and conclude in Section VII.

II. Analytical Methods

A. Some Background

We briefly survey some background knowledge in order for the readers to better understand the paper.

Probabilistic data. We consider the general setting of an uncertain stream database where both *tuple uncertainty* and *attribute uncertainty* [18] may be present (it is also possible that the system has only one of them, as a special case). Let an uncertain stream database contains tuples $\{T_i\}$ $(i \geq 1)$. A tuple

T_i has a membership probability p_i, which is the probability that the tuple exists in the stream.

A tuple T_i has a set of attributes $\{A_j\}$ ($1 \le j \le a$). An attribute A_j of a tuple, in general, is a probability distribution, either continuous (e.g., Gaussians and histograms) or discrete. The distribution can be a single value with probability 1, in which case it is a traditional deterministic field.

As in previous work, following the *possible world semantics* [4], a SELECT query issued on a probabilistic database returns a result set with the same structures as formulated above (i.e., tuple uncertainty and attribute uncertainty). For example, consider this query: *SELECT ObjectID, Speed FROM stream WHERE Speed > 78* in which each field of the *Speed* attribute is a distribution (i.e., attribute uncertainty in the source data). Now, the result has tuple uncertainty because a tuple may have a probability between 0 and 1 to be in the result. The result also has attribute uncertainty because *Speed* is selected.

Confidence intervals and samples. We often need to estimate a parameter (e.g., mean) of a random variable. For that we calculate two numbers that define an interval that will enclose the estimated parameter with a high degree of confidence. The resulting random interval is called a *confidence interval*, and the probability that it contains the estimated parameter is called its *confidence coefficient* (or *confidence level*) [11]. For instance, if a confidence interval has a confidence coefficient equal to .95, we call it a 95% confidence interval. One often uses random samples of a random variable to obtain a confidence interval of a parameter. We use the following definition.

Definition 1 (sample, observation, statistic) [11]. *The random variables X_1, ..., X_n constitute a* **random sample** *on a random variable X if they are independent and each has the same distribution as X. We will abbreviate this by saying that X_1, ..., X_n are* **iid***; i.e., independent and identically distributed. We say that the* **sample size** *is n, and call an instance of each X_i ($1 \le i \le n$) an* **observation**. *Any function $T = T(X_1, ..., X_n)$ of the sample is called a* **statistic**. *The probability distribution of a statistic is called its* **sampling distribution**.

For instance, sample mean ($\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$) and standard deviation are both *statistics*. The probability distribution of the mean over all samples of size *n* is the *sampling distribution* of mean.

B. Accuracy Information

We first extend the basic probabilistic stream database model to include accuracy information. Informally speaking, we measure and represent accuracy by giving *confidence intervals* for some key *parameters* of a distribution. We have seen wide adoption of histogram representation of probability distributions in both the learning and the query processing phases due to its generality [2, 18, 12, 8]. Thus, we discuss the accuracy information specific to histograms, that for all distributions in general, and that for tuple probability (e.g., in a query result).

Histogram distributions. Recall that a histogram distribution has the form $\{(b_i,\ p_i) \mid 1 \le i \le b\}$, denoting that the probability of bucket b_i (which is a set of values) is p_i. We generalize this representation to be $\{(b_i, p_{i1}, p_{i2}, c_i) \mid 1 \le i \le b\}$, denoting that with confidence (i.e., probability) at least c_i (e.g., 95%), the *true* probability of b_i is in the interval $[p_{i1}, p_{i2}]$. That is, we extend the parameter p_i, a statistic which we call a *bin height*, from a fixed value to a confidence interval. Note that the intervals only describe the *marginal distributions* of bin heights. There is correlation among them, but we do not need to consider it in this work; conceptually, there is an implicit normalization step in the end which ensures that the sum of bin heights is 1.

All distributions. For an arbitrary distribution, we extend the parameters μ (expectation) and σ^2 (variance) to confidence intervals $(\mu_1,\ \mu_2, c_\mu)$ and $(\sigma_1^2, \sigma_2^2, c_\sigma)$. They denote that with confidence at least c_μ (c_σ, respectively) the *true* μ (σ^2) is between μ_1 (σ_1^2) and μ_2 (σ_2^2). Clearly, such accuracy information applies to histogram distributions too, although we typically use more specific accuracy information (i.e., confidence intervals on bin heights) for histograms.

The two forms of intervals and their usage are illustrated in Figure 2 and the paragraph thereafter in Section I.

Tuple probability. Finally, a result tuple's membership probability p can be considered as a one-bin histogram, in which the bin probability is the tuple probability. Hence we can get its confidence interval (p_1, p_2) with confidence level c as well. Implicitly, there is a second bin for this binary random variable (for the case that the tuple does not exist), but we do not need to consider it.

The following two lemmas are used in our analytical methods.

Lemma 1. *Suppose a histogram distribution $\{(b_i,\ p_i) \mid 1 \le i \le b\}$ is learned from a sample of size n. Then the accuracy of the distribution is represented by the following b confidence intervals (one for each bin): $\{(b_i, p_{i1}, p_{i2}, c_i) \mid 1 \le i \le b\}$, where p_{i1} and p_{i2} are as follows. If $np_i \ge 4$ and $n(1-p_i) \ge 4$, then p_{i1} and p_{i2} are:*

$$p_i \pm z_{(1-c_i)/2}\sqrt{\frac{p_i(1-p_i)}{n}} \tag{1}$$

Otherwise, they are:

$$\frac{p_i + \frac{1}{2n} z^2_{(1-c_i)/2} \pm z_{(1-c_i)/2}\sqrt{\frac{p_i(1-p_i)}{n} + \frac{z^2_{(1-c_i)/2}}{4n^2}}}{1 + \frac{1}{n} z^2_{(1-c_i)/2}} \tag{2}$$

Here $z_{(1-c_i)/2}$ is the upper $(1-c_i)/2$ percentile of the standard normal distribution.

Proof: Consider a single bucket i ($1 \le i \le b$). The number of observations that fall in this bucket follows a binomial distribution B(n, p_i'), where p_i' is the *true* probability of bucket b_i. We can then use the estimation of population proportion [15]. When $np_i \ge 4$ and $n(1-p_i) \ge 4$, the approximation of this distribution to a normal distribution is valid and we can deduce the

confidence interval as in (1). Otherwise, we can use the Wilson score interval [22] instead, giving (2). □

Example 2 (accuracy of histogram distributions). *Suppose we have a sample of size n* = 20 *and four buckets of a histogram, each of which has* 3, 4, 8, *and* 5 *observations in it, respectively. Thus,* p_1 = 0.15, p_2 = 0.2, p_3 = 0.4, *and* p_4 = 0.25. *We set* c_i = 0.9 *(i.e., 90% confidence) for all i's. Since* $np_1 = 3 < 4$, *we use* (2) *and get the probability interval for the first bucket as* (0.062, 0.322). *For the second bucket,* we get $0.2 \pm z_{0.05}\sqrt{\frac{0.2\times0.8}{20}} = 0.2 \pm 1.645 \times 0.089 = 0.2 \pm 0.15$ *by using* (1) *since* $np_2 = 4 \geq 4$, *which gives an interval* (0.05, 0.35). *In the same manner, we get* (0.22, 0.58) *for the third bucket and* (0.09, 0.41) *for the fourth bucket.*

Lemma 1 indicates that the length of the intervals is (roughly) inversely proportional to the square root of sample size. As the system collects more observations, the accuracy of distributions improves.

From the confidence interval estimation of mean and variance based on samples [11], we have:

Lemma 2 [11]. *Consider an arbitrary distribution that is learned from a sample of size n. Let* $\bar{y}$ *and* s *be the sample mean and standard deviation, respectively. Then, when* $n < 30$, *we have:*

$$\mu_1 = \bar{y} - t_{(1-c_\mu)/2} \cdot \frac{s}{\sqrt{n}}, \quad \mu_2 = \bar{y} + t_{(1-c_\mu)/2} \cdot \frac{s}{\sqrt{n}} \qquad (3)$$

where $t_{(1-c_\mu)/2}$ *is based on a Student's t distribution with* $(n-1)$ *degrees of freedom and is the upper* $(1-c_\mu)/2$ *percentile of this distribution. When* $n \geq 30$, *we have:*

$$\mu_1 = \bar{y} - z_{(1-c_\mu)/2} \cdot \frac{s}{\sqrt{n}}, \quad \mu_2 = \bar{y} + z_{(1-c_\mu)/2} \cdot \frac{s}{\sqrt{n}} \qquad (4)$$

For any n, we further have:

$$\sigma_1^2 = \frac{(n-1)s^2}{\chi^2_{(1-c_\sigma)/2}}, \quad \sigma_2^2 = \frac{(n-1)s^2}{\chi^2_{(1+c_\sigma)/2}} \qquad (5)$$

where $\chi^2_{(1-c_\sigma)/2}$ *and* $\chi^2_{(1+c_\sigma)/2}$ *are values of* χ^2 *that locate an area of* $(1-c_\sigma)/2$ *to the right and* $(1-c_\sigma)/2$ *to the left, respectively, of a chi-square distribution with* $(n-1)$ *degrees of freedom.*

Note that for very skewed distributions (i.e., far from normal distributions), the intervals given by (3) and (5) will be less accurate than our alternative method of using bootstraps (Section III), which we will evaluate in the experiments (Section V).

Another remark is that the results of Lemmas 1 and 2 are consistent with the intuition that a result distribution with a greater variance requires a larger sample size for it to be equally "accurate" as one with smaller variance. For instance, in Lemma 2, Equations (3), (4), and (5) all have s (sample standard deviation) as a factor of the interval length; for the same sample size, the distribution with a greater variance tends to have a greater s, thus longer intervals.

Example 3 (accuracy information for all distributions). *Suppose we have 10 iid observations of the current traffic delay of a road:* 71, 56, 82, 74, 69, 77, 65, 78, 59, *and* 80. *Then* $\bar{y} = 71.1, s = 8.85$. *We look at 90% confidence intervals* (*i.e.,* $c_\mu = c_\sigma = 0.9$). *Since* $n = 10 < 30$, *from* (3), *we have* μ_1 =71.1 − $t_{0.05}$(8.85/3.16). *With* 9 *degrees of freedom,* $t_{0.05}$ = 1.833, *which gives* μ_1 = 65.97. *Similarly,* μ_2 = 76.23. *For the variance, we have* σ_1^2 = 9×78.32/16.919 = 41.66 *and* σ_2^2 = 211.99.

C. Query Result Accuracy

While Lemmas 1 and 2 provide us accuracy of source data, our goal is to provide users with accuracy of query results. We first provide some insights on how the sample sizes of *input r.v.*'s (i.e., source data) translate to the *effective* sample size of an *output r.v.* (i.e., query result). This has a profound impact on the accuracy of result distribution.

Definition 2 (de facto sample). *Consider an r.v. Y in query result. Let* $Y = f(X_1, \ldots, X_d)$ *where* $X_1, \ldots, X_d$ *are input r.v.'s in source data, i.e., Y depends on* $X_1, \ldots, X_d$ *only. Let* $o_1, \ldots, o_d$ *be the observations for* $X_1, \ldots, X_d$, *respectively. We call* $f(o_1, \ldots, o_d)$ *a* **de facto observation** (*or, in short, a* d.f. observation) *of Y. A maximum set of independent de facto observations of Y form a* **de facto sample** (*or, in short, a* d.f. sample) *of Y.*

D.f. observations and d.f. samples for query results are analogous to observations and samples for source data. In essence, a d.f. observation is an instance that we could truly observe if the output r.v. were directly observable.

Lemma 3. *For an output r.v.* $Y = f(X_1, \ldots, X_d)$ *where* $X_1, \ldots, X_d$ *are input r.v.'s as in Definition* 2, *the d.f. sample size of Y is* $n = \min_{1\leq i\leq d} n_i$, *where* n_i *is the sample size of* X_i $(1 \leq i \leq d)$.

Proof: Suppose X_j $(1 \leq j \leq d)$ has the fewest number of observations among the d input r.v.'s, i.e., $n_j = \min_{1\leq i\leq d} n_i$. First, n is *at most* n_j. This is because two independent d.f. observations of Y cannot use the same observation of X_j; otherwise they would be dependent. Since there are only n_j observations of X_j, there cannot be more than n_j independent d.f. observations of Y.

Secondly, n is *at least* n_j. This is because each time we take a *distinct* observation o_i from each of the X_i's $(1 \leq i \leq d)$ and apply function f, we obtain a d.f. observation of Y. In total, we have n_j such d.f. observations and they are independent because they use independent observations of X_i's. Thus, n is at least n_j. Combining the two parts, we have $n = n_j$. □

Note that a result tuple's probability is determined by a boolean r.v. Y indicating whether the tuple exists in the result, and the f function in Definition 2 is a boolean function.

Example 4. *Let us look at a simple example. Consider the following query:*

SELECT (A+B)/2 FROM S WHERE C > 80

For a tuple t *in the result, suppose the A, B, and C attributes have sample sizes 15, 10, and 20, respectively. Then, for the field* $Y_1 = (A+B)/2$ *in the result, its d.f. sample size is*

$\min(15, 10) = 10$. *This result tuple* t *also has a probability that exists in the result, as determined by a boolean r.v.:*

$$Y_2 = \begin{cases} true, & if\ C > 80 \\ false, & otherwise \end{cases}.$$

That is, Y_2 *is a function on* C *only, and hence its d.f. sample size is 20.*

Based on Lemmas 1 and 2, we immediately have the following theorem in place:

Theorem 1. *Let* $\mathcal{D}$ *denote the distribution of a probabilistic field* Y *in a query result tuple as obtained by query processing over input source data. If* $\mathcal{D}$ *is a histogram distribution (an arbitrary distribution, respectively), then Lemma* 1 *(Lemma* 2*, respectively) determines its accuracy information, where we use the d.f. sample size of* Y *as the* n *value, and use the mean and standard deviation of* $\mathcal{D}$ *as* $\bar{y}$ *and* s*, respectively. In addition, the accuracy of a result tuple probability is based on Lemma* 1 *by treating it as a one-bin histogram, where the bin probability is the result tuple probability.*

The d.f. sample size required by Theorem 1 is obtained using Lemma 3. An example follows.

Example 5. *Let us continue on Example* 4*. Suppose for the same result tuple in Example* 4*, the distribution that we have learned for C based on its sample of size* 20 *informs us that* $\Pr[C > 80] = 0.6$*. Then, Theorem* 1 *shows that we can use Lemma* 1 *by treating the boolean r.v. in Example* 4 *as a one-bin histogram in which the bin probability is* 0.6*. Thus, based on Lemma* 1*, a 90% confidence interval of the tuple probability is:* $0.6 \pm z_{0.05}\sqrt{\frac{0.6 \times 0.4}{20}} = 0.6 \pm 0.18$*, i.e.,* $[0.42, 0.78]$*. Similarly, we can compute the accuracy of the field (A+B)/2.*

Since any output random variable in a query result can be expressed as a function of input random variables in source data, Theorem 1 gives us an analytical method to calculate accuracy information of each random variable in a query result, including a probability distribution in a field and tuple probability.

III. Bootstrap Methods

A bootstrap [7, 5] is a statistical technique that has the advantage of being widely applicable, and it is often more accurate than analytical method which are frequently based on assumptions of the underlying distributions. A bootstrap involves a little extra computation; but the recent fast advancement in processors (including multi-cores) and cluster/cloud computing has increased computing power tremendously, which makes the bootstrap a very useful technique. In this section, we devise an algorithm that uses bootstraps to obtain the accuracy information which we have proposed in the previous section. In the experiments (Section V), we actually find that our algorithm not only gives tighter confidence intervals than analytical methods, but it also incurs little extra computational overhead. We start with some background on bootstraps for the reader to understand our algorithm better.

A. Bootstraps

In statistics, the *bootstrap* (a.k.a. *resampling*) is a way of finding the sampling distribution from just *one* sample [7, 5]. There are two steps:

(1) Resampling. Create many *resamples* by repeatedly sampling with replacement from one random sample. Suppose the sample is of size *n*. A resample is created by drawing an observation (with replacement) uniformly at random from the *n* observations of the sample, and repeating this process *n* times. Thus, the resample also has size *n*.

(2) Bootstrap distribution. Over the many resamples created in (1), we compute the distribution of the statistic in question. This is called the *bootstrap distribution*. It is shown that a bootstrap distribution has approximately the same *shape* and *spread* as the sampling distribution, but it may be biased (its *center* is based on the original one sample). Thus, we can use the bootstrap distribution to derive a confidence interval for the statistic, instead of using the sampling distribution.

Example 6. *We illustrate resampling in Figure* 3*. This example comes from* [16] *and is based on real-world data (Verizon's repair times). The box on the top contains an original (true) sample of size 6 (i.e., 6 observations). Each of the three boxes at the bottom is a resample (sampling from the top box). Some values from the original are repeated in the resamples because each resample is formed by sampling with replacement. Each resample is also of size 6. We calculate the statistic of interest—the sample mean in this example—for the original sample and each resample. The means in the resamples give the* bootstrap distribution*, while the mean in the original sample follows a* sampling distribution*. The bootstrap distribution has approximately the same shape and spread as the sampling distribution, but has a biased center.*

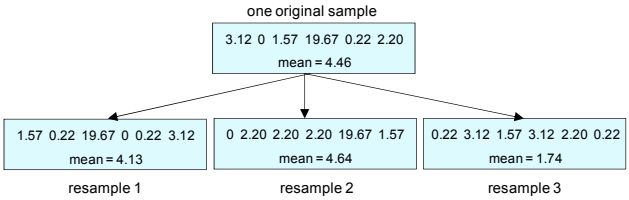


Fig. 3 Illustrating bootstrap resamples

B. Using Bootstraps to Get Accuracy Information

Querying processing algorithms on uncertain streams can have two categories:

- The ones that are based on Monte Carlo algorithms using samples (e.g., [13]);
- Others that do not use samples, but directly operate on probability distributions (e.g., using Gaussian Mixture Models [19]).

For the first category, the query processing algorithm already provides a sequence of values for an output random variable. For the second category, we directly get a distribution for a random variable in query result; thus we sample from

this distribution and also get a sequence of values. In both cases, our bootstrap method will operate on this sequence of values and obtain accuracy information for the probability distribution of the output random variable – confidence intervals of selected parameters, namely bin heights, μ, and σ^2. The algorithm is shown below.

Algorithm BOOTSTRAP-ACCURACY-INFO ($v[\cdot], n, \alpha$)

Input: $\{v[i] \mid 0 \le i \le m-1\}$: a sequence of values of an output random variable Y as a result of query processing;
n: the d.f. sample size of Y;
α: confidence level of the intervals

Output: accuracy information of Y

1: $r \leftarrow \left\lfloor \frac{m}{n} \right\rfloor$ *// number of d.f. resamples*
2: **for each** $i \leftarrow 0$ *to* $r-1$ **do** *// each resample*
3: **for each** $j \leftarrow 0$ *to* $n-1$ **do**
4: $o[j] \leftarrow v[i \cdot n + j]$ *// n observations*
5: **end for**
6: **for each** $k \leftarrow 1$ *to* b **do** *// for each bin in histogram*
7: $p_k[i] \leftarrow \frac{|\{o[j] \mid o[j] \in b_k\}|}{n}$ *// frequency of bin k*
8: **end for**
9: $\bar{y}[i] \leftarrow \frac{1}{n} \cdot \sum_{j=0}^{n-1} o[j]$ *// sample mean*
10: $s^2[i] \leftarrow \frac{1}{n-1} \cdot \sum_{j=0}^{n-1} (o[j] - \bar{y}[i])^2$ *//sample variance*
11: **end for**
12: **for each** $k \leftarrow 1$ *to* b **do** *// for each bin in histogram*
13: **output** the α interval within $\{p_k[i] \mid 0 \le i \le r-1\}$ based on percentiles
14: **end for**
15: **output** the α intervals within $\{\bar{y}[i] \mid 0 \le i \le r-1\}$ and $\{s^2[i] \mid 0 \le i \le r-1\}$, respectively, based on percentiles

In line 1 of the BOOTSTRAP-ACCURACY-INFO algorithm, we start with grouping the m values of Y into r groups, each of which has n (d.f. sample size) values (m is sufficiently large so that the confidence intervals as output by the algorithm converge). Thus, each group is a *d.f. resample*. For completeness, the algorithm computes the intervals for both histogram and arbitrary distributions; but in reality one only needs the confidence intervals specific to the distribution type. Lines 6 to 10 calculate the statistics (bin heights, mean $\bar{y}$ and variance s^2) within each resample. Then, over all r resamples, we have a distribution for each of these statistics, which is the *sampling distribution* (Definition 1). Finally, we obtain the α intervals (i.e., between $100(1-\alpha)/2$ and $100(1+\alpha)/2$ percentiles) for each bin height, $\bar{y}$ and s^2 (lines 12 to 15). Let us look at an example.

Example 7. *Suppose* $n = 15$ *and* $m = 300$*, giving* $r = 20$ *in line 1. Thus, the BOOTSTRAP-ACCURACY-INFO algorithm works with 20 d.f. resamples, each having size 15. For each statistic in question (i.e., a bin height, mean, or variance), we calculate a value for each d.f. resample, and these values form a sampling distribution over all 20 d.f. resamples. For instance, consider how we get accuracy information for* μ *(mean). In line 9, we calculate the sample mean, one for each of the 20 d.f. resamples. The sample mean of a resample (*$\bar{y}[i], 0 \le i \le 19$*) is the average of the 15 values within the resample. These 20 values (*$\bar{y}[i], 0 \le i \le 19$*) form a distribution* $\mathcal{D}$*. Suppose the input parameter* $\alpha = 0.9$ *(confidence level). Then we get two estimated values* μ_1 *and* μ_2 *from* $\mathcal{D}$ *such that the area (i.e., probability) to the left (right, respectively) of* μ_1 *(*μ_2*, respectively) is* $\frac{1-\alpha}{2} = 0.05$*. Then, the 90% confidence interval for* μ *returned by the algorithm is* $[\mu_1, \mu_2]$*. The confidence intervals for each bin height and variance are calculated similarly.*

Based on Lemma 4 below, the Theorem 2 that follows establishes the correctness of our bootstrap algorithm.

Lemma 4. *As in Definition 2, let* $Y = f(X_1, \ldots, X_d)$*, where* $X_1, \ldots, X_d$ *are input r.v.'s, and are arranged in an order such that* $n_1 \le n_2 \le \cdots \le n_d$*. Here* n_i *denotes the sample size of* X_i $(1 \le i \le d)$*. Let* n *be the d.f. sample size of* Y*. Then* Y *has* $c = \prod_{i=2}^{d} \frac{n_i!}{(n_i-n)!}$ *d.f. samples, each of which has size* n*.*

Proof: From Lemma 3, we have $n = n_1$. Now we take the vector of all n observations from the sample of X_1, and take a vector (i.e., a permutation) of n observations from each of $X_2, \ldots, X_d$'s samples, we get a matrix M of input values with n rows and d columns. If we execute the query n times, using a row of M each time for the d input fields, we get n iid d.f. observations of Y, which form a d.f. sample of Y. Because the n input rows are iid, so are the d.f. observations of Y. It is then clear that all together there are:

$$c = \prod_{i=2}^{d} P(n_i, n) = \prod_{i=2}^{d} \frac{n_i!}{(n_i - n)!}$$

matrixes like M, as we are taking permutations from all but the first input field. Thus, there are c d.f. samples in total, each of which has size n. □

Theorem 2. *Algorithm BOOTSTRAP-ACCURACY-INFO returns correct confidence intervals.*

Proof: The sequence of values of the output random variable that are fed into the BOOTSTRAP-ACCURACY-INFO algorithm can be considered as d.f. resamples based on the c d.f. samples as shown in Lemma 4. Thus, BOOTSTRAP-ACCURACY-INFO is a concurrent bootstrap from all c d.f. samples. Therefore, this bootstrap produces a probability distribution of a statistic (i.e., a bin height, or the mean/variance) that is a mixture distribution of multiple simple bootstrap distributions, each of which is based on a single d.f. sample. Bootstrapping from multiple samples is also an effective and valid method to obtain confidence intervals as rigorously studied in statistics (e.g., [7]). Informally, the intuition is that the biases of simple bootstrap distributions are somewhat canceled out through mixing them together while the percentile confidence interval of the mixture distribution is at least as wide as that of a single simple bootstrap distribution. □

In Section V, using both real and synthetic datasets, we evaluate the quality of the obtained confidence intervals.

IV. SIGNIFICANCE PREDICATES

When a stream database system is aware of the accuracy of the probability distributions that it learned from the environment, decision making using these distributions must also be based on this accuracy information. In database applications, decision making is through *predicates*. Unfortunately, none of the existing predicates in SQL or previously proposed probabilistic database systems considers the accuracy information.

In this section, we propose a novel type of predicate just for this purpose. The new type of predicate determines whether the truth of a statement about a probability distribution in the system is statistically significant (i.e., above a significance level), thus having the name *significance predicates*. The truth of a statement is statistically significant if it is unlikely to have occurred by chance alone. Significance predicates are based on the concept of hypothesis testing in statistics, which will be briefly surveyed next.

A. Background on Hypothesis Testing

We can *make decisions* about the truth or falsity of some statement about the parameters by *hypothesis testing*. Hypothesis testing consists of four elements [15]:

- *null hypothesis*, H_0, about one or more population parameters,
- *alternative hypothesis*, H_1, that we will accept if we decide to reject H_0,
- *test statistic*, computed from sample data, and
- *rejection region*, indicating the values of the test statistic that will imply rejection of the null hypothesis.

The rejection region is usually based on a pre-determined *significance level* α, such that the false positive rate (i.e., accepting H_1 when actually H_0 is true) is below α.

B. Basic Significance Predicates in Our System

Let us first look at a motivating example.

Example 8. *Consider two random variable fields X and Y of the attribute "temperature" of a stream. X's distribution is learned from a raw sample of size* 5*:* {82*,* 86*,* 105*,* 110*,* 119}*. Suppose Y's distribution has the same mean, and is learned from a raw sample of a much larger size,* 100*, among which* 40 *observations are below the value* 100 *and* 60 *observations are above.*

*Now consider two predicates P*1 *and P*2*. P*1 *is a probability threshold predicate* [3]*: temperature* $>_{0.5}$ 100*, requiring that with probability at least* 0.5*, temperature is greater than* 100*. Since X and Y's distributions learned by the database should be faithful to their raw samples, they both have a probability of about* 0.6 *being greater than* 100*. Thus they both satisfy P*1*. P*2 *is* E(*temperature*) > 97*, where* E(·) *denotes the expectation of a random variable. Similarly, X and Y both satisfy P*2*.*

The predicate results in Example 8 clearly overlook the fact that an inference based on the *X*'s distribution and that based on *Y*'s distribution have very different confidence levels. We have much more confidence on the result based on *Y*'s distributions than on *X*'s distribution, which is learned through a sample of only five observations – the predicate result is more likely to be by chance.

We propose to add three most common significance predicates into an uncertain stream database system as built-in functions. They are described as follows.

- ***mTest.*** Its syntax is:

  $boolean\ mTest(X, op, c, \alpha)$

  where X is a probabilistic field, op is an operator, c is a constant, and α is the significance level (Sec. IV-A). $mTest$ is short for *mean test*. It determines the relationship between the mean of X and a constant. In the hypothesis testing [15], the null hypothesis H_0 is: E(X) = c, i.e., the expectation of X is c, and the alternative hypothesis H_1 is: E(X) *op* c, where *op* is one of "<", ">", and "<>".

- ***mdTest.*** Its syntax is:

  $boolean\ mdTest(X, Y, op, c, \alpha)$

  where X and Y are two probabilistic fields whose means are to be tested, and op, c, and α are the same as in $mTest$. $mdTest$ is short for *mean difference test*. It determines the relationship between the means of two fields. The null hypothesis H_0 is: E(X) − E(Y) = c, while the alternative hypothesis H_1 is: E(X) − E(Y) *op* c, where *op* is one of "<", ">", and "<>". The most common usage is $c = 0$, which just compares E(X) with E(Y).

- ***pTest.*** Its syntax is:

  $boolean\ pTest(pred, \tau, \alpha)$

  where $pred$ is an arbitrary predicate as in a deterministic database, τ is a probability threshold, and α is the significance level. $pTest$ is short for *probability test*. It determines whether the probability of the truth of a predicate is above a threshold. The null hypothesis H_0 is: $\Pr[pred] = \tau$, while H_1 is: $\Pr[pred] > \tau$.

Probabilistic threshold queries are commonly used in the probabilistic database literature (e.g., [17]). A probabilistic threshold query is simply the alternative hypothesis H_1 in a $pTest$: $\Pr[pred] > \tau$. A $pTest$ adds another level, the significance level α which controls false positive rates (Sec. IV-A).

Example 9. *We use pTest and mTest for the queries in Example* 8*. We can turn P*1 *into pTest("temperature > 100", 0.5, 0.05), which means that we test the original probability threshold query temperature* $>_{0.5}$ 100 *with a significance level of 5%. That is, the false positive rate* (*i.e., mistakenly accepting* H_1) *is no more than 5%. Similarly, we can change P*2 *to mTest*(*temperature, ">",* 97*,* 0.05)*. It can be easily verified that with these new predicates, only the field Y would satisfy them, but X would not.*

For predicate evaluation, the *mTest* and *mdTest* follow the *population mean tests* in statistics [15]. Note that the query evaluation of these significance predicates uses hypothesis testing, which is very efficient by directly operating on the probability distributions using the accuracy information. We will verify this in the experiments. The *mdTest* is useful, for example, when we make assertions on whether the mean of

one field is greater than another (i.e., whether the difference is greater than 0). Likewise, the *pTest* is based on the *population proportion test*.

C. Significance Predicates with Coupled Tests

While the basic significance predicates serve our purpose, there is a shortcoming. Hypothesis testing has type I errors (i.e., false positives) and type II errors (i.e., false negatives). The significance level α in the tests only controls the false positive rates. For example, for the *mTest*($t1.c1$, ">", 97, 0.05) in Example 9, only when H_1 is actually false, are we sure that the error rate is no more than 0.05. That is:

$$\Pr[mTest\ returns\ TRUE \mid E(t1.c1) \leq 97] \leq 0.05$$

However, we have no control over the false negative rate, i.e., $\Pr[mTest\ returns\ FALSE \mid E(t1.c1) > 97]$. It can be large, depending on the sample size and the actual $E(t1.c1)$ value, etc.

To cope with this problem, we propose a technique called *coupled tests*. Coupled tests are two correlated tests run within a significance predicate in order to control both the false positive rate and the false negative rate of the predicate. The key idea is as follows. Suppose originally we have an *mTest* to determine if $E[X] > c$. We couple this test with its inverse test, which is essentially an *mTest* that determines if $E[X] < c$. *The original test rejects if this inverse test accepts*. Therefore, the original *mTest*'s false negative error rate is the same as this inverse test's false positive rate, which is controlled by the significance level of the inverse test. In this way, we can control both the false positive and the false negative error rate of the original test. The algorithm below achieves our goal.

Algorithm COUPLED-TESTS (P, α_1, α_2)

Input: P: a basic significance predicate;
α_1: maximum false positive error rate;
α_2: maximum false negative error rate
Output: an answer from $\{TRUE, FALSE, UNSURE\}$

```
1:  T1 ← P.test        // the original hypothesis test
2:  T2 ← T1            // a copy of the same test
3:  if T1.op =′<>′ then
4:      T1.op ←′<′
5:      T2.op ←′>′
6:      T1.α ← α1/2
7:      T2.α ← α1/2
8:  else
9:      T2.op ← inverse(T1.op)   //'>' and '<' are inverse
10:     T1.α ← α1
11:     T2.α ← α2
12: end if
13: result ← run T1
14: if result = TRUE then
15:     return TRUE
16: else
17:     result ← run T2
18:     if result = TRUE then
19:         return (P.test.op =′<>′) ? TRUE : FALSE
20:     else
21:         return UNSURE
22: end if  end if
```

Since all three basic significance predicates described in Section IV-B have a hypothesis test component, we use $P.test$ (line 1) to denote that test. The algorithm behaves differently if the *op* parameter is '<>' (lines 3 to 7); otherwise T_1 keeps the original *op* while T_2 uses its *inverse* ('>' and '<' are inverse of each other), as in line 9. Again in line 19, the algorithm behaves differently (returning $TRUE$) if the original *op* is '<>'; otherwise it returns $FALSE$. The hypothesis tests (including H_0 and H_1) are the same as in the basic significance predicates.

Note that we have changed the return value of the predicate from two states to three states ($TRUE$, $FALSE$, $UNSURE$); only $TRUE$ and $UNSURE$ may be returned when the *op* is '<>'. Theorem 3 below shows the correctness of the algorithm.

Theorem 3. *Algorithm COUPLED-TESTS has a false positive rate of no more than α_1 and a false negative rate of no more than α_2.*

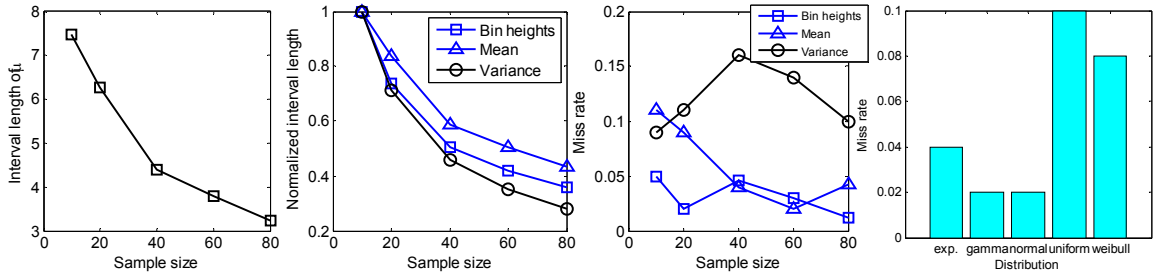
Proof: Let us first consider the case when $P.test.op$ is either $'>'$ or $'<'$. T_1 is the original hypothesis test and has a type I error (false positive) rate of no more than α_1. Similarly, T_2 has a false positive rate of no more than α_2. Since $T_2.op$ is the inverse of $P.test.op$, and when T_2 returns $TRUE$, COUPLED-TESTS returns $FALSE$, it follows that the false negative rate of COUPLED-TESTS is no more than α_2. We next consider the case when $P.test.op = '<>'$. In this case, T_1 and T_2 have their op's as $'<'$ and $'>'$ respectively, and they both have a significance level of $\alpha_1/2$ (lines 3 to 7). Since the algorithm does not return $FALSE$ in this case, the false negative rate is 0. Moreover, COUPLED-TESTS returns $TRUE$ whenever either T_1 or T_2 returns $TRUE$, and thus, it has a false positive error whenever either T_1 or T_2 has a false positive error. Hence, due to union bound [11], the false positive error rate is no more than $\frac{\alpha_1}{2} + \frac{\alpha_1}{2} = \alpha_1$. This concludes the proof. □

Power of Coupled Tests. Note that there is still the notion of the *power* (denoted as γ) [11] of the coupled tests for P, although it is not $1 - \beta$ (as in a single hypothesis test). For example, by definition, the power of *mTest* (X, ">", c, α_1, α_2) is $\Pr[return\ TRUE \mid E(X) > c]$. As in standard statistics, γ is usually a function. As an example, for mTest, the power function $\gamma(\mu)$ is a function on μ, the actual expectation of the probabilistic field. Clearly, the power of coupled tests is closely related to the number of $UNSURE$ return values; the fewer $UNSURE$ values, the more powerful the tests are. We will study the power function experimentally in Section V.

V. Experiments

In this section, we study the following problems through a systematic experimental evaluation:

- How does the accuracy information, as measured by confidence intervals of distribution parameters and derived through analytical methods, change when we vary the sample sizes?



(a) (b) (c) (d)

Fig. 4 Experimental results: (a) sample size vs. interval length of μ, (b) n vs. normalized intervals, (c) miss rates vs. n, and (d) miss rates for various distributions

- What are the miss rates of the confidence intervals? Are these confidence intervals robust when we vary the underlying distributions?
- How does the accuracy information acquired via bootstraps compare with that acquired via analytical methods?
- What is the impact to stream system throughput when using analytical or bootstrap methods to get accuracy information?
- What are the error rates and powers of the significance predicates with and without the coupled-test technique? What is the performance overhead of significance predicates?

A. Datasets and Setup

We perform our experimental evaluations on the following datasets.

- A real-world dataset collected by the CarTel project team [24]. It consists of measurements of actual traffic delays on roads in the greater Boston area performed by the CarTel vehicular testbed, a set of 28 taxis equipped with various sensors and a wireless network. This application is described in Example 1.
- Some synthetic datasets generated using the R statistical package [25]. We experiment with five types of common distributions: *exponential* ($\lambda = 1$), *Gamma* ($k = 2$, $\theta = 2.0$), *normal* ($\mu = 1$, $\sigma^2 = 1$), *uniform* (0 to 1) and *Weibull* ($\lambda = 1$, $k = 1$).

With synthetic datasets we can test our findings over more diverse distributions. We implemented all algorithms in the paper. The experiments were run on a 2.4 GHz Intel Core 2 Quad CPU machine with 1 GB memory.

B. Accuracy Information via Analytical Methods

We examine the relationship between the size (n) of the sample from which a distribution is learned and the confidence interval lengths in accuracy information. We first use the road-delay real dataset. We randomly pick 100 road segments for which we have sufficiently large sample sizes (i.e., at least 600) within a short time period. We consider the distribution that we get from the complete sample of such a road segment as its *true* distribution. We next pick a sample of a small size (e.g., $n = 20$) uniformly at random without replacement from the original large sample. Then we can not only obtain a distribution bundled with its accuracy information using this small sample, but we can also verify the validity of the confidence intervals since we do have the true distribution.

Figure 4(a) plots the relationship between sample size n and 90% confidence interval length of the μ parameter of the distribution. In order to see all three statistics bin heights, mean, and variance in the same plot, in Figure 4(b), an interval length is normalized through dividing it by the length when $n = 10$. Figures 4(a) and 4(b) show a decrease of interval lengths as n grows, for all three kinds of statistics − bin heights, mean, and variance. We next examine the qualities of these confidence intervals.

Clearly, the validity of a confidence interval is determined by whether the true value is really contained in the interval. When it is not contained, we call it a *miss*. Thus we use *miss rate* as a metric (clearly, the *interval length* in the previous experiment is also a metric since shorter intervals are more useful).

Figure 4(c) shows the miss rates of the three types of accuracy information under different sample sizes. We can see that *bin heights* have the lowest miss rates, while *variance* has the highest. The *mean* parameter, however, has higher miss rates when n is small. The reason is that the analytical method Lemma 1 (for bin heights) does not assume a particular shape of the underlying distributions. In Lemma 2, the interval lengths for μ when $n < 30$ and those for the variance all assume that the underlying distributions are (at least approximately) normal. When the actual population distribution deviates from this assumption, the validity of the confidence intervals decreases.

We next experiment on synthetic datasets. We use the R statistical package [25] to generate samples for five types of common distributions: *exponential*, *Gamma*, *normal*, *uniform* and *Weibull*. Their parameters are described in Section V-A. Figure 4(d) shows the average miss rates for the intervals over three kinds of statistics (bin heights, mean, and variance) when $n = 20$. We can see that with all five types of distributions, the miss rates are relatively low (recall that they are 90% confidence intervals; thus there is some inherent error). This verifies the overall accuracy of analytical methods.

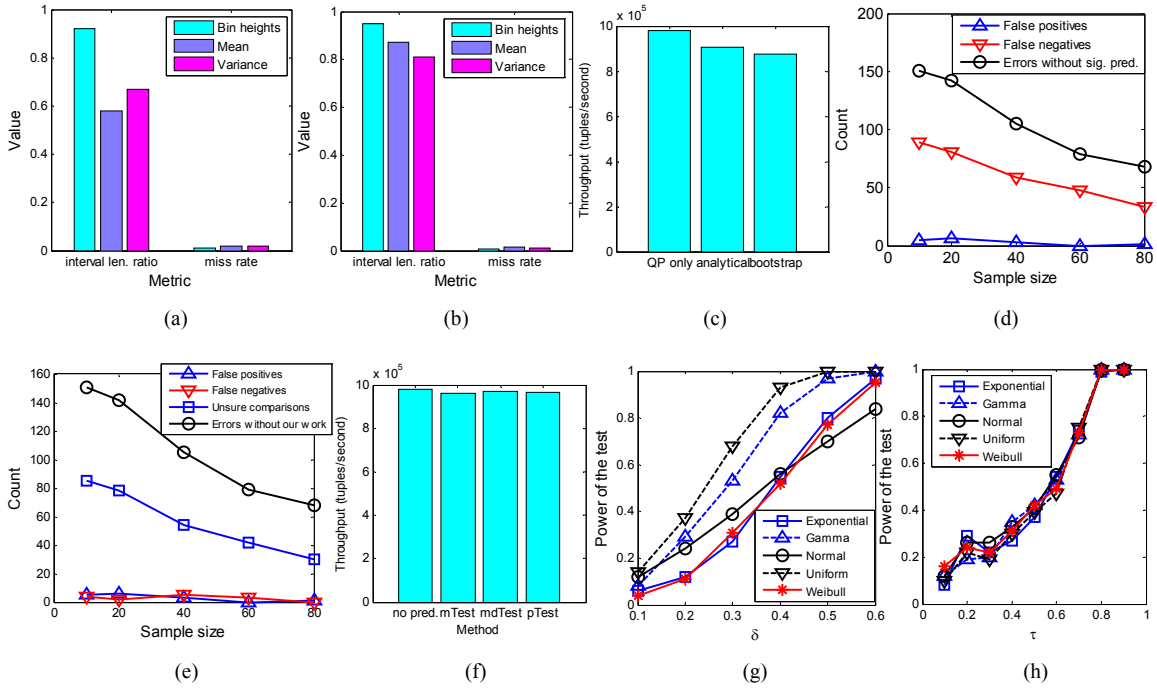


Fig. 5 More experimental results: (a) bootstraps compared to analytical results, (b) Gaussian results, (c) maximum throughput impacts, (d) errors of a single significance predicate, (e) result of coupled tests, (f) maximum throughputs, (g) power vs. δ (mTest), and (h) power vs. τ (pTest).

C. *Comparison of Bootstrap and Analytical Methods, and Their Throughputs*

In this set of experiments, we compare the accuracy information of the bootstrap methods and the analytical methods in query results. We also evaluate the execution overhead in producing the accuracy information. The overhead is measured through the impact on the stream system throughput. Using the road delay dataset, we issue queries that ask for *the total delays of a number of routes*. On average, there are around 20 road segments per route. Different road segments may have different sample sizes.

Secondly, using the synthetic dataset, we test diverse distributions and arbitrary queries. We generate a random query (expression) by assigning equal probabilities to six operators +, −, ×, /, SQRT(ABS(·)), and SQUARE. Together with the five types of distributions described in the previous experiment, the query selects the result of the random expression.

We first evaluate whether bootstraps give more accurate and better confidence intervals than analytical methods. For this, in Figure 5(a), we show the average ratio (averaging over bin heights, mean, and variance) between the confidence interval lengths of bootstraps and those of analytical methods. We also show the miss rates of the confidence intervals of bootstraps. We obtain similar trends from the queries on the road-delay data and from the random queries on synthetic data. We thus show the average results from both datasets. Figure 5(a) indicates that bootstraps give slightly shorter confidence interval lengths for bin heights. For mean and variance, however, bootstraps show significant advantage, with confidence intervals shortened by about 40%. The miss rates, on the other hand, stay very low for bootstraps. The considerable improvement on mean and variance is due to the normality assumption of the analytical methods, which does not always hold true. Bootstraps are always robust to the skewness of the underlying query result distributions.

To verify whether our analysis is correct, we also generate random queries for which we know the result is normal. Specifically, in the random generation process, we limit the distributions to be only from normal distributions and the operators to be only + and −. Figure 5(b) shows the confidence interval length ratios between bootstraps and analytical methods. We can see that the difference between the two methods on mean and variance is less here, with bootstraps having around 20% shorter intervals. This is because when the underlying result distributions are indeed normal, analytical methods on mean and variance are also very accurate.

We now examine the execution overhead of bootstraps and analytical methods for getting accuracy information. In a stream context, this is reflected in the impact on maximum throughputs with which the system is able to handle incoming stream tuples. In order to measure the maximum throughout, we synthetically generate data tuples. For each item, we generate 20 data points and the query processor learns a Gaussian distribution from them. The query is a simple count-based sliding window AVG query (e.g., [9]) with a window size of 1000. Since the inputs are Gaussians, the query processor can compute the AVG result as a Gaussian distribution. For each such query result, we compute its accuracy information (on μ and σ^2) through analytical methods and through bootstraps. Figure 5(c) shows the throughputs of (1) query processing

only, (2) query processing and getting accuracy information via analytical methods, and (3) query processing and getting accuracy information via bootstraps. We can see that analytical methods have less overhead than bootstraps, but the throughput changes due to the overhead of applying analytical methods and bootstraps are both relatively small. Note that we have used a very simple query. A more expensive query would only make the throughput impact of getting accuracy information less significant. These experiments also inform us that bootstraps in general give better quality accuracy information, while only having slightly more overhead compared to analytical methods.

D. Significance Predicates

We now study significance predicates using both real and synthetic datasets. For the road-delay real dataset, we choose 100 pairs of routes and use *mdTest* predicates, i.e., a query asks whether the mean of the first route's delay is greater than that of the second route. We intentionally choose pairs of routes whose *true* mean values are close, which makes comparisons using small sample sizes more challenging. In other words, we expect to see a large number of errors when sample sizes are small. For 100 pairs of routes, we perform 200 comparisons, with and without significance predicates (mdTest). In the first 100 tests, we make the null hypothesis H_0 to be (actually) true and test false positives (type I errors) with various sample sizes. This is done through arranging the order of each pair of routes such that $E(X) \leq E(Y)$ yet the predicate is "$E(X) > E(Y)$". In the second 100 tests, we make the alternative hypothesis H_1 to be true to test false negatives. This is similarly done through arranging the order of each pair such that $E(X) > E(Y)$ instead. For comparison, we also check the number of errors when we do not use significance predicates (i.e., merely $E(X) > E(Y)$ as in previous work).

We show the results in Figures 5(d) and 5(e). In Figure 5(d), we do not use our *coupled tests* technique, but only use a single hypothesis test. We set the significance level of the test $\alpha = 0.05$. The figure verifies that the false positive rate is below 5%. However, with a single test, the false negative rate is not really under the control of the user; it is a function of the sample size and the actual $E(X)$ and $E(Y)$ values. From Figure 5(d), we also see that, as sample size increases, errors decrease.

By contrast, in Figure 5(e), we show the result with our *coupled tests*. The user can specify both types of error rates (in this case, both 0.05), and when the system cannot make a decision that respects these error rates, it returns the result UNSURE. We can see that the number of unsure comparisons decreases as sample size increases, and that the actual number of errors strictly follows the error rate specification.

Following the setup in Section V-C (Figure 5(c)) for testing stream throughput, we issue the same count-based sliding window AVG query, followed by significance predicates with coupled-tests using *mTests* (whether the mean is greater than a value), *mdTests* (whether the mean is greater than the one in the previous window), and *pTests* (whether the average is greater than a constant with probability at least 0.8), respectively. Figure 5(f) shows the throughputs of no significance predicates (first bar) and each of the other three cases. It is clear that significance predicates have little overhead (even less than computing the accuracy information) since they are efficient hypothesis testing on the distributions.

We next use synthetic datasets, with the same five types of distributions, and study *mTest* and *pTest*. We have *mTest* (X, *op*, c, α_1, α_2), where *op* is ">", $c = (1 + \delta)\mu$, and $\alpha_1 = \alpha_2 = 0.05$. Here μ is the true expectation of X and we use a sample of size 20. Since $E(X) > (1 + \delta)\mu$ is false for any $\delta > 0$, any error of this test is a false positive. We have verified that the error rates are well below 0.05, and omit the figure.

An important aspect of significance predicates is the *power* of the test. Even with our test-coupling technique, the power of the test is directly related to the number of UNSURE results (the higher the power, the fewer the unsure results). We now observe the relationship between the power of the COUPLED-TESTS algorithm and the δ value in the mTest within. The result is in Figure 5(g). As δ increases, the difference between the actual $E(X)$ and the value to be compared with increases, which makes the test easier. Thus, we see an increase in the power. Also observe the interesting fact that the test power increases faster with the uniform and Gamma distributions. The reason for the uniform (0, 1) distribution is because it has a very small variance (1/12) compared to other distributions which have variances of at least 1. The reason for the Gamma distribution ($k = 2$, $\theta = 2.0$) is because the probability decreases much faster as δ increases than other distributions; thus it is easier to determine the test result as δ increases.

We finally use *pTest* (*pred*, τ, α_1, α_2), where *pred* is simply $X > v$ and $\alpha_1 = \alpha_2 = 0.05$. To produce false positives, we choose v such that the *true* Pr $(X > v)$ is $\tau\ (1 - \delta)$. Likewise, for false negatives, we choose v such that the *true* Pr $(X > v)$ is $\tau\ (1 + \delta)$. We fix $\delta = 0.3$ and vary the τ parameter. We find that the two types of error rates are well below α_1 and α_2 values. We study the relationship between the power of the test and the τ parameter, with the result shown in Figure 5(h). As τ increases, the difference between the true Pr $(X > v)$ and τ increases, which makes the comparison an easier decision (i.e., higher power). Moreover, because the decision of Pr $(X > v) \leq \tau$ (using a sample) is based on quantiles, it is independent of the actual distribution. Thus, we see that, for all five distributions, the power of the test increases at about the same rate.

VI. Related Work

There have been a number of projects on probabilistic databases, including Orion [18], MystiQ [4], Trio [21], and MCDB [12]. Previous work that deals with uncertain data streams (like ours) includes [3, 6, 8, 12, 19]. However, to the best of our knowledge, none of that work considers the accuracy of the probability distributions acquired; they all assume that distributions are already obtained in the first place and completely trust them.

A related concept in AI is the so-called imprecise probabilities [20, 23]. The basic idea there is to use a probability interval to represent the really available knowledge, and provides tools to model and work with weaker states of information. We study this in databases and propose a novel technique of

using *confidence intervals* on the *parameters* of distributions to model the uncertainty of distributions, which has not been explored in AI.

Koch and Gotz [14, 10] study the reliability of query results, which bears some similarity to our work. However, they have a different motivation, and solve a different problem with a different approach. Their goal is to provide a compositional framework for queries over unreliable data resulted from *approximate query processing* (e.g., Monte Carlo algorithms). [14, 10] use a *discrete* model that only deals with tuple confidence. By contrast, we solve the accuracy of probability distributions and decision making starting from *learning distributions* based on samples to query results, as required by many data stream and sensor networks applications. In addition, [14, 10] only consider the basic positive relational algebra (with an extension of the repair-key construct), while we handle arbitrary (extended) SQL queries, which are more powerful and are suitable for stream databases.

Finally, Perez et al. study the evaluation of probabilistic threshold queries in MCDB [17]. However, like others, they assume the input probability distributions are accurate and get samples from them for query processing, which is completely different from the problem that we solve in this work.

VII. Conclusions and Future Work

While probability distributions are introduced into stream databases to make noisy data more useful, what is lacking is the important information of how accurate such distributions are. We propose and perform a first study on an accuracy-aware uncertain stream database in a general setting that can have continuous distributions. We gauge the accuracy of data and query results both analytically and through efficient bootstraps. Moreover, we propose a new type of predicate for decision making in this context. We have performed extensive experiments on real and synthetic datasets to validate our schemes and algorithms.

As future work, we plan to study the idea of using samples of different weights to quantify the accuracy of probability distributions that are learned from them. For instance, observations that are obtained more recently can have more weights in determining the accuracy information.

Acknowledgment

This work was supported in part by the NSF, under the grant IIS-1017452. The authors gratefully acknowledge Sam Madden and Arvind Thiagarajan for the CarTel dataset, and the anonymous referees for their insightful comments.

References

[1] R. Agrawal et al. The Claremont report on database research. In *ACM SIGMOD Record*, V.37, 3, 2008.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[3] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD,* 2003.

[4] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[5] Davison, A., Hinkley, D. *Bootstrap Methods and Their Application*. Cambridge University Press, 1997.

[6] A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.

[7] B. Efron. Bootstrap Methods: Another Look at the Jackknife. In *Annals of Statistics*, Volume 7, 1: 1-26, 1979.

[8] T. Ge and S. Zdonik. Handling Uncertain Data in Array Database Systems. In *ICDE*, 2008.

[9] L. Golab. *Sliding Window Query Processing over Data Streams*. Ph.D. Thesis, David R. Cheriton School of Computer Science, University of Waterloo, 2006.

[10] M. Gotz, C. Koch. A Compositional Framework for Complex Queries over Uncertain Data. In *ICDT*, 2009.

[11] R. Hogg, J. McKean, and A. Craig. *Introduction to Mathematical Statistics*. Sixth Edition. Prentice Hall. 2005.

[12] R. Jampani, F. Xu, M. Wu, L. Perez, C. Jermaine, P. Haas. MCDB: A Monte Carlo Approach to Managing Uncertain Data. In *SIGMOD*'08.

[13] B. Kanagal and A. Deshpande. Online Filtering, Smoothing and Probabilistic Modeling of Streaming data. In *ICDE*, 2008.

[14] C. Koch. Approximating Predicates and Expressive Queries on Probabilistic Databases. In *PODS*, 2008.

[15] Mendenhall, W., and Sincich, T. *Statistics for Engineering and the Sciences*. Fourth Edition. Prentice-Hall, Inc. 1994.

[16] D. Moore, G. McCabe and B. Craig. *Introduction to the Practice of Statistics*, 6th Edition, W. H. Freeman, 2007.

[17] L. Perez, S. Arumugam, C. Jermaine. Evaluation of Probabilistic Threshold Queries in MCDB. *SIGMOD*, 2010.

[18] S. Singh, C. Mayfield, R. Shah, S. Prabhakar, S. Hambrusch, J. Neville, R. Cheng. Database Support for Probabilistic Attributes and Tuples. In *ICDE*, 2008.

[19] T. Tran, L. Peng, B. Li, Y. Diao, A. Liu. PODS: A New Model and Processing Algorithms for Uncertain Data Streams. In *SIGMOD*, 2010.

[20] Walley, P. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, 1991.

[21] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *CIDR*, 2005.

[22] Wilson, E. B. Probable inference, the law of succession, and statistical inference. *J. of American Statistical Assoc.*, 1927.

[23] W. Zhao, A. Dekhtyar, J. Goldsmith. Databases for interval probabilities. In *Intl. J. of Intelligent Systems*. V 19, 2004.

[24] MIT Cartel: http://cartel.csail.mit.edu/doku.php.

[25] The R Project for Statistical Computing: www.r-project.org.

[26] http://www.inrix.com/.