

Synopses for Probabilistic Data over Large Domains

Nicholas D Larusso
Dept. of Computer Science, UC Santa Barbara
nlarusso@cs.ucsb.edu

Ambuj Singh
Dept. of Computer Science, UC Santa Barbara
ambuj@cs.ucsb.edu

ABSTRACT

Many real world applications produce data with uncertainties drawn from measurements over a continuous domain space. Recent research in the area of probabilistic databases has mainly focused on managing and querying discrete data in which the domain is limited to a small number of values (i.e. on the order of 10). When the size of the domain increases, current methods fail due to their nature of explicitly storing each value/probability pair. Such methods are not capable of extending their use to continuous-valued attributes. In this paper, we provide a scalable, accurate, space efficient probabilistic data synopsis for uncertain attributes defined over a continuous domain. Our synopsis construction methods are all error-aware to ensure that our synopsis provides an accurate representation of the underlying data given a limited space budget. Additionally, we are able to provide approximate query results over the synopsis with error bounds.

We provide an extensive experimental evaluation to show that our proposed methods improve upon the current state of the art in terms of construction time and query accuracy. In particular, our synopsis can be constructed in $O(N^2)$ time (where N is the number of tuples in the database). We also demonstrate the ability of our synopsis to answer a variety of interesting queries on a real data set and show that our query error is reduced by up to an order of magnitude over the previous state-of-the-art method.

General Terms

Data synopsis, Probabilistic databases

Categories and Subject Descriptors

H.2 [Database Management]: Database applications—*Database applications*

1. INTRODUCTION

Reasoning about complex systems and events is often accomplished through probabilistic modeling in which attributes of interest are assumed to be drawn from a probability distribution conditioned on the current state of the system. Such probabilistic mod-

eling can account for many sources of data uncertainty, such as inferring unobserved values given a set of relevant variables and handling sensing or measurement errors among others. Storing measurements in their true uncertain form (i.e. as probability distributions) allows for useful data analysis in which uncertainty can be taken into account throughout the entire analysis pipeline, providing statistically reliable results. Handling uncertainty in a principled manner poses new challenges for data management which has only recently begun to be addressed by several probabilistic database management systems (PDBMSs) [4, 7, 15, 20].

PDBMSs provide a concise representation of a probability distribution over an exponential number of certain databases, or *possible worlds*. In each *world*, tuples exist with certainty, just as they would in a standard database. To answer a query in a PDBMS, it is necessary to evaluate the query in each world and aggregate across all possible worlds by summing the probability of each world in which a tuple appears in the query answer set. While this procedure provides an intuitive semantics on the query results, it has been shown, in general, that answering queries with respect to the *possible worlds* semantics is $\#P$ -hard [20].

Data uncertainty, and the associated query complexity, add challenges to data management as well as common analysis tasks such as data exploration and summarization. Data exploration is an important preliminary step in any large scale data analysis to identify any interesting patterns in the data. At this stage, it is usually preferable for the user to be able to obtain approximate query results very quickly, rather than wasting time *exploring* the data [14]. Additionally, data summarization provides a concise synopsis which can aid exploration and mining since the summary is typically orders of magnitude smaller than the original data while preserving the important attributes. Both of these tasks are even more crucial in the presence of uncertainties since there is now an extra dimension to the data.

To exacerbate this complexity, previous attempts to summarize probabilistic data have made the limiting assumption that attributes are discrete values with a small domain size, typically on the order of 10 possible values [5]. However, values based on applications in sensor network data management and managing scientific data (eg. experimental measurements in the physical sciences), which are often cited as a motivating applications for probabilistic databases (eg. [3, 1]), are typically concerned with continuous attributes and discrete values over large domains [18]. In such applications, the current PDBMS internal representation scheme is inadequate, requiring every possible value of each tuple to be explicitly listed along with its probability value.

To address these issues, we propose two polynomial-based techniques to represent arbitrary probability density functions. Using these representations, we develop methods to efficiently construct

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EDBT 2011, March 22–24, 2011, Uppsala, Sweden.

Copyright 2011 ACM 978-1-4503-0528-0/11/0003 ...\$10.00

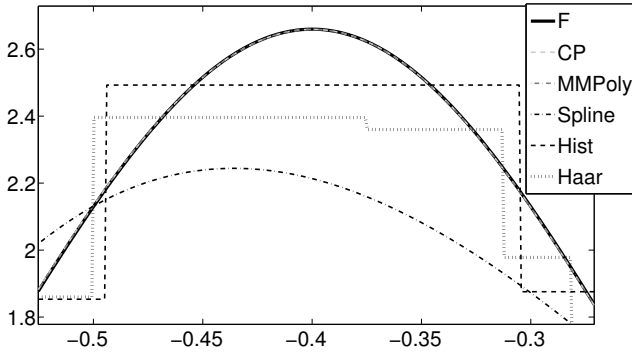


Figure 1: The peak of a Gaussian distribution along with function approximates. CP: Chebyshev polynomials, MMPoly: minimax polynomial, Spline: cubic splines, Hist: histogram optimized for L_∞ , Haar: unrestricted Haar wavelets optimized for L_∞ [16].

a space-constrained database synopsis for continuous uncertain attributes that is capable of answering approximate (error bounded) queries. We focus on continuous attributes here because they have been largely ignored in past work. We note, however, that discrete variables defined over a large ordinal domain may be appropriately represented by our techniques as well.

We introduce *minimax* and Chebyshev polynomials, in contrast to histograms or wavelets, to represent distributions over real-valued attributes. Chebyshev polynomials (CP) are a set of basis polynomials that can be used to provide a close approximation of the minimax polynomial of a function on the bounded continuous interval $[-1, 1]$. A *minimax polynomial*, P_m^* , is a polynomial that interpolates a continuous function, f , and has the minimum L_∞ error of all possible polynomials of degree m that interpolate f . In addition to being defined over a continuous interval, this representation has the benefit of minimizing the L_∞ error uniformly over the interval. We utilize the natural ability of these polynomials to minimize the L_∞ error in order to provide tight bounds on error induced from answering queries on the synopsis. Further details and background on minimax and CPs are provided in section 2.

To provide query error bounds with our probabilistic data synopsis we utilize the L_∞ error metric throughout this work. We choose L_∞ because it allows us to provide a bound over an arbitrary range of the domain. This means that, for an arbitrary range, we can compute the (approximate) probability that a tuple lies within this range as well as a bound on the error of this probability. We develop a technique for constructing a database synopsis which considers the L_∞ error. Furthermore, we show that our methods result in significantly lower error than the current state of the art (over 10x improvement).

We argue that such a parameterized representation is more appropriate for approximating smooth, continuous functions than other commonly used methods. As an example, consider approximating a normal distribution as shown in figure 1. The original function, F , is well represented by both minimax and Chebyshev polynomials (in the figure the curves completely overlap, making it difficult to clearly show their approximations) while histograms and wavelets induce large errors with the allocated space. Additionally, splines can perform poorly because the knots, the points along the x-axis in which the function values are known, are evenly spaced across the domain. Therefore, depending on where the interesting part of the function lies, the resulting spline curve can be shifted (as we observe here) or miss regions of the function completely.

Method	Probability Error (space: 31 coefficients)	Space (coefficients) (error: ~ 0.01)
MMPoly	4.0e-5	21
CP	9.2e-3	31
Spline	1.0 ¹	92
Hist	0.581	750
Haar	0.367	900

Table 1: Probability errors for each representation when space is held constant, and space requirements when error is held constant.

We provide function approximations under two different sets of constraints. First, we hold the space budget constant at 31 coefficients (248 bytes) and measure the approximation error, then we hold the approximation error constant and measure the space necessary for each method. These values are shown in table 1.

To put the error into more understandable terms, we equate the L_∞ approximation error to a *probability error* defined as maximum deviation of the approximation from the real probability over any range. For example, a probability error, $\epsilon = 0.1$, would mean that computing $\hat{P}(a \leq X \leq b)$ using the approximate distribution for any range $[a, b]$ would deviate by at most 0.1 from the real probability.

When we hold the space budget constant, only the Chebyshev and minimax polynomials provide approximations with usable error. Furthermore, minimax polynomials provide an improvement of two orders of magnitude over CPs. We observed this order of improvement frequently from our experiments with non-parametric distributions as well. Additionally, we see that histograms and wavelets require over 35x more space to reach the same level of error and, while not quite as drastic, splines require 4x more space. This simple example demonstrates the importance of providing more appropriate representation techniques for density functions when space is limited. With this motivating example in mind, we next introduce our problem statement and summarize the contributions made in this work.

Problem statement: Given a collection of N probabilistic tuples with attribute uncertainty over a continuous domain and a space budget B , our goal is to efficiently construct a probabilistic data synopsis which is capable of accurately answering (approximate) queries with an error bound.

Our contributions in this paper can be summarized as follows.

- We propose a space efficient technique for accurately representing arbitrary probability density functions based on minimax polynomial interpolation. We further show how to construct a probabilistic synopsis for a database capable of answering approximate queries and show that our methods provide more accurate results when compared to other representation techniques.
- We derive upper and lower bounds on the L_∞ approximation error of Chebyshev polynomials. We use the upper bound to (1) avoid re-clustering the tuples during our synopsis construction, (2) develop and apply an *adaptive coefficient allocation* algorithm, and (3) derive a distance computation which is independent of the size of the domain which is used for clustering similar distributions.
- We perform an extensive experimental evaluation of our methods using a real dataset consisting of uncertain tuples over a continu-

¹Values that fall below 0.0 or above 1.0 can be truncated since probability densities are strictly positive and integrate to 1.

ous domain. We show that our synopsis can be constructed more efficiently, providing several orders of magnitude improvement over competing methods. We also show the flexibility of our synopsis in answering several different types of queries in which our methods provide up to a 35x reduction in error over previous methods.

2. BACKGROUND

In this section, we first describe the key properties of Chebyshev polynomials and how they are used for function interpolation. Then we introduce the notion of *minimax polynomials*. We describe their mathematical properties and the Remez exchange algorithm (REA) which is used to approximate the minimax polynomial for an arbitrary function.

2.1 Chebyshev Polynomials

Chebyshev polynomials are a set of orthogonal polynomials in x of degree m , defined as $T_m(x) = \cos(m \cos^{-1}(x))$ for $x \in [-1, 1]$. This stems from de Moivre's Theorem which states that $\cos(m \theta)$ is a polynomial of degree m in $\cos(\theta)$ (where $x = \cos(\theta)$) [19]. The m^{th} degree Chebyshev polynomial can be defined through the following recurrence.

$$T_m(x) = 2xT_{m-1}(x) - T_{m-2}(x)$$

The first few Chebyshev polynomials are listed below (T_0 and T_1 provide the initial conditions for the recurrence):

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \end{aligned}$$

Although x is only defined on the interval $[-1, 1]$, it is easy to generalize this to the an arbitrary interval $[a, b]$ [19].

Because Chebyshev polynomials are a set of basis functions, they can be used to represent any polynomial function exactly. Additionally, they are typically capable of providing a good approximation of any continuous function defined over a closed interval. In fact, previous work has shown that they provide a good estimate of the minimax polynomial for many functions.

To transform any continuous function f into Chebyshev space, we must compute the polynomial coefficients for the function in Chebyshev space (using the Chebyshev polynomials as basis functions). Equation (1) shows the formula for computing the coefficient of the j^{th} Chebyshev polynomial for a polynomial of order m .

$$c_j = \frac{2 - \delta_{\{j=0\}}}{m+1} \sum_{k=0}^m f(x_k) T_j(x_k) \quad (1)$$

where $\delta_{j=0}$ is the Kronecker delta which is 1 only if $j = 0$ and 0 otherwise, the x_k 's are the roots of the Chebyshev polynomials, otherwise referred to as the Chebyshev nodes. For $k = 0, 1, \dots, m$,

$$x_k = \cos\left(\frac{k + \frac{1}{2}}{m+1}\pi\right) \quad (2)$$

The Chebyshev nodes offer a particularly good set of points at which to interpolate¹ a function on the interval $[-1, 1]$. These points are typically chosen as the initialization values for iterative methods which compute the best *minimax* polynomial of a function, that is, the polynomial which uniformly minimizes the L_∞

distance to the function.

2.2 Minimax Polynomials

Chebyshev polynomials typically provide a close approximation of the minimax polynomial of a function. However, for more complex functions, computing the (approximate) minimax polynomial can provide significant improvements in the L_∞ approximation error. We introduce minimax polynomials here and provide an outline of the algorithm used to approximate them.

The problem of computing the minimax polynomial of a continuous function over a closed interval is a well studied problem in the applied mathematics [10] and numerical methods communities [25]. The solution is typically computed by some form of the Remez Exchange Algorithm (REA) [21] which utilizes the alternating theorem (described below) to iteratively update the polynomial coefficients and the interpolation points. We first define minimax polynomials and introduce some key theorems from the literature, then provide a brief outline of REA for completeness.

DEFINITION 1. *Minimax polynomial: A polynomial approximation which, of all the polynomials of the same order, has the smallest L_∞ distance to the true function.*

We begin by presenting a theorem stating proof of existence of a polynomial approximation. For more information on the theorems presented in this section (or their proofs) we refer the interested reader to [19] and [22].

THEOREM 1. (*Weierstrass' Theorem*) [19] *For any function f defined over the closed interval $[a, b]$ and for any given $\epsilon > 0$, there exists a polynomial p_m for some sufficiently large m such that $\|f - p_m\|_\infty < \epsilon$*

This theorem offers a powerful theoretical result: that we may approximate any continuous function on a closed interval to an arbitrarily small error using the minimax polynomial. Unfortunately, for arbitrary functions, the minimax polynomial can only be computed analytically for $m = 1$. For higher order polynomials, iterative numerical methods, such as the REA [21], must be employed.

Next, we explain the optimality constraint for a polynomial approximation.

THEOREM 2. (*Alternating Theorem*) [19] *For any function f over the interval $[a, b]$, a minimax polynomial approximation p_m exists, and is uniquely characterized by the 'alternating property' that there are (at least) $m + 2$ points in $[a, b]$ at which $f - p_m$ attains its maximum absolute value with alternating signs.*

This theorem provides the mechanism by which it is possible to check for the optimality of any given approximation. Given an approximating polynomial, we can compute the error function over the closed interval, if the peaks and troughs of the error function are all approximately of the same magnitude, we've reached the optimal polynomial (within some precision). Using this theorem, we also have an idea of how close to optimal our approximation is at any intermediate step of REA.

Remez Exchange Algorithm Outline: REA is an iterative algorithm which computes an approximate minimax polynomial for an arbitrary function. The Remez algorithm works by alternating between updating estimates of polynomial coefficients and refining the set of interpolation points. The algorithm is briefly outlined below.

¹We use the terms function *interpolation* and *approximation* interchangeably.

We first initialize the set of interpolation points, x_i , usually by setting them to the Chebyshev nodes for the m^{th} degree Chebyshev polynomial, which we know from theorem 2 has m roots.

The first step of the iterative REA is to set up and solve a set of $m + 2$ (independent) linear equations which gives us the $m + 1$ polynomial coefficients and the resulting approximation error. For our input function F , interpolation points x_i 's, and error $E = \max_x |F(x) - P_m(x)|$, we solve the following set of linear equations:

$$F(x_i) - P_m(x_i) = (-1)^i E \quad \forall x_i$$

With the updated polynomial coefficients, c_j 's and error, E , the second step of the algorithm seeks to update the $(m + 2)$ interpolation points to approach the minimax condition (equal error magnitude). This is accomplished by performing a local hill climbing on the residual function, which is defined as the error between the approximation and input functions, for each point so that each x_i is shifted toward the closest local maxima while maintaining alternating signs.

This process is iterated until the error maxima converge to the same magnitude. In our experiments we found that REA typically converged in under ten iterations and reduced the L_∞ error by at least half and as much as six orders of magnitude over a direct application of Chebyshev Polynomials.

3. PROBABILISTIC DATA SYNOPSIS

In this section we introduce our uncertainty model and provide an overview of the process used for building our probabilistic data synopsis. We explain what data composes the synopsis and how it is used to answer queries and provide bounds on the error.

3.1 Uncertain Data Model

We assume we are given a set of independent tuples with a single uncertain attribute over a continuous domain. This fits within the *attribute uncertainty* model presented in prior work because we assume that each tuple exists in the dataset and only its value is uncertain. Extending this to the *tuple uncertainty* model is possible by simply allowing the each probability distribution to sum to a value *at most* one, where the total area under the curve denotes the probability of existence. For presentation simplicity, we do not explore this model further.

We focus on continuous probability distributions, however, the methods developed here are suitable for discrete distributions on an ordered domain as well. This may improve space efficiency (over histograms or wavelets) for complex distributions defined over a large domain. Furthermore, interpolating discrete functions over a closed interval using CPs has been addressed in [2] and has shown to produce impressive results.

3.2 Synopsis Overview

For each tuple in the database, we first compute the minimax polynomial approximation of the density function by applying the Remez algorithm. Empirically, we have found that a 40^{th} order polynomial provides a good representation for the majority of the density functions we have seen in practice. Additionally, we have experienced numerical instability when computing polynomials of larger order making them difficult to evaluate. Once the polynomial coefficients have been computed, we transform each tuple into Chebyshev space. Note that there is no information loss in this step since both representations are polynomials. Alternatively, we can trade-off accuracy for a faster construction time by applying Chebyshev polynomial interpolation directly. This will reduce construction time significantly as the Remez algorithm takes, on av-

erage, an order of magnitude longer than Chebyshev interpolation. In our experiments, we provide a comparison of both methods.

Data reduction is performed by first aggregating tuples with similar density functions and then applying an adaptive coefficient allocation algorithm which efficiently allocates space to represent tuples by considering a global error measure. We use hierarchical clustering [8] for data reduction, and introduce an algorithm that improves the synopsis error by alternating between searching for an appropriate number of clusters and allocating coefficients to these clusters given the space budget [8]. Each of these methods is described in detail in the following sections.

Finally, the synopsis stores the CP approximation for each cluster representative along with the cluster size and the maximum within cluster L_∞ error. This information is used to answer queries and provide error bounds. We illustrate this procedure by considering a range query, though the process is similar for other query types. First, we need to compute the probability that each tuple belongs to the result set. We do this by integrating the CP over the specific range. This probability is used to represent all tuples assigned to this cluster. Furthermore, to compute the error bounds, we take the maximum within cluster L_∞ error and add this value to the 0^{th} order coefficient and integrate this function over the query range. This is the probability value upper bound for this cluster (considering our error). Similarly, we can compute the lower bound and return these values to the user.

Although this error bound is correct, it will tend to provide an overestimate of the potential error in our results. We improve upon this method by considering that Chebyshev polynomials interpolate their target functions, that is, they pass through the function at a known set of points. Additionally, we have from the alternating theorem, that the sign of the error function will alternate and reach maximum magnitude $m + 2$ times, therefore having m roots at the Chebyshev nodes. Using this, we need only to compute the error over alternating intervals, since the error will be additive over these regions (i.e. the error will have the same sign). We show the effectiveness of this approach over various query range widths in our experiments (see figure 7(a)).

In the case of minimax polynomials, this does not provide a strict upper bound on the error since new interpolation points are computed. To fix this, we may simply compute the deviation of each new interpolation point from the original Chebyshev nodes and store the maximum of these values, δ , in the synopsis. Then, when we compute the error, we can assume the current interval is expanded by δ in both directions and compute the error in the same manner as described above. This will provide a strict bound since we are assuming the worst case for each interval. Additionally, it only adds a single extra value to the synopsis.

4. SYNOPSIS CONSTRUCTION

Next we explain each step of the synopsis construction process in detail. We start by deriving bounds on our approximation error that results from pruning coefficients from the representation. Then we introduce our adaptive coefficient allocation algorithm and examine the relationship between our error bound and the quality of the resulting coefficient allocation. Next, we explain how to cluster distributions more efficiently by again utilizing our approximation bound and deriving a bound on the distance between two tuples. Finally, we provide an algorithm that combines these methods to construct a synopsis that effectively helps reduce the global error of the synopsis given the user defined space budget.

4.1 Approximation Error Bounds

Constructing a data synopsis is often computationally expensive

and this can be exacerbated by using approximation methods that must be recomputed each time their space budget is altered (i.e. histograms). We avoid this problem by developing an ‘incremental’ construction process for CPs. We avoid this problem by developing an ‘incremental’ construction process for CPs. For each tuple, we compute a CP for this distribution just once. Then as we are figuring out how many tuple representatives are needed and how much space should be allocated to each, we can remove (and replace) coefficients quickly and evaluate the synopsis error for each configuration.

We derive bounds on the approximation error induced by pruning Chebyshev coefficients from the complete polynomial representation. Our error bound offers several benefits over the actual error and is used throughout our synopsis construction process. In addition to providing computational savings in the synopsis construction process, our upper bound provides an error function that is convex in the number of pruned coefficients. Utilizing the fact that the (bounded) error for each representation is strictly decreasing with each new coefficient, we develop an algorithm that allocates space to a set of tuples quickly and effectively while reducing a global error. We discuss our adaptive coefficient allocation algorithm and its connection to the quality of this bound in the following section.

Our approximation bound is based on the magnitudes of the coefficients for the Chebyshev polynomials. This ensures an efficient computation which requires only the Chebyshev approximation, whereas computing the real error would require the true distribution as well.

THEOREM 3. *The L_∞ error E , induced from pruning coefficients of a Chebyshev polynomial is bounded by the following:*

$$\left| \sum_{g \in G} c_g \right| \leq E \leq \sum_{g \in G} |c_g| \quad (3)$$

where G denotes the set of indices of the pruned basis functions and c_g is the coefficient value of the g^{th} polynomial.¹

This theorem provides us with an efficient procedure with which to compute the upper bound on the approximation error after pruning a subset of coefficients. This also presents us with a pruning order which minimizes L_∞ error. From the bounds, we see that by pruning coefficients in order of increasing magnitude (prune the coefficient with the lowest absolute value first) we add the smallest potential for error at each step. This result is important because it allows us to reduce our space consumption by simply pruning coefficient instead of recomputing the approximation which can be costly for minimax polynomials.

Despite the simplicity, our upper bound tends to be tight in practice. To see this, remember that Chebyshev polynomials have a cosine basis, $T_m(x) = \cos(mx)$, and the number of interpolation points increases linearly with the polynomial order. Therefore, we see that coefficients naturally tend toward 0 as the polynomial order increases: $\lim_{m \rightarrow \infty} \int_{-1}^1 \cos(mx) dx = 0$

Using this upper bound, we can pre-compute a polynomial representation of each tuple, pruning coefficients from less complex distributions adaptively to adjust the synopsis to our space requirements. Our coefficient allocation algorithm is described next.

4.2 Adaptive Coefficient Allocation

Due to the large number of functions we aim to approximate, tuples are likely to exhibit varying levels of complexity. This provides an opportunity to further reduce our space consumption (or reduce error) by intelligently allocating coefficients to each tuple (or representative tuple) to reduce the global error of our synopsis.

¹All proofs are available in the appendix.

For example, consider the three density functions shown in figure 2. Function C is more complex than A and as such, coefficients should be allocated appropriately to provide the best approximations for the entire collection of functions. Table 2 shows a comparison of a uniform coefficient allocation vs. our adaptive approach for a budget of 24 coefficients along with the associated errors. Although the uniform allocation provides very low approximation errors for functions A and B , C contains a very large L_∞ error. In contrast, the adaptive allocation algorithm assigns the majority of the allotted coefficients to function C , thereby significantly reducing the *global* L_∞ error of these functions.

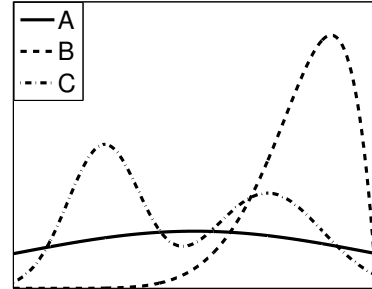


Figure 2: Example density functions of various complexity.

To address the issue of managing functions of various levels of complexity, we propose an algorithm to adaptively allocate coefficients to those tuples that would most benefit from the additional representational power. The idea is to incrementally assign the next coefficient to the tuple that is currently represented the least accurately (i.e. has the largest L_∞ error). We assume that we have the CP for each tuple precomputed and we find the tuple with the maximum error bound. We assign an extra coefficient to this tuple and repeat until there are no more coefficients to assign. The algorithm has time complexity $O(B)$ and space complexity $O(N)$, where B is the space budget and N is the size of the database. Furthermore, the resulting allocation can be shown to be optimal under certain assumptions which we discuss next.

Function	Uniform		Adaptive	
	Alloc	Error	Alloc	Error
A	8	2.73e-12	3	6.14e-04
B	8	4.44e-15	7	2.20e-03
C	8	1.10e-01	14	4.62e-03

Table 2: Probability errors for each representation method.

From our upper bound, we see that pruning coefficients in order of increasing magnitude minimizes our L_∞ error. This guarantees that our (upper bound) error function is convex. Using this, we can show that if the true error is strictly decreasing, then our algorithm provides an optimal allocation. While, in general, this may be difficult to guarantee, an upper bound on the error with this property may be used instead (as we do here). In this case, the quality of our algorithm is directly dependent on the relative tightness of our error bound. Our experiments show this algorithm performs very well in practice, typically providing near optimal coefficient allocations and a significant improvement over a uniform allocation (see section 5.2.3).

We denote $F = \{f_1, \dots, f_K\}$ as a set of K functions to be approximated given a space budget, B . We denote the local budget for each function f_i as x_i . An optimal allocation for space B is

$X_B^* = \{x_1, \dots, x_K\}$ such that $\sum_i^K x_i = B$ where each x_i is a positive integer denoting the number of coefficients allocated to the approximation of function f_i . An allocation is optimal in the sense that the global error, the error defined over the set of all function approximations, is minimized over all other allocations with the same space budget: $Err(X_B^*) \leq Err(X_B)$. We note that there may be more than one optimal allocation for a given budget.

We make a distinction between local and global error. The local error function is defined between a function, f_j , and its b -space constrained approximation, p_m^b . As discussed earlier, we use the L_∞ error: $err(f_j, b) = \max_i |f_j(i) - p_m^b(i)|$. The global error is defined over the entire set of functions, F , here we use *maximum* since this value affects our error bounds: $Err(X_b) = Err(F, X_b) = \max_j err(f_j, x_j)$.

Intuitively, we can see that at each step we should be assigning more coefficients to the worst approximation, otherwise our error is not being reduced. Also, due to the fact that each error is strictly decreasing we can see that if, at one step we were to reassign a coefficient from f_i to f_j , then the error for f_i would increase. Since we assigned a coefficient to f_i at some point, obviously this helped us reduce the global error and thus this reassignment actually increases our error.

THEOREM 4. *If $err(f, b)$ is strictly decreasing (with respect to b), then an optimal allocation X_{b+1}^* can be computed greedily (by adding 1 to exactly one element of X_b^*). That is, $\exists! i, x_{i,b+1} = x_{i,b} + 1$ and $\forall j \neq i x_{j,b+1} = x_{j,b}$.*

This theorem shows that, if the true approximation error is strictly decreasing, a greedy algorithm provides an optimal allocation. Since we use an upper bound instead of the true error, the quality of the allocation produced by our algorithm is dependent on the tightness of our upper bound.

4.3 Clustering

Combining similar tuples to reduce the database size is a key step in constructing any data synopsis. Previous work in probabilistic data synopses has addressed this problem by assuming an ordering on the tuples in the database in order to utilize a dynamic programming approach for combining similar probability distributions [5]. Here, we remove this limiting assumption and propose a clustering based approach.

One difficulty with this approach is that computing the distance between two density functions requires an expensive integral. To circumvent this complexity, we take advantage of our representation and show that we can bound the distance between two distributions by only evaluating the distance between their associated CP coefficient vectors. Again, we utilize our approximation error bound and further derive a bound on the distance between two tuples by comparing only their Chebyshev coefficients.

We build on the result of a lemma introduced in [2] (labeled lemma 3) which states that a linear function combination corresponds to a linear combination of CP coefficients.

LEMMA 1. *let c_1, c_2 , and c_z be the Chebyshev coefficients for the functions f_1, f_2 and their difference $f_z = f_1 - f_2$, then $0 \leq i \leq m, c_z(i) = c_1(i) - c_2(i)$, where m is the total number of coefficients.*

Using this result along with our upper bound, we can show that comparing two vectors of CP coefficients using the L_1 distance metric provides an upper bound on the L_∞ distance between the two functions.

THEOREM 5. *Minimizing L_1 distance between vectors of CP coefficients bounds the L_∞ distance in function space.*

The above theorem enables us to perform clustering in the reduced space of the CP coefficient space. In addition to the computational savings, we show experimentally that this distance bound provides quality clustering results.

4.4 Iterative Synopsis Refinement

Next we show how these methods are combined in our synopsis construction algorithm. Our goal is to allocate the budgeted space throughout the synopsis in such a way as to reduce the synopsis error. We start by applying hierarchical clustering to the database (using our distribution distance bound). This enables us to explore different clustering configurations while computing the distances and linkages (distances between sets of observations) just once which is important as these are the most computationally intensive operations in hierarchical clustering.

Algorithm 1 Space budgeted synopsis construction algorithm

Input: Probabilistic database: *probDB*, space budget: *B*

Output: Probabilistic Data Synopsis: *probSyn*

```

1: Perform hierarchical clustering, initialize  $\alpha$  (gradient step size)
   and  $k$  (initial no. of clusters),  $E_0 = \infty, \minErr = \infty, i = 0,$ 
    $\beta = 0.75$ 
2: while  $|\alpha| > 1$  do
3:   Compute cluster representatives,  $\text{clusterRep}[1..k]$ 
4:    $\text{probSyn} = \text{AdaptCoeffAlloc}(\text{clusterRep}[1..k], B)$ 
5:    $E_i = \text{error}(\text{probSyn})$  // Compute global error of probSyn
6:   if  $E_i > E_{i-1}$  then
7:      $\alpha = -\beta\alpha$  // change directions
8:   end if
9:   if  $E_i < \minErr$  then
10:     $\minErr = E_i$ 
11:     $\text{optProbSyn} = \text{probSyn}$  // save clustering
12:   end if
13:    $k = k + \alpha$ 
14:    $i++$ 
15: end while
16: return  $\text{optProbSyn}$ 

```

The intuition behind this method is to alternate between modifying the number of clusters and the allocation of coefficients to the representative tuples such that we continue improving the synopsis error. Naturally, there is a trade-off between the two, by storing more representatives, we are able to allocate fewer coefficients to each which may raise the synopsis error. Therefore, we perform a search over the number of clusters in which we keep increasing (or decreasing) k as long as the synopsis error continues to decrease (increase). Because we only take a step if it reduces the global synopsis error, we converge to a local minimum. An outline of this method in algorithm 1.

5. EXPERIMENTS

We examine the following aspects of our probabilistic data synopsis: scalability, approximation quality, query approximation and query error bounds. Specifically, we discuss the rate at which our synopsis construction algorithm scales as the database size grows, exploring the individual components involved (i.e. Chebyshev interpolation, clustering, etc.). Next, we thoroughly examine each of our proposed approximations and space saving techniques described in section 4. Then we execute several queries and again show that our synopsis provides results with significantly lower error than competing methods. Lastly, we examine the query error

bounds resulting from our synopsis and examine the difference between using CPs and minimax polynomials.

We evaluate our proposed methods on a dataset consisting of real-valued measurements of cloud cover taken around the globe from 1981 to 1991¹. The raw reports contain a value between 0 and 10 which provides a measure of cloud cover at a specific location along with the time and date the measurement was observed. The measurements were taken approximately every 3 hours on selected days over this period. To transform these measurements into an uncertain dataset, we chose to aggregate the set of observations over each day. That is, we assume the measurements from each date were generated i.i.d. from a continuous density function. To estimate this density function we apply a Gaussian kernel² to the set of observations. The resulting uncertain dataset contains 100k tuples, each with an arbitrary probability density over the bounded domain $[0, 10]$, which describes the density of cloud cover observations for a particular location on a specific date.

All experiments were run on an Intel i7 2.67 GHz machine with 12.0 GB of main memory. Our code was written primarily in Java (1.6), but some functions were written in C and Matlab (R2009). Most notably, the Remez algorithm was written in Matlab for implementation simplicity. This will certainly provide a distorted comparison of running times, however, we do not investigate this in depth (see figure 3(c)).

5.1 Scalability

Our synopsis construction is composed of three major steps: (1) polynomial interpolation, (2) hierarchical clustering, and (3) the *K-Search*, which improves the error by searching for an appropriate number of clusters. We first analyze the time breakdown of the construction in terms of each of these steps.

The most expensive step of the construction process is clustering, which has time and space complexity $O(N^2)$. The cost of computing the minimax polynomial of each density and the adaptive coefficient allocation algorithm both scale linearly with the number of tuples.

Comparing CP with minimax polynomials (figures 3(b) and 3(a)), we see that the Remez algorithm takes a significant portion of the total time, and therefore one must evaluate the trade-off between the increased accuracy of minimax polynomials and the time to compute them. However, the overall time complexity for our synopsis construction for both Chebyshev and minimax polynomials is $O(\min(N^2, T) + N + K)$, where the $\min(N^2, T)$ term is for clustering where T represents the maximum number of tuples that may be clustered in memory, N is for polynomial approximation, and K is for our *K-search* algorithm.

Next, we compare the construction time against probabilistic histograms, the current state-of-the-art method in probabilistic data summarization. However, we were not able to build a probabilistic histogram directly given our data since our domain is continuous. To ameliorate this, we sampled 100 equally spaced points to use as our domain and interpolated the density at those points. This still provides us with a domain and space budget that are orders of magnitude larger than the experiments carried out in [5], both of which affect the construction time.

We were only able to construct a probabilistic histogram with 1,000 tuples, for 2,500 tuples we allowed the program to run for over 24hrs before stopping the (incomplete) process³. To construct a probabilistic histogram more quickly, we divided the complete database (10k tuples) into blocks of 500 tuples each and assigned each block an equal amount of space. A probabilistic histogram was then built on each block and the total time was computed by taking the summation of the construction time over all blocks. A

comparison of the construction times is presented in figure 3(c).

Even computing the minimax polynomials, we are able to construct a synopsis in less time than using the blocked probabilistic histograms. We only experiment with up to 10k tuples in our experiments in order to provide a fair comparison to probabilistic histograms. However, as we show in figures 3(b) and 3(a), our synopsis scales much better in terms of both the number of tuples and length of the tuple domain.

5.2 Approximation Quality

In this section we will investigate several aspects of our synopsis related to the quality of the approximation. We start by examining the quality of the bounds we derived for pruning CP coefficients and show that they tend to be useful in practice. Then, we show that using the CP bounds to compute distances for clustering provides quality clusters. Lastly, we experiment with our adaptive coefficient allocation algorithm and show that it provides significant (near-optimal) improvements over a uniform allocation.

5.2.1 Approximation Error Bounds

Many of the methods we describe for constructing our probabilistic database synopsis are dependent on the approximation error bounds we derived for pruning Chebyshev coefficients. We evaluate the tightness of these bounds on our dataset by sampling 10k tuples and evaluating the difference between the real probability error and the error bounds after pruning a variety of coefficients (out of a maximum of 41). Figure 4 shows the average difference between the real probability error and the lower and upper bound (the deviation of each bound from the true error) along with error bars denoting one standard deviation.

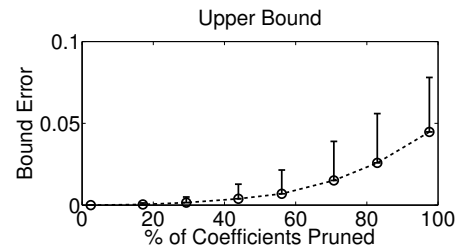


Figure 4: Approximation error bounds.

The upper bound tends to be very tight when pruning up to 50% of the coefficients and loosens up slightly afterwards (this is explained in section 4.1). Additionally, we note that in the context with which we use the bound, it is not the absolute error that matters, but the rank ordering of the bound relative to the true error. This is explored further in the next section.

5.2.2 Clustering Quality

To validate our approach to clustering tuples based on their Chebyshev coefficients, we compute the real and approximate distances and compare the results. Due to the nature of clustering, we are most interested in the relative distance between distributions using our bound in place of the real distances. That is, if $\text{dist}(A, B) > \text{dist}(A, C)$, we would like $\widehat{\text{dist}}(A, B) > \widehat{\text{dist}}(A, C)$ as well, where $\widehat{\text{dist}}(\cdot, \cdot)$ represents our distance bound. We compute the Spearman's rank correlation coefficient between the two sets of distances, which measures how well the relationship between the values can be described as a monotonic function. Sampling 100 tuples (and averaging over 100 trials), we found this correlation to be 0.9950 (a value of 1 would mean the order of the values aligned perfectly). This means that using our bound for hierarchical clus-

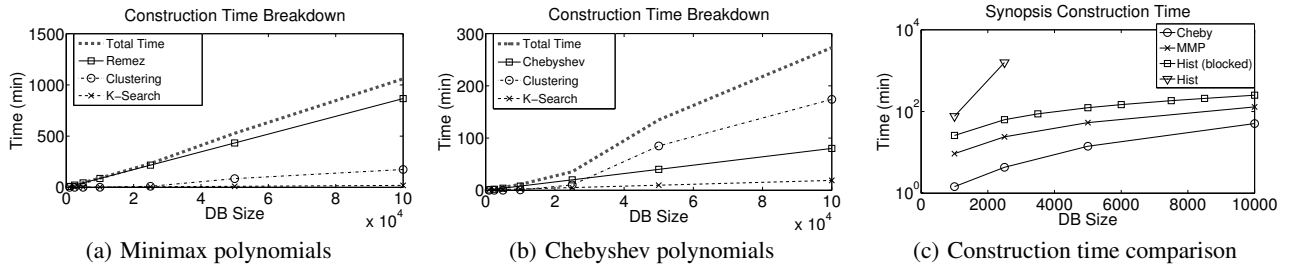


Figure 3: Synopsis construction time

tering will result in nearly exactly the same clusters as using the true distances.

5.2.3 Adaptive Coefficient Allocation

We test the effectiveness of our adaptive coefficient allocation algorithm by comparing it to both a uniform and optimal allocation. Our optimal allocation is computed using a dynamic programming algorithm, similar to that proposed in [13]. We sampled 100 tuples from our dataset and computed the error for each allocation over a variety of space budgets. This was repeated 100 times and the average errors are shown in figure 5.

Note that we are representing each tuple individually here, not clusters as we do later. This experiment validates the use of this algorithm for allocating coefficients across several tuples. We reason that if we choose good representative tuples, then this method will reduce the error of our complete synopsis.

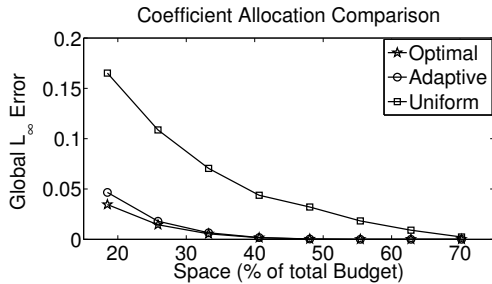


Figure 5: Adaptive coefficient allocation algorithm evaluation.

From this graph, we notice two interesting trends: (1) our adaptive algorithm significantly outperforms the uniform allocation, especially when space is limited and (2) our adaptive algorithm consistently performs at near optimal levels. The near optimal performance further corroborates the utility our error bounds, as the quality of this algorithm depends directly on the relative differences between the approximation upper bound for each tuple.

5.3 Query Approximation

In this section we test the quality of approximate query results of our probabilistic database synopsis as compared to the same queries on the original database. We pose four different queries to showcase the versatility of our synopsis and the quality of our results as compared with probabilistic histograms [5]. The probabilistic histogram synopsis was constructed as described in section 5.1 and we use 10,000 tuples from our database with a space budget of 20,000 coefficients (80kB). Note that, using a 40th order polynomial for each tuple, this budget is 5% of the space required to completely represent the complete database without clustering.

Probabilistic Range Query (RNG): The probabilistic range query is defined as: $P(a < x < b) = \int_a^b p(x)dx$. The result of this

query is, for each tuple, the probability of that tuple containing a value in the range $[a, b]$. To evaluate the quality of this query we run the same query on the original dataset and the synopsis and compare the probability returned for each tuple in the original dataset with the probability returned by the synopsis representative corresponding to that tuple. An example use of this query would be “What is the probability that the cloud coverage was in the range $[0.75, 2.25]$ at location A”. This query would tell us the probability of (nearly) clear skies at the given location, however, it could be posed using any range in the bounded domain, thus demonstrating the utility of the L_∞ error for providing bounds.

Probabilistic Threshold Query (THRESH): It is often the case that users are interested in a tuple satisfying a set of constraints only if it is very probable, thus a probabilistic threshold query filters out any ‘unlikely’ tuples from the result set. For example, we may pose the same query presented above, but threshold at a probability of 0.8 to ensure we only see the tuples that are very likely to have a value in the range. The probabilistic threshold query is defined as: $P(a < x < b) > T$. The result of this query is a count of the tuples that passed the threshold. To evaluate this query, for a given range, we compare the counts of the query on the synopsis and the original database. For our experiments, our threshold is kept proportional to the length of the range we are evaluating.

Probabilistic Join Threshold Query (JOIN): It is often of interest to understand the relationship between tuples. For example, we may be interested in finding pairs of locations that had similar cloud cover at a specific time. Since we are dealing with probabilistic data, we may not be able to find tuples that match exactly, but again, we can threshold on the probability to only present matches that occur with a high probability. The probabilistic join threshold query is defined as: $P([a < x_i < b] \wedge [a < x_j < b]) > T$ for two tuples x_i and x_j , a range $[a, b]$ and a threshold T . Semantically, this query evaluates the probability of a pair of tuples containing values in a given range and selects the pairs that have a high probability of jointly existing in that range. Our evaluation of this query is the same as the probabilistic threshold query, however, due to the potentially large result set, we sample two sets of 100 tuples and perform the join between this subset. That is, database size referred to in figure 6(c) is $100^2 = 10,000$.

Probabilistic Comparison Query (COMP): Comparing random variables from different distributions is a common technique in statistical analysis. For example, it may be of interest to find if the difference in cloud cover in two specific locations is significant. To do this, we would pose the query “what is the probability that the cloud cover at location A is greater than that of location B”. The probabilistic comparison query is defined as $P(x > q) = \int p_x(x)P'_q(x)dx$ for some query random variable, $q \sim p_q$, where P'_q denotes the cumulative density function (cdf) of p_q . This is the only query that is not dependent on a range, since the integral is performed over the entire domain. The result of this query is, for

each tuple, the probability of that tuple being greater than the query distribution. We evaluate this query similarly to the range query.

Except for *COMP*, we repeat each query using a range that varies in length from 5 - 75% of the width of the domain. For each range length, we perform 10 query trials such that the range location is picked randomly. We average our results for each range length over all the trials and plot the errors in figure 6.

Both CPs and minimax polynomials provide very low query errors across the board, with no distinguishable difference between the two methods. This is intuitive as most of the error is introduced during clustering. After this step, the more advanced interpolation technique does little to regain and lost representation accuracy (the real benefit of minimax polynomials comes from the tighter query bounds, discussed next).

More interesting is the vast difference in quality between our synopsis and probabilistic histograms. While we are able to answer each query with near perfect accuracy the probabilistic histogram introduces significant errors across the board. Our synopsis provides an error reduction of at least one order of magnitude for each of the queries. We ran the same queries varying the synopsis space budget between 10,000 and 50,000 total coefficients, each resulted in similar improvements over probabilistic histograms.

5.4 Query Error Bounds

In this section, we examine the quality of the query error bounds produced by our synopsis. We first examine the difference between the naive error bound and our improved computation which utilizes the locations of the Chebyshev nodes. Then, we examine the difference in the bound quality between CPs and minimax polynomials.

We rerun the probabilistic range query and compute the error bound on each tuple probability using both the naive and improved error bounds and plot the resulting errors and bounds in figure 7(a). As expected, we see a substantial improvement in the error bound that increases with the width of the query range. This is intuitive since the longer query ranges will cross more Chebyshev nodes, thus providing a greater error reduction.

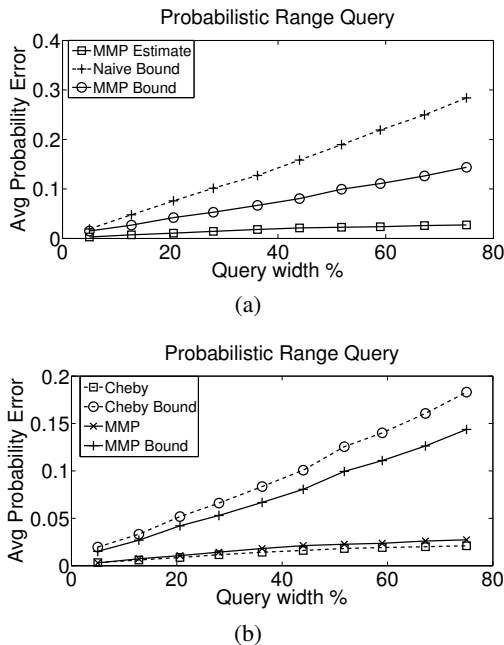


Figure 7: Query error bounds.

In figure 7(b) we compare the error bounds produced from Cheby-

shev vs minimax polynomials. Because minimax polynomials minimize the L_∞ error, we expect that their main advantage will come in terms of the resulting error bounds. We see that both methods produce similar query estimates, but minimax polynomials consistently provide a 20-30% improvement in their error bound across the width of the query range. Thus given sufficient space and more stringent error requirements, minimax polynomials may become worthwhile. For example, if we construct a synopsis on double the space budget, minimax polynomials consistently provide bound improvements of more than 50% over CPs.

6. RELATED WORK

We highlight areas of work most relevant to our own and provide the reader with a concise comparison. Specifically, we review work on summarization techniques, and managing and querying uncertain data.

Continuous vs Discrete Uncertain Attributes: Singh et al. [24] introduce the first system to integrate support for continuous distributions by integrating some common parametric distributions and discretization techniques. This is one of the first papers in the area to highlight the importance of supporting continuous distributions in probabilistic database systems. Previous work in managing uncertain data has primarily focused on managing discrete probability distributions over small domains [1, 20, 23]. In these systems, tuples have only a handful of possible values, and thus may be stored efficiently by simply enumerating the values and their associated probabilities. However, there are many applications in which there exists a large number of possible values for each tuple, or even continuous domains, which require more sophisticated techniques for storing the distributions. Additionally, this system incorporates a set of parametric probability distributions and thus lacks a unified representation technique, as well as support for arbitrary *density* functions.

Korn et al. [17] utilize cubic splines to represent a probability density of tuples over a continuous domain for the task of selectivity estimation. The authors show that splines provide significant improvements in accuracy over histograms for the same space budget as we corroborate in our own experiments when using similar error metrics. While this work provides a representation for arbitrary density functions, our work goes one step further to provide improved accuracy in limited space.

Data Summarization: Summarizing data is a general area of research concerned with accurately approximating a signal in less space than would be necessary to maintain the raw data. Haar wavelets are a popular tool in the database community for creating compact data synopses mostly due to the simplicity in which Haar decompositions may be computed (and their applicability to streaming data). Another nice property of Haar wavelets is that pruning the coefficient with the minimum magnitude is guaranteed to provide an approximation with minimum L_2 error with respect to the number of coefficients [11]. This is useful when it is the signal in its entirety that is of interest, however, when querying over local regions of a signal, the L_∞ is a more desirable metric because it provides local guarantees on the error.

¹The clouds dataset may be downloaded from <http://cdiac.esd.ornl.gov/cdiac/ndps/ndp026b.html>.

²We used the `ksdensity` command implemented in Matlab.

³Note that the experiments carried out in [5], used a database with 10k tuples as well, however, their domain was set at 10 values. Because the algorithm for constructing probabilistic histograms is dependent on the domain size, an order of magnitude increase in domain size results in an order of magnitude increase in running time as well.

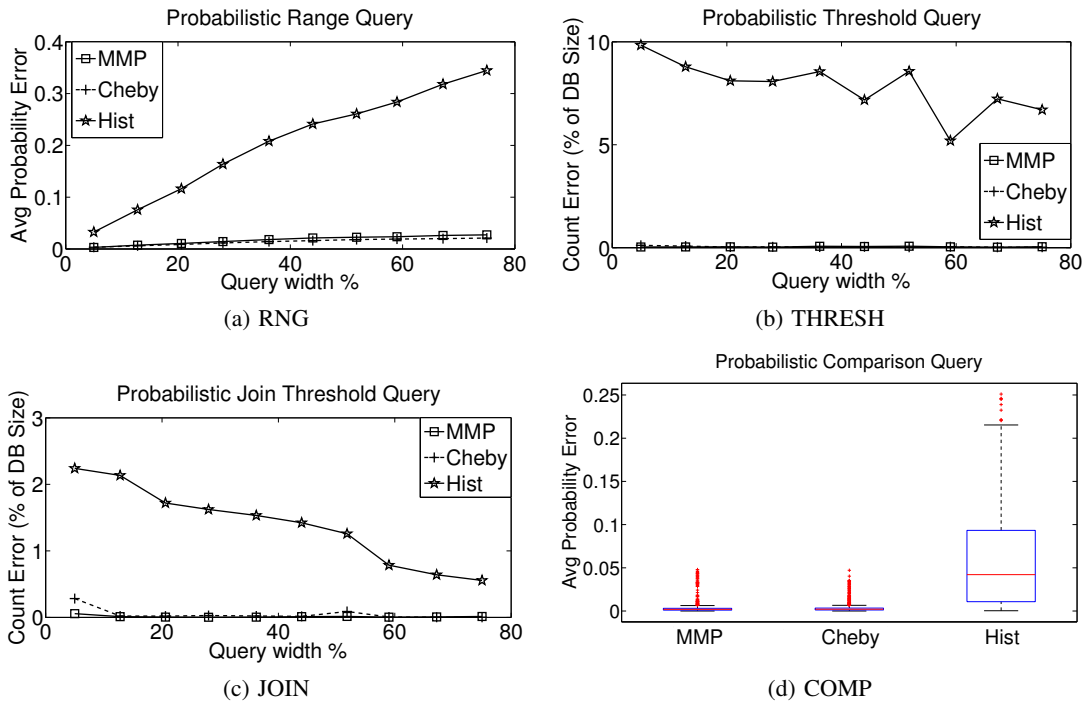


Figure 6: Query error.

In recent years, there has been a large amount of work on building space constrained wavelet decompositions that optimize the L_∞ error for its role in providing error bounds [11, 12, 16]. Garafalakis and Kumar [11] study the restricted version of this problem, in which given a space budget, and the wavelet decomposition of an input signal, the problem is only to pick the set of wavelet coefficients to keep in order to minimize L_∞ error. Later, Karras et al. [16] and Guha and Harb [12] studied the unrestricted version of this problem, in which the goal is to compute both which coefficients to maintain as well as their values to minimize the maximum error for a given space budget.

Uncertain Data Summarization: The problem of summarizing static probabilistic data was only recently addressed by Cormode and Garafalakis [6]. In this work, the authors introduced methods to build optimal histograms and Haar wavelets for probabilistic data by optimizing the expected value of an error metric. Although the expected value is often an important statistic, it is insufficient for answering approximate queries.

To address this, Cormode et al. [5] provide a technique for building a *complete* synopsis over a probabilistic database that supports approximate query evaluation based on histograms. Their approach builds a histogram over the entire database where the representative value in each bucket is a distribution represented as a histogram. This ensures that, provided sufficient space, it is possible to represent the database exactly. However, the authors assume that there is a natural ordering over the set of density functions in the database, such that tuples close to each other will have similar distribution functions. This is often not the case in reality, and we propose a solution for aggregating tuples based on similar probability densities in an efficient manner.

Chebyshev Polynomials: Previously, Cai and Ng [2] have applied Chebyshev polynomials to summarize time-series data. Whereas their work utilizes Chebyshev polynomials as a mechanism to approximate time series for indexing, we are concerned with building

an accurate representation of density functions with the added constraint of a space limitation. Therefore we utilize the salient features of Chebyshev polynomials such as their good approximation of the minimax polynomial of a function. Additionally, we further utilize the Remez algorithm [21] to compute a better approximation of the minimax polynomial and show that this step is able to provide significant gains in accuracy for smooth functions.

A plethora of other signal approximation methods have been used in the time-series literature, however, these works are typically interested in indexing and searching over complete (often discrete) signals. Therefore most of the techniques have been built or tailored for Euclidean error, thus we do not explore or compare with these methods. We refer the interested reader to [9] for a comparison of various methods.

Space Constraints: While the methods mentioned so far are concerned with representing an individual input signal, Jagadish et al. [13] introduce two algorithms to allocate buckets to a set of m histograms to minimize the global error. Improvements come from summarizing a diverse set of distributions where each requires a different number of buckets to reach the same error. We extend this idea to our task of approximating a database of probability distributions using minimax or Chebyshev polynomials. We introduce a greedy allocation algorithm and show how to avoid recomputing the summary of a distribution every time we update the space budget. Due to this, we are able to provide much better scalability than the previous work.

7. CONCLUSION

The inability of current probabilistic databases to efficiently handle data that is defined over large domain spaces limits the applicability of these systems to real world data, specifically to scientific applications which often deal with continuous attributes. In this paper, we have introduced an efficient method to construct a space constrained synopsis for probabilistic data which is capable

of answering queries with error bounds. We derived a useful upper bound on the probability distribution approximation error that allows us to prune coefficients to reduce the space of a summary while still reducing the L_∞ error. Using this bound, we also derive a distance metric based on the Chebyshev coefficients which allows for efficient clustering of distributions. Additionally, our upper bound enables us to efficiently allocate coefficients among the set of tuples to reduce the global error of our synopsis.

We believe this work provides an important step forward in terms of achieving fast and accurate *exploratory* querying over probabilistic tuples as well as extending the utility of these systems to more scientific domains.

8. REFERENCES

- [1] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, 2006.
- [2] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD*, 2004.
- [3] R. Cheng, S. Singh, S. Prabhakar, R. Shah, J. Vitter, and Y. Xia. Efficient join processing over uncertain data. In *CIKM*, 2006.
- [4] N. D. Christopher Re and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.
- [5] G. Cormode, A. Deligiannakis, M. Garofalakis, and A. McGregor. Probabilistic Histograms for Probabilistic Data. In *VLDB*, 2009.
- [6] G. Cormode and M. Garofalakis. Histograms and wavelets on probabilistic data. In *ICDE*, 2009.
- [7] N. Dalvi and D. Suciu. Management of probabilistic data foundations and challenges. In *SIGMOD*, 2007.
- [8] W. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1), 1984.
- [9] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. In *VLDB*, 2008.
- [10] C. Dunham. Remez algorithm for Chebyshev approximation with interpolation. *Computing*, 28(1), 1982.
- [11] M. Garofalakis and A. Kumar. Deterministic wavelet thresholding for maximum error metrics. In *PODS*, 2004.
- [12] S. Guha and B. Harb. Wavelet synopsis for data streams: Minimizing noneuclidean error. In *KDD*, 2005.
- [13] H. Jagadish, H. Jin, B. Ooi, and K. Tan. Global optimization of histograms. In *SIGMOD*, 2001.
- [14] H. V. Jagadish, R. T. Ng, B. C. Ooi, and A. K. H. Tung. Itcompress: An iterative semantic compression algorithm. In *ICDE*, 2004.
- [15] R. Jampani, F. Xu, M. Wu, L. Perez, C. Jermaine, and P. Haas. Mcdm: A monte carlo approach to managing uncertain data. In *SIGMOD*, 2008.
- [16] P. Karras, D. Sacharidis, and N. Mamoulis. Exploiting duality in summarization with deterministic guarantees. In *KDD*, 2007.
- [17] F. Korn, T. Johnson, and H. Jagadish. Range selectivity estimation for continuous attributes. In *SSDBM*, 1999.
- [18] U. Lerner, B. Moses, M. Scott, S. McIlraith, and D. Koller. Monitoring a Complex Physical System using a Hybrid Dynamic Bayes Net. In *ICML*, 2002.
- [19] J. Mason. *Chebyshev polynomials*. Chapman & Hall/CRC, Boca Raton, FL, 2003.
- [20] D. S. Nilesch Dalvi. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.
- [21] A. Ralston. Rational Chebyshev approximation by Remes' algorithms. *Numerische Mathematik*, 7(4), 1965.
- [22] T. Rivlin. *Chebyshev polynomials: From Approximation Theory to Algebra and Number Theory*. Wiley, 1974.
- [23] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, 2007.
- [24] S. Singh, C. Mayfield, R. Shah, S. Prabhakar, S. Hambrusch, J. Neville, and R. Cheng. Database support for probabilistic attributes and tuples. In *ICDE*, 2008.
- [25] L. Veidinger. On the numerical determination of the best approximations in the Chebyshev sense. *Numerische Mathematik*, 2(1), 1960.

APPENDIX

A. ACKNOWLEDGMENTS

The first author was supported by an NSF graduate research fellowship. Additionally, this research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

The authors would also like to thank Dr. Panagiotis Karras for offering his source code and assistance in computing unrestricted optimal wavelets. Lastly, the authors thank the anonymous reviewers for their useful comments.

B. ERROR BOUNDS

LEMMA 2. For each Chebyshev basis polynomial, $T_n(x) = 1$ when $x = 1$.

PROOF. First, recall that we are only concerned with the function values within the closed interval $[-1, 1]$. It is easy to see that this is the case in the first two basis polynomials.

$$T_0(x) = 1$$

$$T_1(x) = x$$

Additionally, using the recursive definition we see the inductive argument.

$$T_n(x) = 2xT_{n-2}(x) - T_{n-1}(x)$$

If $T_{n-2} = 1$ and $T_{n-1} = 1$ at $x = 1$ then we have the following

$$T_n(1) = 2(1)(1) - (1) = 1$$

□

Proof of theorem 3:

PROOF. By definition, we have

$$P_m - c_1T_1 - \dots - c_mT_m = 0$$

We represent the set of basis function to economize as G , so we can add this set from both sides of (4).

$$P_m - c_1T_1 - \dots - c_mT_m + \left(\sum_{g \in G} c_g T_g\right) = \sum_{g \in G} c_g T_g$$

$$P_m - \left(\sum_{g \notin G} c_m T_m\right) = \sum_{g \in G} c_g T_g$$

We can put this into a form more similar to the L_∞ -norm by taking the maximum value with respect to x of both sides.

$$\max_x |P_m(x) - \left(\sum_{g \notin G} c_g T_g(x)\right)| = \max_x \left| \sum_{g \in G} c_g T_g(x) \right| \quad (4)$$

From (4), we can derive upper and lower bounds of the approximation L_∞ error. We simplify the right side of the equation by setting $x = 1$ since, by lemma 2, $T_g(1) = 1$, $\forall g$, resulting in a lower bound of the L_∞ error.

$$\max_x |P_m(x) - \left(\sum_{g \notin G} c_g T_g(x)\right)| \geq \left| \sum_{g \in G} c_g \right| \quad (5)$$

To see why this is a lower bound, consider the following two cases.

1. \max_x occurs at $x = 1$. In this case, equality will hold since the assumption we made on the right side of the equation holds on the left as well.
2. \max_x occurs at $x \neq 1$. In this case, both sides of the equation are maximized at an x other than $x = 1$. Since we assume $x = 1$ on the right side of the equation, obviously this term will be less than the left side.

For the upper bound, we take equation 4 and move the absolute value inside of the summation (on the right hand side) creating an inequality. Additionally, we know that $|T_g(x)| \leq 1$, which gives us the following inequality.

$$\begin{aligned} \max_x |P_m(x) - (\sum_{g \notin G} c_g T_g(x))| &\leq \sum_{g \in G} |c_g T_g(x)| \\ &\leq \sum_{g \in G} |c_g| \end{aligned}$$

We take the far right hand side of this equations our upper bound.

$$\max_x |P_m(x) - (\sum_{g \notin G} c_g T_g(x))| \leq \sum_{g \in G} |c_g| \quad (6)$$

□

C. COEFFICIENT ALLOCATION

LEMMA 3. For an optimal allocation X_b^* , reducing the number of coefficients allocated to any approximation will increase the global error. That is, $\forall j$ s.t. $x_j > 0$, $\text{err}(f_j, x_j - 1) \geq \text{Err}(X_b^*)$.

PROOF. Assume that $\text{err}(f_j, x_j - 1) < \text{Err}(X_b^*)$. There must exist some function f_i , $i \neq j$ such that $\text{err}(f_i, x_i) = \text{Err}(X_b^*)$. In fact there are two cases to consider.

First, the case in which there exists only one signal influencing the global error. Here we see that transferring a coefficient from f_j to f_i would produce a new allocation X_b^1 such that

$$\text{Err}(X_b^1) < \text{Err}(X_b^*)$$

This causes a contradiction since X_b^* is optimal.

Second, the case in which there exist multiple functions, $F_{\max}(X_b^*)$, which influence the global error. Here, an optimal allocation will minimize the size of this set. Transferring a coefficient from f_j to any function in the set $F_{\max}(X_b^*)$, to form a new allocation X_b^1 gives us: $|F_{\max}(X_b^*)| > |F_{\max}(X_b^1)|$ since f_j is not contained in this set. Again, this causes a contradiction since X_b^* is optimal, thus the lemma is proved. □

Proof of theorem 4:

PROOF. We use δ_i to denote a unit vector of length k with such that $\delta_i(i) = 1$ and $\delta_i(j) = 0 \forall j \neq i$. $\widehat{\delta}_i$ is a vector of length k with such that $\widehat{\delta}_i(i) > 1$ and $\sum_j \widehat{\delta}_i(j) = 1$. Each element in any δ vector may take on any integer value (positive values denotes adding space and negative values denote removing space).

We assume f_i is one of the functions with a maximum local error in the current allocation: $\text{err}(f_i, x_i) \geq \text{err}(f_j, x_j) \forall j \neq i$. Because we are interested in the maximum L_∞ error metric, the only way to decrease the global error is to improve the worst approximation. Hence, we partition all possible transformations from X_b^* to X_{b+1}^* into the following two cases

$$X_{b+1}^{1*} = X_b^* + \widehat{\delta}_i \quad (7)$$

$$X_{b+1}^{2*} = X_b^* + \delta_i \quad (8)$$

where (8) is the greedy approach and (7) represents all other possible transforms, where x_i is increased by more than one.

In the case of (7), there must be some index $j \neq i$ such that $x_{j,b} > x_{j,b+1}$. By definition, $\text{err}(f_j, x_{j,b+1}) > \text{err}(f_j, x_{j,b})$, causing the global error to (possibly) increase since $\text{Err}(X_b^*) \leq \text{Err}(X_b)$. To see this, we can view (7) as a two step transform. We describe the simple case in which the coefficient of the δ vectors is one, however, the following logic is applicable for any coefficients.

$$X_b^{1*} = X_b^* + \delta_i - \delta_j \quad (9)$$

$$X_{b+1}^{1*} = X_b^{1*} + \delta_i + \widetilde{\delta}_{i,j} \quad (10)$$

where $\widetilde{\delta}$ contains the portion of the transformation that does not include functions f_i and f_j .

After (9), $\text{Err}(X_b^*) \leq \text{Err}(X_b^{1*})$ by definition of optimality, since both are allocations over a budget of b . Additionally, from lemma 3 we see that f_j is guaranteed to influence the global error. However, in the second step of the transform, (10), x_j is not increased, so the global error of this allocation is left unchanged and $\text{Err}(X_{b+1}^{1*}) \geq \text{Err}(X_b)$, thus a contradiction.

In the case of (8) we only assign the unallocated space to f_i . This guarantees that $\text{Err}(X_{b+1}^{2*}) \leq \text{Err}(X_b)$ and that the set of signals contributing to the global error is minimized. We conclude with the following inequality:

$$\text{Err}(X_{b+1}^{2*}) \leq \text{Err}(X_b^*) \leq \text{Err}(X_{b+1}^{1*})$$

Thus X_{b+1}^{2*} is an optimal allocation for a budget of $b + 1$. □

We note that monotonicity is all that is required in this theorem since we are interested in the L_∞ error. As long as the local error is guaranteed to decrease with an increase in space, we can always reduce the error by allocating more space to the tuple with the maximum error.

D. CLUSTERING

Proof of theorem 5:

PROOF. Given Lemma 1, consider the upper bound (equation 6) of the error function f_z and its approximation in the extreme case in which we prune all Chebyshev coefficients.

$$\max_x |f_z(x)| \leq \sum_{g \in G} |c_z(g)|$$

$$\max_x |f_1(x) - f_2(x)| \leq \sum_{g \in G} |c_1(g) - c_2(g)|$$

$$\text{dist}_{L_\infty}(f_1, f_2) \leq \text{dist}_{L_1}(c_1, c_2)$$

□