Entropy Based Approximate Querying and Exploration of Datacubes

Themistoklis Palpanas *

themis@cs.toronto.edu

Dept. of Computer Science

University of Toronto

10 King's College Road, Toronto

Ontario, M5S 3G4, CANADA

Nick Koudas

koudas@research.att.com

AT&T Labs-Research

Department of Computer Science

University of Toronto

Technical Report CSRG-409

Abstract

Much research has been devoted to the efficient computation of relational aggregations and specifically the efficient execution of the datacube operation. In this paper we consider the inverse problem, that of deriving (approximately) the original data from the aggregates.

We motivate this problem in the context of two specific application areas, that of approximate query answering and data analysis. We propose a framework based on the notion of information entropy, that enables us to estimate the original values in a data set, given only aggregated information about it. We then show how approximate queries on the data from which the aggregates were derived can be performed using our framework. We also describe an alternate utility of the proposed framework, that enables us to identify values that deviate from the underlying data distribution, suitable for data mining purposes.

Finally, we present a detailed performance study of the algorithms using both real and synthetic data, highlighting the benefits of our approach as well as the efficiency of the proposed solutions.

*Contact author and address.

1 Introduction

In recent years, there has been an increasing interest in warehousing technology and OLAP applications which view data as having multiple dimensions, with hierarchies defined on each dimension. Users typically employ OLAP applications for decision making. They inquire about the values, and analyze the behavior of measure attributes in terms of their dimensions. Consider for example Figure 1(a) showing a typical OLAP table, with two dimension attributes (location and jeans), and hierarchies defined per dimension. Users, for analysis purposes, commonly inquire about the values of aggregates, like the total volume of sales of jeans in NY state. Such aggregates can be precomputed, by the execution of the datacube operator [GBLP96], and queries of this kind can be efficiently supported.

The volume of data stored in OLAP tables is typically huge, in the order of multiple gigabytes, or even terabytes. In some cases, users store on disks only a subset of the data they own. They move the rest of the detailed data on tertiary storage systems, or even take them off line, while keeping only a small number of aggregated data that are of interest. A common example is history sales data, where only the data of the recent few years are stored online, and the rest are archived. When older years are archived, possibly data of finer granularity, such as sales of women's Levi's in New York city, are permanently deleted, and only aggregated values are stored for future reference, for example, sales of Levi's in each state. In other cases, even if the data remain online, they are aggregated not only to support user queries faster, but also to save space. For example, in several organizations, such as AT&T, terabytes of data are generated every day. It is common that these data are aggregated to save storage space, thus the detailed information that produced the aggregates is lost.

Given the summarized form of data, users are often interested in inquiring about the data that generated the summarized form. These data either might have expired from the warehouse, and thus are no longer available, or they might not be available online. In such cases, generating good estimates for the original data in response to queries is a pressing concern. In the example of Figure 1(a), assume that for the state of NY (i.e., the upper half of the table) we only store the *aggregated* sales for each city and for each category as shown in Figure 1(b), and that we have deleted all the detailed values. The users might want to inquire about the number of men's Levi's jeans sold in Queens NY (a point query), or they might request the number of men's and woman's jeans sold in each city of NY state (a range query). We want to be able to answer these queries by using only the stored aggregate values. In this work we present a technique that tackles this problem. Similar issues arise in transaction recording systems [JMS95] as well as in statistical databases [AM92, Mal93].

Even if the original base data exist, the ability to reconstruct the original data from the summaries is of great value. Computing summaries is essentially a data reduction process in which great information loss takes place. In order to reconstruct the data, various assumptions have to be made

			7 All			
			5 Levi's		6 CK	
			1 men's	2 women's	3 men's	4 women's
7 All	5 NY	1 Queens	40	10	50	5
		2 New York	15	45	65	10
	6 Ontario	3 Toronto	15	40	65	18
		4 Ottawa	55	40	35	30

(a) The entire dataset.

			55	55	115	15
NY	Queens	105	x	x	x	x
	New York	135	x	x	x	x

(b) Aggregated values for the upper half of the dataset.

Figure 1: Example dimension hierarchies on two dimensional sales data.

about the statistical properties of the reduced data. Given the reconstructed and the original data at hand, we can test how strong our assumptions about the original data were, just by comparing the two. This is useful in reasoning about the properties of the underlying data set and could be of great value in data mining. It can help detect correlations in the data, and identify deviations, that is, values that do not conform to the underlying model. Such results are of great interest to the analyst, because they indicate local or global abnormalities.

In this paper, we propose the use of an information theoretic principle for the reconstruction of the original data from the summarized forms. Since data in a warehouse are aggregated as a means of summarization, we examine the problem of reconstructing the original data from the aggregates.

Our reconstruction technique is based on the well recognized and widely applicable information theoretic principle of *maximum entropy* [KK92]. We present algorithms for the efficient reconstruction of data from the aggregates. Moreover, using an information theoretic formalism we identify and describe in detail an alternate benefit of the proposed reconstruction techniques, namely the ability to 'rank' each reconstructed value by its potential 'interest' to the user, as a means of aiding data analysis. The notion of interest in data mining is a difficult problem and several approaches have been proposed for its formal definition [ST96]. Although all the measures of interest are subjective and heavily dependent on the application, we argue that an information theoretic approach to this problem, besides being mathematically rigorous, appears conceptually appealing as well.

1.1 Our Contributions

In this paper, we make the following contributions:

- We propose a method for reconstructing multidimensional values from aggregate data. In the OLAP environment our technique uses the already computed aggregated values, and there is no need for any additional special data structures.
- We describe an extension to the above method, in which we are able to provide quality (error) guarantees for the reconstruction. Moreover, the quality of the reconstructed information can be controlled by the user, achieving any degree of desired accuracy.
- We present a method to identify and rank deviations in multidimensional datasets, that is, values that do not follow in general the underlying data distribution. These values are of particular interest to the analysts since they indicate local or global abnormalities. The power of the method we propose is that it does not depend on any a priori or domain knowledge for the problem at hand, and it also does not require any parameter settings or calibrations.
- The properties and special characteristics of the above methods are explored with an extensive experimental evaluation, using both synthetic and real datasets.

1.2 Paper Outline

The outline of the paper is as follows. Section 2 reviews related work. In Section 3 we present some background material necessary for the rest of the paper and Section 4 presents the reconstruction algorithm. Section 5 discusses the two particular application scenarios of the algorithm, and associated optimization problems. In Section 6 we present experimental results evaluating the performance and the utility of the proposed algorithms, and we present our conclusions in Section 7.

2 Related Work

The principle of maximum entropy [CT91] has been successfully applied in different domains, including linguistics [CR99] [BPP96] and databases [FJS97]. Faloutsos et al. apply maximum entropy in addition to other techniques, for one dimensional data reconstruction [FJS97]. In a sense our work, generalizes the work of Faloutsos et al. in multiple dimensions. Maximum entropy has also been used for the identification of 'interesting' correlations in data [Tho98].

There exists a sizeable bibliography in histogramming techniques and approximate query answering [IP95] [PIHS96] [JKM+98] [VWI98] [AGPR99] [SFB99] [BS97] [BW00]. Our approach is fundamentally different. Previous work focused on the problem of data reconstruction by constructing specialized summarized representations (typically histograms) of the data. We argue, that since data are already stored in an aggregated form in the warehouse, it is imperative to examine the quality of reconstruction one can attain from the aggregates. This approach could potentially have a great deal of benefit, if the quality of the reconstruction is good.

The problem of identifying interesting values in a dataset is related to deviation detection. Arning et al. [AAR96] try to identify the subset of a database that is most dissimilar to the rest of the data. Other approaches discuss algorithms specialized in metric spaces that scale to large datasets [KN98] [KN99]. The drawback of the above approaches is that the user is required to come up with the right selection of functions and parameters which requires a great deal of effort. Our algorithm does not need such input, making the whole procedure less cumbersome and more robust. In that sense, our work is closer to the framework proposed by Sarawagi et al. [SAM98]. They describe an algorithm that mines the data in a data cube for exceptions. However, this method is computationally expensive, depends on the computation of the entire data cube, and cannot accommodate updates.

3 Background

Consider a schema $R = (A_1, A_2, \ldots, A_n, Y)$ and r an instance of R. $A_1, \ldots, A_n$ are attributes having some specified type and Y is a measure attribute. Attribute Y could represent volume of sales, dollar amount, number of calls, etc. Attributes $A_i, 1 \leq i \leq n$ include dimension attributes and hierarchies possibly defined on them. For example in Figure 1(a) the schema is R(STATE, CITY, BRAND, GENDER, SALES). Attributes STATE and BRAND define hierarchies on CITY and GENDER respectively.

With a schema of n attributes, assume that h attributes define hierarchies on the remaining $d = n - h$ attributes. We can enumerate the nodes of hierarchy $i, 1 \leq i \leq d$ starting bottom up, such that $h_i = j$ denotes the jth node of hierarchy i. Viewing the table of Figure 1(a) with its dimensions and hierarchies as a multidimensional grid, following Jagadish et al. [JLS99] we refer to the d-dimensional vector $(h_1, \ldots, h_d)$ as a *grid query*. For example, the grid query $(2, 1)$ is a point query asking for the sales of women's Levi's in Queens, while the grid query $(7, 5)$ is a range query referring to the entire upper half of the dataset. In our experience, grid queries are the most common in warehouse environments. The reason we are interested in this special class of queries is that it significantly reduces the space of possible queries. For the rest of this paper we will refer to grid queries simply as queries, and to the subset of the entire dataset they involve as dataset. Note though, that by answering a grid query using our techniques, we can also answer any other query asking for a subset of the answer we computed.

A basic observation about any instance r of R is that it can be viewed as a discrete n dimensional probability distribution, $P_r(A_1, \ldots A_n)$. This can be accomplished by normalizing the value Y on each row of r by the sum of all Y values. The analogy between r and P_r can be extended further. Consider for example the following query:

```
SELECT  A_1, A_2, ..., A_{n-1}, sum(Y)
FROM    r
GROUPBY A_1, A_2, ..., A_{n-1}.
```

The outcome of this query is one of the n marginal distributions of P_r of order (number of variables) $n-1$. All we need to do, is normalize $sum(Y)$ by the sum of all Y values. (This normalization is needed for mathematical correctness, but is not required by the techniques discussed in this paper.) Reasoning similarly, one can draw the analogy between all the group by's on r and all the marginal distributions of P_r. Thus, we can view the problem of reconstructing r from its aggregates as analogous to the problem of reconstructing an n dimensional probability distribution from a number of its marginal distributions. Based on this analogy, in the rest of this paper, we use the terms group by on instance r and marginal distribution of P_r interchangeably.

3.1 Maximum Entropy Distributions

Let $P(A_1, \ldots, A_n)$ be an n dimensional discrete probability distribution, to be estimated from a number of its marginals. With n variables, there are $2^n - 2$ marginals (excluding the grand total and the base data) of P in total. Moreover, there are $\binom{n}{k}$ marginals with k variables (equivalently of order k). Let S be an arbitrary subset of the powerset of $X = \{A_1, \ldots, A_n\}$. The problem of *maximum entropy estimation of* P is defined as follows:

Problem 1 *Maximize the entropy* $H(P)$ *of* P *over all probability distributions satisfying:*

$P(X) \geq 0$, *with equality outside the support of* P,

$\sum P(X) = 1$, *and*

$\forall i \in S, \sum_{j \in S-i} P(j) = P(i)$,

where $P(j)$ *is a marginal distribution of* P, *with* $j \in S$.

The maximum entropy estimation of P is a model fitting technique. It finds the model with the 'least' information or fewest assumptions given the specified constraints. The method appears ideal for this estimation problem as it is designed for lack of data rather than an excess thereof. The overriding principle in maximum entropy is that when nothing is known the distribution should be as uniform as possible. The constraints (the marginal distributions in our case) specify the regions where the estimated distribution should be minimally non-uniform in order to satisfy these constraints.

3.2 Properties of Maximum Entropy

Let $P(X)$ be an n-dimensional probability distribution. We denote by $P_i, 1 \leq i \leq n-1$ the maximum entropy approximation to P using only marginals of order i as constraints. Then the following theorems hold.

Theorem 1 *[Kul68] Let p be a member of the class of discrete probability distributions that can be constructed using a specified set of lower order marginal distributions of order i. Then, among all distributions p, P_i minimizes:*

$$D(p, P) = \sum_X p(X) \log \frac{p(X)}{P(X)} \tag{1}$$

The measure D is known in the literature as the relative entropy [CT91] and measures the similarity of two probability distributions. More precisely D is a measure of the inefficiency of assuming that the distribution is p when the true distribution is P. In statistics D arises as an expected logarithm of the likelihood ratio, and one can show that by minimizing $D(p, P)$ we also minimize the χ^2 test between p and P [Kul68].

Theorem 2 *[Kul68] Let $P_i, 1 \leq i \leq n-1$ be the maximum entropy approximation of P using only marginals of order i. Then the following inequality holds:*

$$D(P_1, P) \geq D(P_2, P) \ldots \geq D(P_{n-1}, P) \tag{2}$$

Theorem 2 states that a better estimation of P can be performed by using marginals of order $i+1$ than marginals of order i, for $1 \leq i \leq n-1$.

3.3 Computing the Maximum Entropy Solution

Problem 1 is a constraint optimization problem that is not amenable to a general closed form solution. The problem is in general under-constrained and the theory of inverse problems is applicable for its solution. The standard technique for solving this maximization problem is the method of *Lagrange Multipliers* [Ber82]. One has to form equations based on the optimization objectives and the constraints imposed, and solve a rather complex system. In its one dimensional (single variable) version this problem is amenable to an efficient solution [FJS97]. However, the efficient solution of the one dimensional case does not generalize in multiple dimensions, because the computational cost is prohibitive. Moreover, even if one is willing to devote the computational power to solve this problem in the multidimensional case, a different system of equations has to be derived each time, depending on the specified constraints (marginal distributions available). This is not very appealing for automation. Ideally, we are after an algorithmic approach that can operate directly on the specified constraints and derive the estimation with minimal human intervention.

4 Algorithmic Solution

In this section, we propose the use of an algorithmic approach for the solution of problem 1. The technique is called *Iterative Proportional Fitting* (IPF), and was introduced in [DS40]. It is an iterative algorithm that converges to the maximum entropy solution. IPF has the following properties [BFH75]:

1. it always converges monotonically to the required unique maximum entropy estimation, given a number of marginals;

2. a stopping rule may be used that ensures accuracy to any desired degree in the solution;

3. the estimates depend only on a particular small set of marginals;

4. convergence, as well as the speed of convergence, are not directly affected by the starting values;

5. in some cases, convergence to the exact estimates is achieved after only a single iterative cycle.

4.1 Description of IPF

Let $r(A_1, A_2, \ldots, A_n, Y)$ be a relational instance. We specify a multidimensional range of interest $[d_{1_s}, d_{1_e}] \times \ldots \times [d_{d_s}, d_{d_e}]$, defined by a d-dimensional grid query $Q = (h_1, \ldots, h_d)$. We assume that all the marginals of order $k, 1 \leq k \leq d-1$ needed for the reconstruction of Q have been located[1] and are given as constraints on Problem 1. To illustrate the above with an example we use the dataset of Figure 1(a). The highlighted box in the figure defines a range query; the corresponding marginals of order 1 for this query are the row and column aggregates shown in Figure 1(b).

Without loss of generality, assume that the set $\{A_1, \ldots A_d\}$ contains all the attributes with hierarchies defined on them. Let S_k be the set containing all subsets of the powerset of $\{A_1, \ldots, A_d\}$ with k elements. Clearly $|S_k| = \binom{d}{k}$. Let $P_i(j), 1 \leq i \leq \binom{d}{k}, j \in S_k$, be the marginals of P of order k corresponding to Q. On each $P_i(j)$, the aggregation has been computed on all attributes not present in j.

We denote as $Y_{A_1 A_2 \ldots A_n}$, the value of attribute Y for the specific combination of the A_i, attribute values (d of those are specified by the grid query and the remaining $n-d$ from the hierarchy). We denote as $\hat{Y^{(t)}}_{A_1 A_2 \ldots A_n}$ the estimate of the value of $Y_{A_1 A_2 \ldots A_n}$ during the t-th iteration of the algorithm. IPF starts the reconstruction by initializing a d-dimensional grid, G, of size $d_{i_e} - d_{i_s} + 1$ per dimension i, to one. We refer to each element of the d-dimensional grid as a *cell*. In addition, given the initialized G, it computes its marginals of order k. Let $\hat{P}_i^{(t)}(j)$, denote the marginals computed from G in the t-th iteration of the algorithm. Denote $\hat{P}_i^0(j)$, the marginals after the initialization.

At each iteration, IPF loops over all marginals $i, 1 \leq i \leq \binom{d}{k}$, and all grid cells, $l_1 \in [d_{1_s}, d_{1_e}], \ldots, l_n \in [d_{d_s}, d_{d_e}]$, and adjusts the values of the grid cells according to the formula:

$$\hat{Y}^{(t+1)}_{l_1 l_2 \ldots, l_d} = \hat{Y}^{(t)}_{l_1 l_2 \ldots l_d} \frac{P_i(j)}{\hat{P}_i^{(t)}(j)} \tag{3}$$

[1]We defer our discussion of the issues related to the identification of the suitable marginals to Section 5.

This procedure is guaranteed to converge to the maximum entropy estimation of P given the specified collection of marginals. The estimates converge in a monotonically decreasing fashion, which allows the application of a stopping rule. Commonly, we choose to terminate the iterations when the change in each individual estimate becomes smaller than some user specified δ value. For all the experiments presented herein we set δ to 10% of the median of the values designated by Q. We chose the median because it is not affected by any extreme values, and it can be efficiently computed [MRL99] and stored in the DBMS. In addition, as we show in the experimental section, the algorithm is not very sensitive to the value of δ, making the specific choice less important. A skeleton of the IPF algorithm is given in Figure 2.

Objective: Compute the maximum entropy estimates $\hat{Y}_{A_1 A_2 \ldots A_d}$ of a given d-dimensional dataset $Y_{A_1 A_2 \ldots A_d}$.
Input: A set of $\binom{d}{k}$ marginals $P_i(j), i = 1, \ldots, \binom{d}{k}$ and a stopping threshold δ.
Output: The maximum entropy estimates $\hat{Y}_{A_1 A_2 \ldots A_d}$

let step t:=0;
initialize $\hat{Y}_{A_1 A_2 \ldots A_d}$;
do {
  for i:=1 to e:=$\binom{d}{k}$ {
    $\forall l_1 \in [d_{1_s}, d_{1_e}], \ldots, l_d \in [d_{d_s}, d_{d_e}]$
    $\hat{Y}^{(t+1)}_{l_1 l_2 \ldots l_d} = \hat{Y}^{(t)}_{l_1 l_2 \ldots l_d} \frac{P_i(j)}{\hat{P}^{(t)}_i(j)}$
    t++;
  }
} while $\left| \hat{Y}^{(t)}_{A_1 A_2 \ldots A_d} - \hat{Y}^{(t-e)}_{A_1 A_2 \ldots A_d} \right| \geq \delta$;

Figure 2: The IPF algorithm.

4.2 An Example

We explain the operation of the algorithm with an example. For illustration purposes, we discuss an example in 2-dimensions, involving attributes A_1 and A_2. The generalization to higher dimensions is straightforward.

Assume that we have the dataset depicted in Figure 3(a), and we want to focus on the highlighted subset. Figure 3(b) shows the particular subset along with its marginals. The IPF algorithm starts by initializing the estimates $\hat{Y}^{(0)}_{l_1 l_2} = 1, l_1 = 2, \ldots, 4; l_2 = 1, \ldots, 4$ (Figure 3(c)). Then, in the subsequent steps, it fits the marginals one by one. First, it fits $P_1(A_1)$ using the formula $\hat{Y}^{(1)}_{l_1 l_2} = \hat{Y}^{(0)}_{l_1 l_2} \frac{P_1(A_1)}{\hat{P}^{(0)}_1(A_1)}, \forall l_1 \in [2, \ldots, 4], l_2 \in [1, \ldots, 4]$ and we obtain the table of Figure 3(d). Then, it fits marginal $P_2(A_2)$ using the formula $\hat{Y}^{(2)}_{l_1 l_2} = \hat{Y}^{(1)}_{l_1 l_2} \frac{P_2(A_2)}{\hat{P}^{(1)}_2(A_2)}, \forall l_1 \in [2, \ldots, 4], l_2 \in [1, \ldots, 4]$. The result is depicted in Figure 3(e), and this is the final set of estimates. Indeed, if we run one more iteration of the algorithm the elementary cell estimates will not change. Therefore, the condition at the end of the main loop of the algorithm will be satisfied, and the procedure will terminate.

	1	2	3	4	5
1	22	19	28	20	33
2	27	38	43	32	35
3	25	31	40	28	29
4	26	35	37	30	36
5	27	38	43	81	41

(a)

	1	2	3	4	$P_1(A_1)$
2	27	38	43	32	140
3	25	31	40	28	124
4	26	35	37	30	128
$P_2(A_2)$	78	104	120	90	

(b)

	1	2	3	4	$\hat{P}_1^{(0)}(A_1)$
2	1	1	1	1	4
3	1	1	1	1	4
4	1	1	1	1	4
$\hat{P}_2^{(0)}(A_2)$	3	3	3	3	

(c)

	1	2	3	4	$\hat{P}_1^{(1)}(A_1)$
2	35.00	35.00	35.00	35.00	140
3	31.00	31.00	31.00	31.00	124
4	32.00	32.00	32.00	32.00	128
$\hat{P}_2^{(1)}(A_2)$	98.00	98.00	98.00	98.00	

(d)

	1	2	3	4	$\hat{P}_1^{(2)}(A_1)$
2	27.86	37.14	42.86	32.14	140
3	24.67	32.90	37.96	28.47	124
4	25.47	33.96	39.18	29.39	128
$\hat{P}_2^{(2)}(A_2)$	78	104	120	90	

(e)

Figure 3: An example of applying the IPF algorithm. Figure (a) shows the whole dataset, and the portion of it that we wish to estimate. In (b) we have the subset we focus on, and its two corresponding marginals. The initialization step is depicted in (c). Figures (d) and (e) show how the algorithm fits the two marginals one at a time.

4.3 Algorithmic Complexity and Implementation Details

The IPF algorithm requires as input the marginals corresponding to a specific query Q, and then iterates over a grid G which is the same size as the result set of Q. We can safely assume that the marginals fit in main memory, but this may not be true for G.

If the available memory is large enough to fit the grid G then the algorithm only needs to read from disk the marginals, which is a negligible cost[2]. All the subsequent operations take place in main memory. Taking into consideration the large sizes of memory that are commonplace nowadays, we expect that the algorithm will be able to provide fast answers to a significant number of queries by operating on memory-resident data.

In the case where G does not fit in memory, the algorithm has an increased I/O cost. During each iteration it makes $\binom{n}{k}$ passes over G, where n is the dimensionality of Q, and k the order of the

[2]As we will show in Section 5 this involves just a selection query to the DBMS storing the underlying data.

marginals that we use for the estimation. After having computed the estimates based on a single marginal for all the values in G, the algorithm has to make one more disk pass over G in order to calculate the new estimated marginals. In our implementation, we managed to cut the number of passes in *half* by incorporating the update of the estimated marginals with the computation of the base values, reducing the cost of the algorithm to $[\binom{n}{k}t]$ passes over G, where t is the number of iterations. This called for efficient indexes on the marginals, which were implemented using hashing techniques. The results that we report in the experimental section illustrate the **worst case** scenario, where the dataset does not fit in main memory.

The number of iterations the algorithm performs until convergence is achieved is usually small. Typically, six or fewer iterations are enough to get an accurate estimate, even for data sets of large dimensionality. In addition, since the algorithm is guaranteed to converge to the final estimates in a monotonic fashion, the user can stop the procedure when she is happy enough with the accuracy of the current estimated values. Even though the parameter δ (which controls the accuracy of the estimation) is user defined, the above approach can still result in substantial savings in the computation time. An example of such a case is when the user is interested in a subset of the values that are being estimated, and for which the desired accuracy has already been achieved. Since this is a user interface issue, we do not pursue it further in the experimental section.

4.4 Error Guarantees for the Approximation

The reconstruction process is only dependent on the *marginals* of the base data. This implies a significant reduction in the available information from which the base data will be estimated. For various applications, being able to provide error guarantees for the reconstruction of individual values is imperative. We observe that the estimation error is going to be significant at those values where the difference between the original data values and the estimated ones is large. We can safely assume that at the time of the computation of the aggregates the original data are still available.

Let W be the set of grid queries of interest. One approach to provide error guarantees for each query in W would be the following: estimate the values of each query in W, while the original data are still available, compute the largest difference (between the actual and the estimated value) for each query in W, and store them separately. This would incure a storage overhead of $O(|W|)$. More formally, let us denote by Y_i the original value of some cell i, and with $\hat{Y}_i$ its estimated value. We can use the absolute difference $d_i = |\hat{Y}_i - Y_i|$ in order to provide an upper bound for the error. Assuming that a specific grid query encompasses N cells from the base data, the upper bound can be calculated using the formula[3] $N \cdot \max\{d_i\}, 1 \leq i \leq N$. Thus, the total error for the query is not going to be greater than N times the largest individual cell error.

The above error bound provides an indication of the accuracy of the reconstruction. Yet, some

[3]Without loss of generality, we assume that the error metric is the Root Mean Square Error. Similar arguments hold if we choose other error metrics as well.

applications may require tighter error guarantees. In order to provide such guarantees we introduce the following approach: store a number k (user defined) of the largest estimation errors for each query in W. Given a query in W that involves a number of the cells whose correct values have been explicitly stored, the reconstruction algorithm uses these correct values, and thus induces no error for the specific cells. Since we have chosen to store the cells with the largest errors, the overall error for the query will be dramatically reduced. If the error guarantee per query should be bounded by a user specified value, we can choose k (the number of values to store) such that the overall error of reconstruction satisfies the error bound.

In many cases the number of cells for which the approximation is really poor will be relatively small. Therefore, the number of values that need to be materialized separately will be small as well. In fact, we expect that in most datasets the distribution of the reconstruction error magnitudes will follow a Zipf distribution. In the experimental section we evaluate this argument and we present graphs depicting the distribution of errors for the real datasets we used. As will become evident from the graphs, only a minor percentage of cells exhibit high error rates, and thus, can be efficiently stored, inducing a small storage overhead.

5 Using IPF

In the following sections, we discuss issues related to the applications of IPF, both for query answering and knowledge discovery.

5.1 Query Answering

Given a d-dimensional grid query Q the algorithm has to determine the order of the marginals to use for reconstruction as well as the ranges of these marginals pertinent to Q. In light of Theorem 2, marginals of order $d-1$ will produce the most accurate estimate. In general, using marginals of order k with IPF will yield a more accurate estimate than using marginals of order $k-1$. In the experimental section we present graphs exploring the time and accuracy tradeoffs related to this choice.

Assuming one decides to use the marginals of order k, the exact marginals relevant to Q have to be determined. A simple rewriting of Q can produce the $\binom{d}{k}$ marginals of order k that should be queried in order to retrieve the values for IPF to operate on. Any grid query on $r(A_1, \ldots, A_n)$ where $A_1, \ldots A_d$ are dimension attributes and $A_{d+1} \ldots A_n$ define hierarchies on them, can be expressed in SQL as:

`SELECT` $A_1, A_2, \ldots A_d$

`FROM` r

`WHERE` $A_{d+1} = a_1$ and $\ldots A_n = a_{n-d}$, where the values $a_1, \ldots, a_{n-d}$ designate the multidimensional range of Q. Let $S = \{A_1, \ldots, A_d\}$. Then, the marginals of order $d-1$ relevant to the reconstruction of the grid query can be retrieved by issuing the following SQL query:

SELECT $S - A_i$

FROM r

WHERE $A_{d+1} = a_1$ and $\ldots A_n = a_{n-d}$,

for $1 \leq i \leq d$. This will give us all the $\binom{d}{d-1} = d$ marginals. Similarly, the marginals of order $d-2$ relevant to the reconstruction of the grid query can be retrieved by:

SELECT $S - \{A_i, A_j\}$

FROM r

WHERE $A_{d+1} = a_1 \ldots A_n = a_{n-d}$,

for $1 \leq i \leq d$, $i \leq j \leq d$. Reasoning similarly we can construct the expressions for the choice of marginals of any order.

5.2 Mining Interesting Patterns

In the case that the base data are available, the proposed reconstruction technique has a different utility. Maximum entropy reconstruction from a number of marginals is performed based on the assumption that the marginals of interest are pairwise independent. By reconstructing the data and comparing them with the base data, the validity of the pairwise independence assumption can be tested. Any data value that violates the pairwise independence assumption, will induce a larger reconstruction error than one that does not. This provides an automatic way to reason about the underlying correlations among attributes.

For example if the volume of sales of men's Levi's jeans in Queens incurs the largest estimation error than all sales of Levi's in the cities of NY state, we know that the volume of sales at Queens represents the strongest violation of the independence assumption among sales in NY state. We term such values *deviations*, because they deviate from the estimation model. Such values can potentially be of great interest to the analyst since they record actions out of the norm. This property can be utilized during grid query execution, since we can automatically identify and report to the analyst, values of potential interest.

The basis for terming a specific value as a deviant can be a measure of the distance between the actual and the estimated value, i.e., the estimation error, such as absolute difference. However, the aforementioned metric may not always produce qualitative results. An example of this case would be a dataset with values drawn from a uniform distribution with large variance. Then, most of the values in this dataset would qualify as deviants, which is not correct. In order to remedy this situation, we can use a formula which normalizes the estimation error of a value with respect to the standard deviation σ of all the estimation errors returned by the algorithm for the underlying dataset

$$s = \frac{|\hat{Y}_i - Y_i|}{\sigma},$$

where with Y_i we denote the original value of cell i, and with $\hat{Y}_i$ its estimation. Then, we choose a cutting threshold for s, that can effectively prune all the normal perturbations in the dataset, leaving us with only the large deviations. The above technique splits the sorted set of deviations into two regions: it assigns the statistically large deviations to the first region, and the rest to the second one. We refer to the boundary point between those two regions as the *cutoff point*. A commonly used threshold is for $s = 2$, which will prune 95% of the approximation errors as trivial, leaving only the largest 5% for consideration (the values follow from the properties of Normal distributions). The system can subsequently sort those deviating values, and pick the top-k among them. In the experimental section, we present graphs that visualize the ability of the algorithm to spot deviations, using both synthetic and real datasets.

5.3 Optimization Problems

So far we have assumed that we have enough space to materialize all the marginals needed to reconstruct each query in our workload. In the case where only a subset of the marginals can be materialized due to space constraints, one would be interested in obtaining the most benefit in reconstructing queries while satisfying the imposed constraints. The benefit in query reconstruction is defined as the number of "important" grid queries that can be reconstructed with the greatest accuracy, where importance may be user-defined, or determined by the frequency with which queries appear in the workload. The above situation gives rise to an optimization problem that one can show is NP-Hard, but is amenable to an efficient solution using heuristics. Moreover, assuming that we wish to utilize the reconstruction for identifying values of interest, ideally one would be interested in materializing the most informative marginals, that is the ones that are most likely to contain the values with the largest deviations from the pairwise independence assumption of the reconstruction process. This is a similar optimization problem, and can be solved using the same basic algorithms.

The aforementioned problems constitute ongoing work.

6 Experimental Evaluation

In the following sections we present the experiments we used to evaluate the efficiency and behavior of the IPF algorithm.

First, we describe the real and synthetic datasets we employed in our evaluation. Then, we assess the ability of the algorithm to make close estimates of some unknown multidimensional distributions. Finally, we examine another application of the IPF algorithm, namely mining interesting patterns and identifying deviations.

6.1 Description of Experiments

In order to test the IPF algorithm we used a mixture of synthetic and real datasets. The synthetic datasets produced are derived by sampling uniform and Gaussian data distributions.

Uniform: We produced datasets of dimensionality 2, 3, and 4. For each of the above datasets of different dimensionality, the values were drawn uniformly from the range $[0, 10]$ (*uniform10*), $[0, 100]$ (*uniform100*), and $[0, 1000]$ (*uniform1000*). The size of the datasets varied from 1,000 to 20,000 tuples. The variance of the values in the *uniform10* is smaller than that of *uniform100* which is in turn smaller than that of *uniform1000*.

Gaussian: We produced datasets of dimensionality 2, 3, and 4. The values were sampled from independent Gaussian distributions. For sigma σ we chose 3 different values that altered the distribution of the values in the data space. In the experiments we refer to these values for σ as *small*, *medium*, and *large*. Once more, the size of the datasets varied from 1,000 to 20,000 tuples.

We also experimented with three Gaussian datasets which had additional random noise coming from a uniform distribution. The third dataset was derived from a mixture of two multidimensional Gaussian distributions. We refer to these datasets as *gauss_small_sigma*, *gauss_large_sigma*, and *gauss_combined* respectively. The statistical properties of those datasets are reported in Table 1.

dataset	*min*	*max*	*mean*	*median*	*std.dev.*	*skew*
gauss_large_sigma	9.0	76.72	28.15	28.00	5.80	0.40
gauss_small_sigma	0.0	72.14	20.72	20.00	12.45	0.36
gauss_combined	0.0	106.68	24.66	22.00	13.57	0.53

Table 1: The statistical properties (mean, median, standard deviation, and skew) for the 3 synthetic datasets used in the experiments.

Real: The first of the real datasets, *calls* and *calls3*, are derived from AT&T proprietary data. They represent aggregated telephone calls in certain regions of North America over time. They are 2- and 3-dimensional, and their size is 10,000 tuples.

Finally, we used *census*, a dataset from the U.S. Census Bureau, containing information about the age, education, and income of individuals. It is a 4-dimensional dataset from which we extracted instances of 10,000-50,000 tuples.

Table 2 summarizes the statistical properties of *calls* and *census_10K*. The rest of the real datasets exhibit similar characteristics.

dataset	*min*	*max*	*mean*	*median*	*std.dev.*	*skew*
calls	1.0	729.00	18.01	4.00	37.79	5.37
census_10K	1000.00	196623.00	24735.73	20000.00	23449.87	3.67

Table 2: The statistical properties (mean, median, standard deviation, and skew) for the real datasets *calls* and *census_10K*.

The error metric that we report in the experiments is the *Root Mean Square Error (RMSE)*, defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N}},$$

where Y_i represents the original values in the dataset, $\hat{Y}_i$ the corresponding estimated values, and N is the total number of values in our dataset (remember that this refers to the dataset determined by some grid query).

Note that all the above datasets also have an additional measure attribute. Therefore, the total number of attributes for the datasets we used ranges from 3 to 5.

6.2 Exploring the Properties of the Algorithm

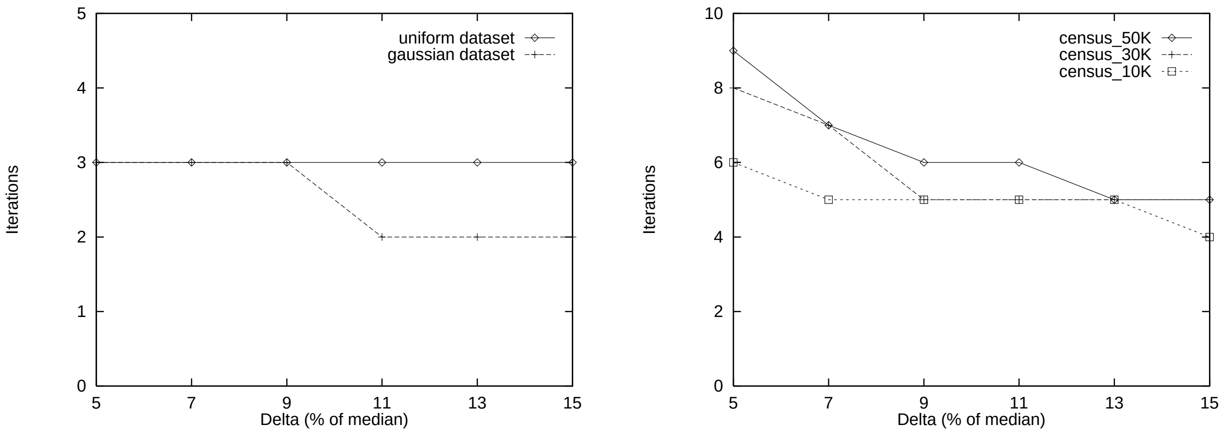
In this section we present experiments concerning the choice of the parameter δ, the run-time of the algorithm, and the property stated in Theorem 2.

The following experiments examine the effect of the parameter δ on the performance of the algorithm. We ran the algorithm with δ varying from 5% to 15% of the median (see Section 4.1) for several synthetic and real datasets. Table 3 shows that the error does not vary much when the

δ (% median)	*root mean square error*				
	gaussian	**uniform**	**census_10K**	**census_30K**	**census_50K**
~ 0	2.072	227.02	16889.76	18184.32	18564.82
5	2.072	227.09	16892.78	18185.64	18565.95
15	2.072	227.09	16895.72	18187.71	18567.60

Table 3: The statistical properties (mean, median, standard deviation, and skew) for the real datasets *calls* and *census_10K*.

value of δ increases. Even if we set δ to a very small value the benefit we get in the reduction of the error is not significant (results in the first row of the table). Since the choice of δ is not crucial for the error, we would like to pick a value that results in a small number of iterations for the least possible error. The graphs in Figure 4 depict the number of iterations needed for each value of δ. The largest variations in the number of iterations are observed in the datasets with the highest skew (Figure 4.b). Yet, even for those datasets setting δ to 10%, which is the setting we used throughout our experiments, proves to be a good choice. We expect that for most of the



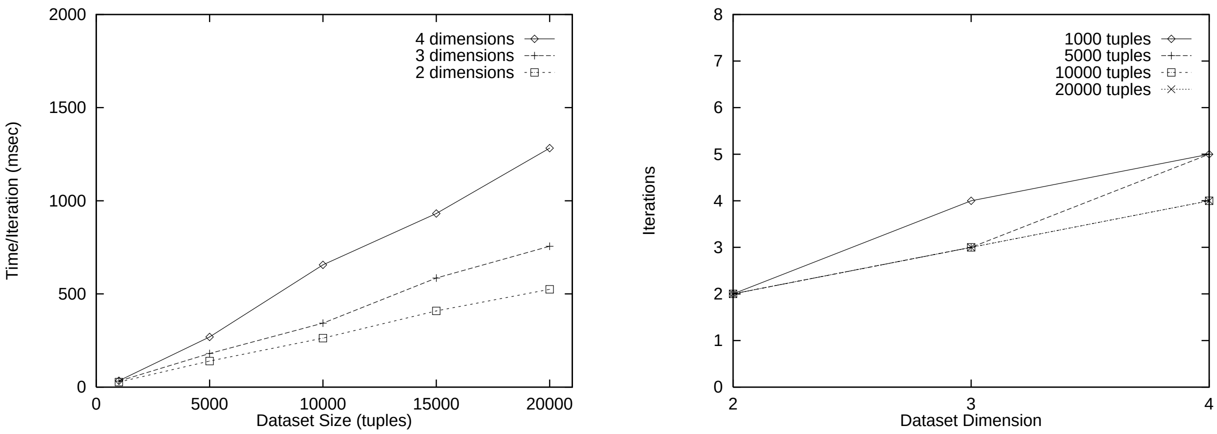
(a) Iterations vs Delta, synthetic datasets

(b) Iterations vs Delta, *census* datasets

Figure 4: The effect of varying the parameter δ on the number of iterations. All datasets are 4-dimensional.

datasets this choice will lead to good performance of the algorithm.

Figure 5(a) shows how the run-time of the algorithm changes when the dataset size increases. The graph demonstrates the outcome of experiments for datasets of different dimensionalities. As



(a) Time vs Dataset Size

(b) Iterations vs Dataset Dimensionality

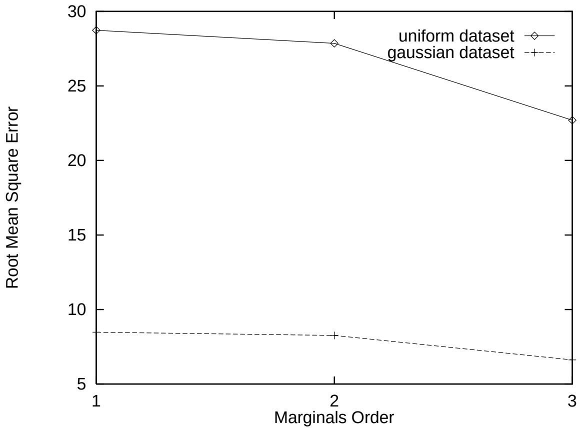
Figure 5: The effect of dataset size on the run-time of the algorithm (time per iteration), and of dataset dimensionality on the number of iterations.

expected, there exists a linear relationship between the size of the dataset and the run-time of the algorithm. Moreover, when dimensionality increases, the number of marginals that the algorithm uses increases as well. This explains the steeper curves of the graph for the higher dimensions.

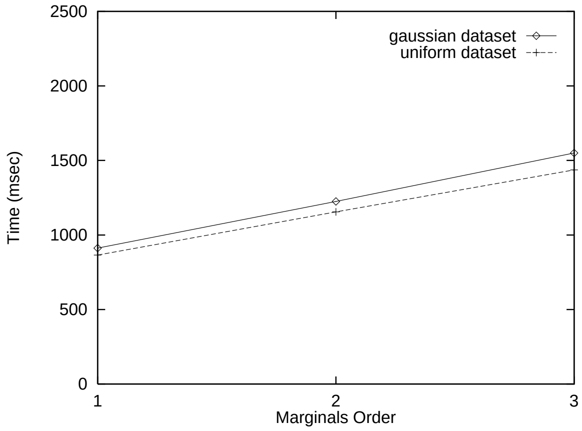
Other experiments that we performed indicate that there is no correlation between the dataset size and the number of iterations that the algorithm needs to perform in order to converge. Nevertheless, as Figure 5(b) depicts, the number of iterations increases with the dimensionality of the dataset. In the above experiments, we used the marginals of the highest possible order, and set δ to 1% of the median in order to exaggerate the differences.

The above results demonstrate that this approach can be effectively used by the analyst in real time, and in an interactive fashion, for middle-sized queries even when they do not fit in main memory.

The following graph is the experimental verification of Theorem 2. We used 4-dimensional datasets, and measured the error of the estimation when the algorithm operates with marginals of orders 1 to 3. Figure 6(a) is an illustration of the fact that when we use higher order marginals for the reconstruction of same dataset, the error is diminishing. It is interesting to note here, that the



(a) Error vs Marginal Order



(b) Time vs Marginal Order

Figure 6: The effect of the order of the marginals used on the error of reconstruction and the run-time of the algorithm.

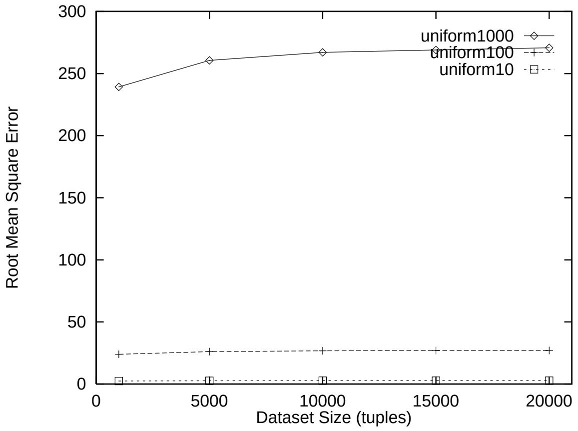
relative benefit of employing marginals of higher order is increasing. Thus, when we use marginals of order 3 the reduction of the error is more acute. The greater accuracy that we are gaining by using marginals of high order comes at the expense of time. The run-time of the algorithm increases with the order of the marginals (Figure 6(b)).

6.3 Evaluating the Accuracy of Reconstruction

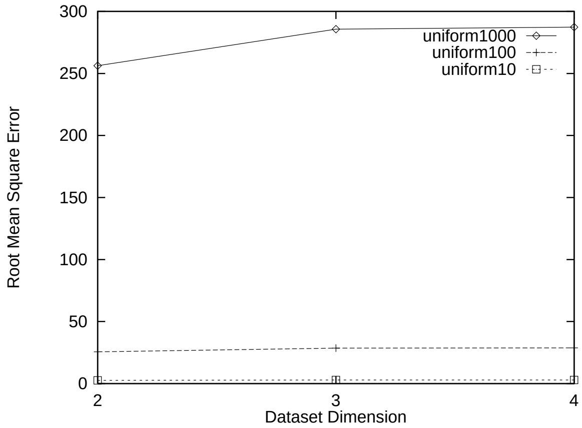
A number of experiments try to investigate the behavior of the algorithm when reconstructing an unknown dataset.

6.3.1 Synthetic Datasets

The graph in Figure 7(a) shows how the error changes when the dataset size increases, for the three uniform distributions. All the datasets have three dimensions. As is evident from figure 7(a), the error of reconstruction is related to the variance of the underlying dataset and it increases as the variance increases. By increasing the size of the dataset, the error increases (but not dramatically)



(a) Error vs Dataset Size

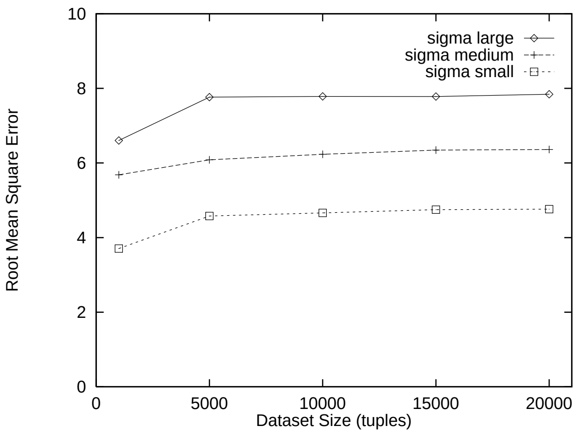


(b) Error vs Dataset Dimensionality

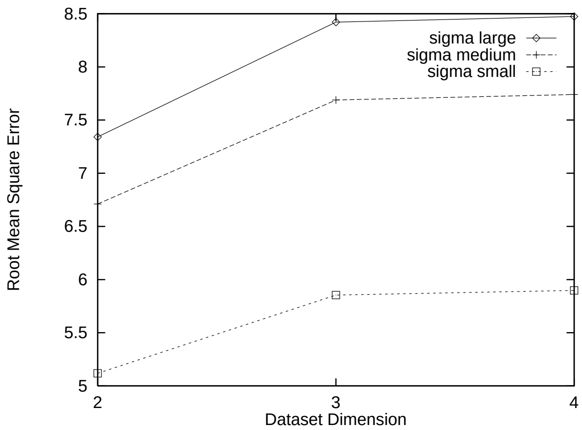
Figure 7: The effect of dataset size and dimensionality on error, for uniform datasets. Three uniform distributions with different mean values are represented in the graphs.

as more values are included in the computation of the error. The next graph, Figure 7(b), depicts how the dataset dimensionality affects the accuracy of the estimations. During this experiment we instructed the algorithm to use the marginals of the highest order common to all the datasets. It is evident that the reconstruction error increases with dimensionality; however, the increase seems correlated to the variance of the underlying dataset, since the increase of the error as the dimensionality increases is only nominal for datasets with similar variance.

Figure 8 depicts the way error changes with respect to dataset size and dimensionality, in the case of Gaussian datasets. The curves represent three different 3-dimensional datasets, that were



(a) Error vs Dataset Size



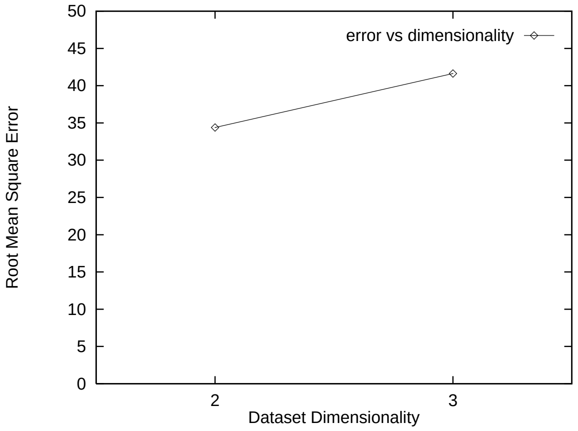
(b) Error vs Dataset Dimensionality

Figure 8: The effect of dataset size and dimensionality on error, for Gaussian datasets. Three Gaussian distributions with different sigma value are represented in the graphs.

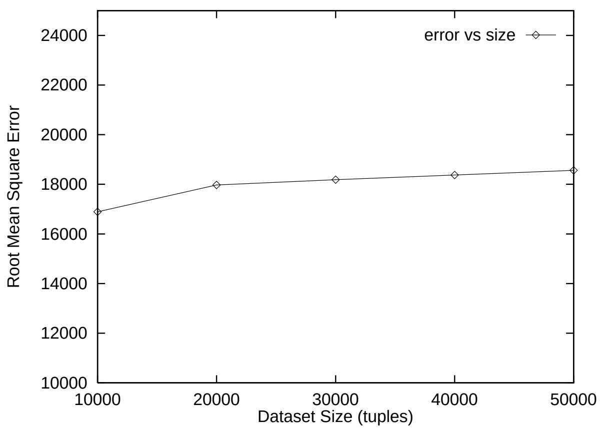
produced by sampling Gaussian distributions with the same mean μ, but different sigma σ, as described in the previous section. The results we get are similar to those in the experiment with uniform datasets.

6.3.2 Real Datasets

In this experiment, we used the real datasets to verify the trends that the reconstruction error follows as indicated by the previous experiments. Figure 9(a) depicts a graph of the error for increasing dimensionality, while Figure 9(b) demonstrates the behavior of the error as the dataset size grows. These graphs show a deterioration in the accuracy of reconstruction both when dimensionality



(a) *calls* Datasets.



(b) *census* Datasets.

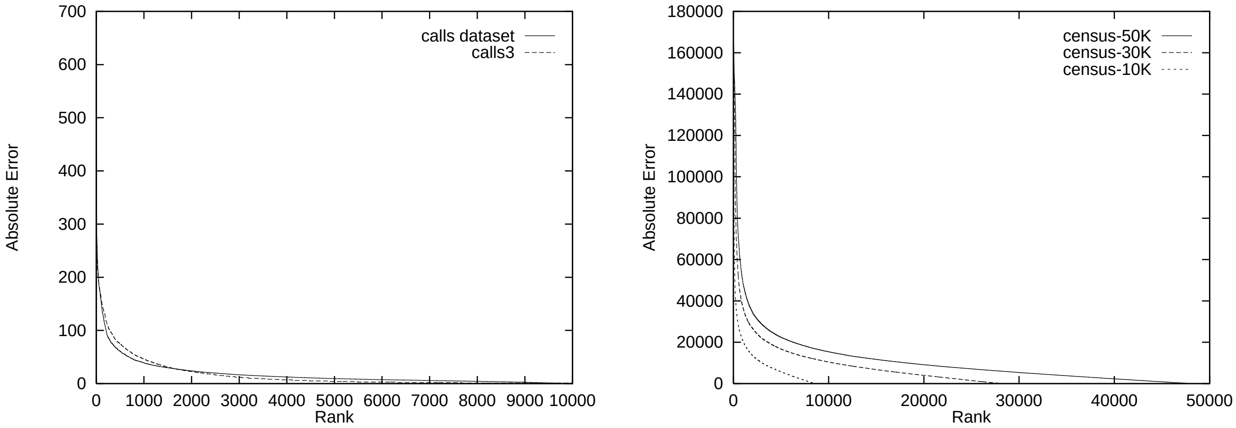
Figure 9: Error vs dimensionality and error vs dataset size for the real datasets.

goes up and when the size of the dataset increases, which is in accordance with the results of the experiments using the synthetic datasets.

6.4 Reconstruction with Error Guarantees

In the following experiments, we try to assess the benefit of providing error guarantees. The method we propose in order to achieve this goal (as discussed in Section 4.4) explicitly stores a small number of deviating values, which are subsequently used during the reconstruction phase to diminish the error.

In the first set of experiments we explore the distribution of the size of the estimation errors (i.e., the absolute error between the real and the estimated value for an individual cell). The graph in Figure 10(a) depicts the distributions for the datasets *calls* and *calls3*. Both curves indicate that the error sizes follow a highly skewed Zipf-ian distribution, with a very small fraction of the instances having large values. This fact indicates that the choice to store the largest estimation errors as extra information is very likely to pay off during reconstruction. Figure 10(b) shows the same graph for different sizes of the *census* dataset. The results show that for all the sizes the error

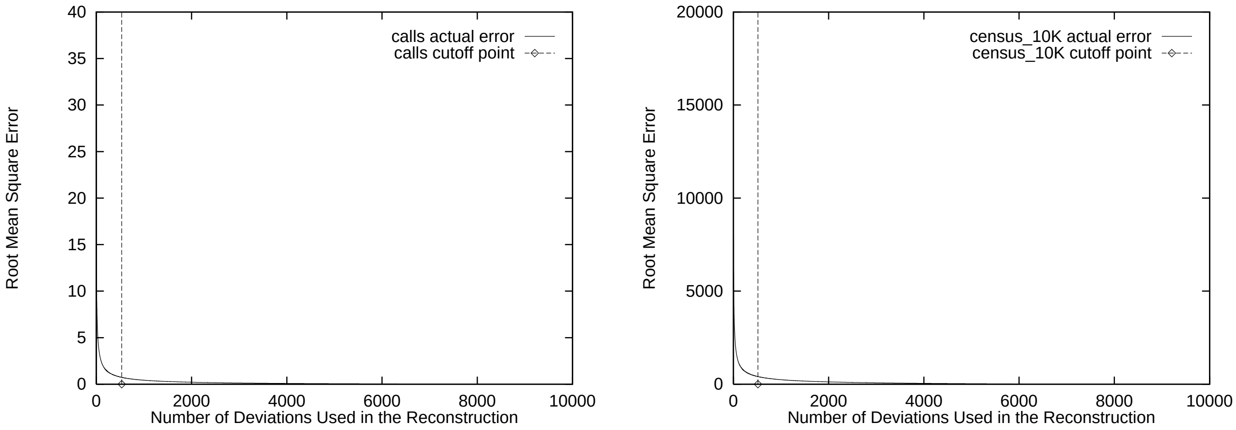


(a) *calls* Datasets. (b) *census* Datasets.

Figure 10: The distribution of the absolute error for the real datasets.

follows a skewed Zipf-ian distribution.

The next experiments evaluate the relative benefit of storing a number of deviating values per query in order to guarantee a specified reconstruction error level. Figure 11 shows the error of the reconstruction when the number of deviations that the algorithm is using increases from 0 to the size of the dataset. The user may choose between those two ends according to the application requirements for error guarantees, and the space restrictions on the number of extra values that the algorithm will use.



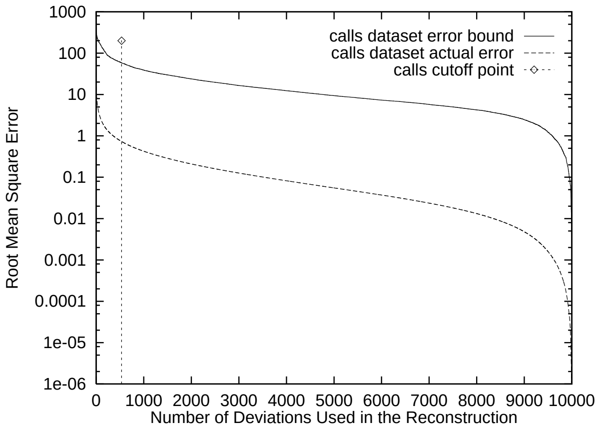
(a) *calls* Dataset. (b) *census_10K* Dataset.

Figure 11: The reconstruction error when a varying number of deviations is used by the algorithm.

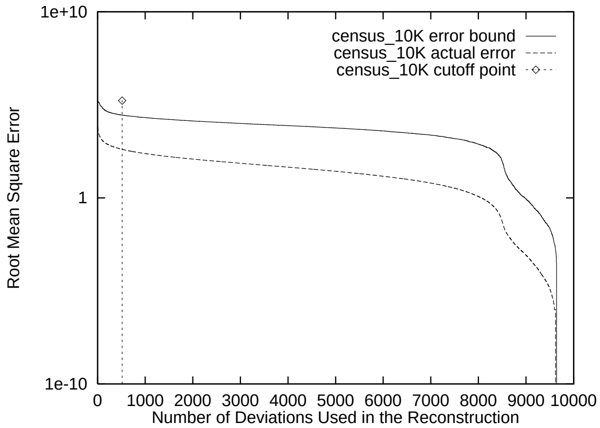
In every case, storing only a very small number of deviations is enough to dramatically decrease the error. In both cases of the real datasets we used, storing only the few largest deviations decreased the error by two orders of magnitude. As expected, at the other end of the spectrum, when the algorithm has knowledge of all the errors, it uses this information to achieve a perfect reconstruction.

It is interesting to note here the role that the *cutoff point* can play in this situation (see Section 5.2). In the graphs, the cutoff point, is marked with a vertical line, and it can be used to determine the point (and subsequently the number of values to materialize) at which the relative benefit of storing additional deviating values becomes negligible.

In Figure 12, we use a logarithmic scale for the y-axis to present the same graphs as before.



(a) *calls* Dataset.



(b) *census_10K* Dataset.

Figure 12: The actual reconstruction error and an upper bound when a varying number of deviations is used by the algorithm. The y-axis is plotted in logarithmic scale.

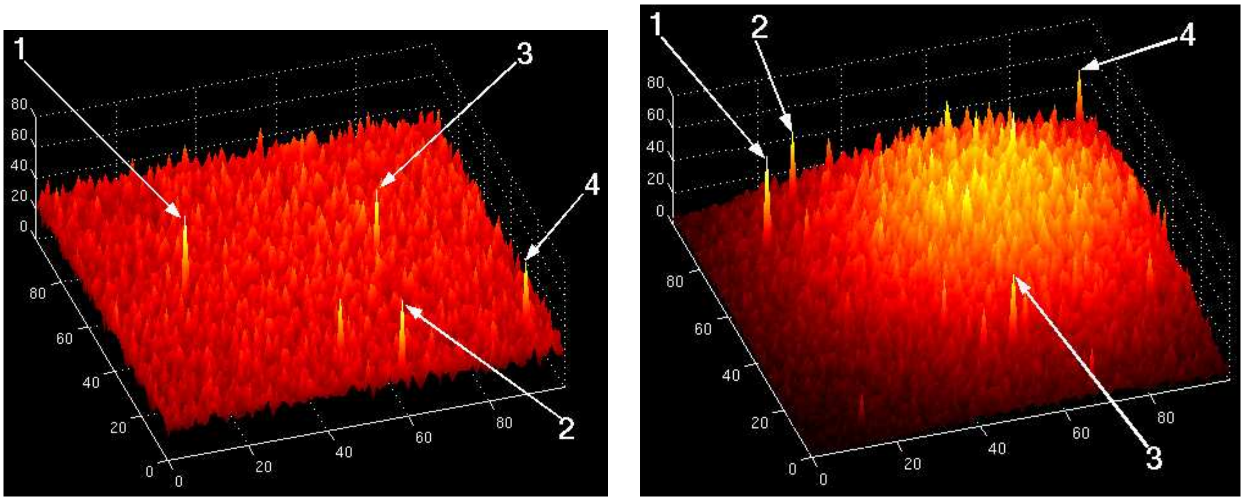
In addition to the actual reconstruction error, we plot a theoretical upper-bound for the error, following the discussion in Section 4.4. Even though this bound is not very tight, it may still be useful for certain kind of applications. We are currently working on ways to achieve a tighter bound. These graphs also enforce our argument about the *cutoff point*. It is clear that the *cutoff point* separates the initial region of dramatic decrease of the error from the plateau that follows. The added benefit of storing extra values from the latter region is quite small.

6.5 Mining Interesting Patterns

We evaluate the ability of the IPF algorithm to mine the underlying general structure of the data and report any deviations with the following experiments with synthetic and real datasets. Note, that the graphs we present involve datasets in two dimensions only, for illustration purposes.

6.5.1 Synthetic Datasets

We produced two datasets (namely *gauss_large_sigma* and *gauss_small_sigma*) drawn from Gaussian distributions, one with large (Figure 13(a)), and one with small sigma value (Figure 13(b)). We then added some uniform noise on top of the Gaussian distributions. The algorithm captured the



(a) *gauss_large_sigma* Dataset (b) *gauss_small_sigma* Dataset

Figure 13: Illustration of the original datasets following Gaussian distributions with uniform noise added on top. The four largest deviating values are marked in the graphs.

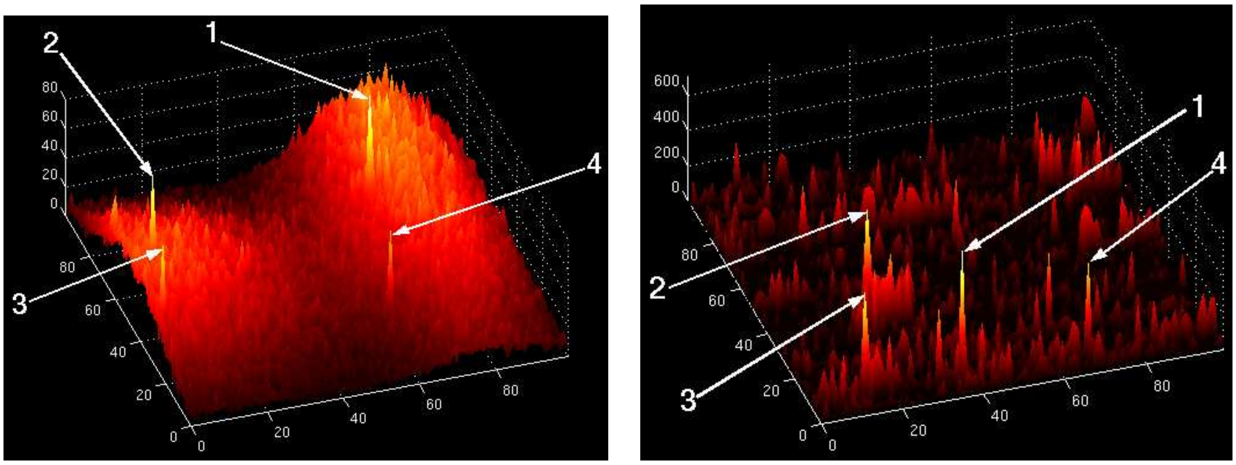
general trends of the data, and was able to report the values that deviate the most from the norm. The top-4 of these values are presented in Table 4. Manual inspection of the results reveals that

gauss_large_sigma		*gauss_small_sigma*		*gauss_combined*	
cell	diff.	cell	diff.	cell	diff.
(50,21)	47.34	(74,14)	56.80	(60,67)	62.78
(2,59)	47.27	(89,25)	44.01	(48,6)	58.33
(48,68)	42.15	(19,54)	42.74	(28,2)	52.26
(11,93)	41.24	(95,95)	36.98	(23,60)	51.54

Table 4: The top-4 deviations reported for each of the three synthetic datasets. The metric used is the *difference* of the real value from the estimated.

these are indeed the predominant deviations in the datasets.

The third synthetic dataset (*gauss_combined*) we tested is a combination of two multidimensional Gaussian distributions with different mean and sigma values, and some noise on top (Figure 14(a)). Once more, the algorithm correctly identified the base distributions, and singled out the most significant deviating values (reported in Table 4). Note that the algorithm does not merely identify global phenomena, e.g., reporting the maximum value along a dimension. Instead, it takes into account the local neighborhood in which a particular value appears, and reports any incongruities therein.



(a) *gauss_combined* Dataset (b) *calls* Dataset

Figure 14: Illustration of the two datasets where the four largest deviating values are marked. The dataset on the left is synthetic (combination of two different Gaussian distributions with uniform noise added on top), while the one on the right is the *calls* dataset from AT&T.

6.5.2 Real Datasets

In the following experiments, we instructed the algorithm to find the most deviating values in two of the real datasets, the *calls*, and the *census_50K* dataset.

Figure 14(b) depicts the *calls* dataset along with the top-4 deviating values, which are also listed in Table 5. All the marked values are instances of unusually high volume of calls. This information

calls	
cell	diff.
(7,37)	611.45
(38,24)	572.39
(7,13)	506.32
(7,68)	434.07

Table 5: The top-4 deviations reported for the *calls* datasets. The metric used is the *difference* of the real value from the estimated.

is important to the analyst since it indicates exceptional behavior which can either be fraudulent, or mark special cases in the dataset.

The outcome of the second experiment, with *census_50K*, cannot be graphically depicted, because the dataset is 5-dimensional. However, it is interesting to report some of the findings of the algorithm. The attributes of the dataset are age, command of English, number of children, level of education, and income. As expected, the above attributes are not independent. For example, the income tends to get larger with age, and when the level of education is higher. Nevertheless, there exist values that do not follow these patterns. Among the top deviations are a middle-aged person with high level of education who earns less than 20K, a person with a PhD degree who earns merely 3K, and a 24-year old who earns 200K. These are certainly results that deviate from the norm, and therefore are interesting.

Note that the algorithm is able to identify all the above results as interesting even though it has no domain knowledge, and it gets no user input.

7 Conclusion

In this paper we considered the problems of using just the aggregate information in order to provide approximate answers to queries and identify interesting values in multidimensional datasets. Each problem is of particular interest in the field of data analysis and approximate query answering respectively, especially since the volume of data stored in warehouses is huge. The techniques we discussed are based on the well recognized and widely applicable information theoretic principle of *maximum entropy*. We also proposed an extended framework that allows the user to choose specific error guarantees for the reconstruction process. Finally, we presented a detailed performance study using both real and synthetic data, highlighting the applicability and benefits of the approach, as well as the efficiency of the proposed algorithms.

Acknowledgements

We would like to thank Alberto Mendelzon for his comments that helped improve the content of this paper.

References

[AAR96] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. A Linear Method for Deviation Detection in Large Databases. In *International Conference on Knowledge Discovery and Data Mining*, pages 164–169, Portland, OR, USA, August 1996.

[AGPR99] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join Synopses for Approximate Query Answering. In *ACM SIGMOD*, pages 275–286, Philadelphia, PA, USA, June 1999.

[AM92] Soraya Abad-Mota. Approximate Query Processing with Summary Tables in Statistical Databases. In *EDBT*, pages 499–515, Vienna, Austria, March 1992.

[Ber82] D. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[BFH75] Yvonne M. M. Bishop, Stephen E. Fienberg, and Paul W. Holland. *Discrete Multivariate Analysis: Theory and Practice*. The MIT Press, 1975.

[BPP96] A. Berger, S. Pietra, and V. Pietra. A Maximum Entropy Approach to Natural Language Modelling. *Computational Linguistics, 22(1)*, May 1996.

[BS97] Daniel Barbará and Mark Sullivan. Quasi-Cubes: Exploiting Approximations in Multidimensional Databases. 26(3):12–17, 1997.

[BW00] Daniel Barbará and Xintao Wu. Using Loglinear Models to Compress Datacubes. In *Web-Age Information Management*, pages 311–322, Shanghai, China, June 2000.

[CR99] S. F. Chen and R. Rosenfeld. A Gaussian Prior For Smoothing Maximum Entropy Models. *Technical Report CMU-CS-99-108, Carnegie Mellon University*, February 1999.

[CT91] Thomas Cover and Joy Thomas. *Elements of Information Theory.* Wiley, 1991.

[DS40] W. E. Deming and F. F. Stephan. On a Least Square Adjustment of a Sampled Frequency Table When the Expected Marginal Totals Are Known. *Annals of Mathematical Statistics*, 11:427–444, 1940.

[FJS97] C. Faloutsos, H. V. Jagadish, and N. Sidiropoulos. Recovering Information from Summary Data. *VLDB, Athens, Greece*, pages 36–45, August 1997.

[GBLP96] Jim Gray, Adam Bosworth, Andrew Layman, and Hamid Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. In *International Conference on Data Engineering*, pages 152–159, New Orleans, LO, USA, March 1996.

[IP95] Y. Ioannidis and Viswanath Poosala. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. *ACM SIGMOD, San Jose, CA*, pages 233–244, June 1995.

[JKM+98] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. *VLDB*, pages 275–286, August 1998.

[JLS99] H. V. Jagadish, Laks V. S. Lakshmanan, and Divesh Srivastava. Snakes and Sandwiches: Optimal Clustering Strategies for a Data Warehouse. In *ACM SIGMOD*, pages 37–48, Philadephia, PA, USA, June 1999.

[JMS95] H. V. Jagadish, I. S. Mumick, and A. Silberschatz. View Maintenance Issues in the Chronicle Data Model. *ACM PODS*, pages 113–124, June 1995.

[KK92] J. N. Kapur and H. K. Kesavan. *Entropy Optimization Principles with Applications.* Academic Press Inc, 1992.

[KN98] Edwin M. Knorr and Raymond T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*, pages 392–403, New York, NY, USA, August 1998.

[KN99] Edwin M. Knorr and Raymond T. Ng. Finding Intensional Knowledge of Distance-Based Outliers. In *VLDB*, pages 211–222, Edinburgh, Scotland, September 1999.

[Kul68] Solomon Kullback. *Information Theory and Statistics.* John Wiley and Sons, 1968.

[Mal93] F. Malvestuto. A Universal Scheme Approach to Statistical Databases Containing Homogeneous Summary Tables. *ACM TODS, 18(4)*, pages 678–708, December 1993.

[MRL99] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. In *ACM SIGMOD*, pages 251–262, Philadephia, PA, USA, June 1999.

[PIHS96] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. *ACM SIGMOD, Montreal Canada*, pages 294–305, June 1996.

[SAM98] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven Exploration of OLAP Data Cubes. In *EDBT*, pages 168–182, Valencia, Spain, March 1998.

[SFB99] Jayavel Shanmugasundaram, Usama M. Fayyad, and P. S. Bradley. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. In *International Conference on Knowledge Discovery and Data Mining*, pages 223–232, San Diego, CA, USA, August 1999.

[ST96] A. Silberschatz and A. Tuzhilin. What Makes Patterns Interesting in Knowledge Discovery Systems. *IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No 6.*, pages 970–974, December 1996.

[Tho98] Joy Thomas. Personal Communication. 1998.

[VWI98] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. Data Cube Approximation and Histograms via Wavelets. In *ACM CIKM*, pages 96–104, Washington, DC, USA, 1998.