Using Datacube Aggregates for Approximate Querying and Deviation Detection

Themis Palpanas, Nick Koudas, *Member*, *IEEE Computer Society*, and Alberto Mendelzon

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, 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 use of the proposed framework that enables us to identify values that deviate from the underlying data distribution, suitable for data mining purposes. 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. Finally, we evaluate our techniques with a case study on a real data set, which illustrates the applicability of our approach.

Index Terms—Data warehouse, datacube, approximate query answering, deviation detection.

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, Fig. 1a showing a simplified three-dimensional OLAP cube, with three dimension attributes (location, jeans, and gender), and hierarchies defined per dimension. This cube stores the total volume of sales of different kinds of jeans for men and women across different cities. Users, for analysis purposes, commonly inquire about the values of aggregates, like the total sales of jeans in the state of New York. Such aggregates can be computed using the datacube operator [17], and queries of this kind can be efficiently supported.

The volume of data stored in OLAP tables is typically huge, on the order of many gigabytes, or even terabytes. In some cases, users store on disk only a subset of the data they own. They move the rest of the detailed data to tertiary storage systems, or even take them offline, while keeping only a small amount of aggregated data that are of interest. A common example is historical sales data, where only the data of the most recent years are stored online, and the rest are archived. In other cases, even if the data remain online, they are aggregated not only to support user queries faster, but also to save space.

Given the summarized form of data, users are often interested in inquiring about the data from which the summarized form was generated. In such cases, generating good estimates for the original data in response to queries is a pressing concern. Returning to our example of Fig. 1a, assume that for the women's jeans and for the state of New York (i.e., the upper front portion of the data cube), we only store the *aggregated* sales for each city and for each kind of jean as shown in Fig. 1b, and that we have deleted all the detailed values. The users might want to inquire about the number of redtab Levi's jeans sold in Queens, New York (a point query), or they might request the number of women's jeans sold in each city in the state of New York (a range query). In the latter case, the answer consists of *all the individual* values marked as "x" in Fig. 1b. We want to be able to answer these queries approximately using only the stored aggregate values. In this work, we present a technique that addresses this problem. Similar issues arise in transaction recording systems [21] as well as in statistical databases [4], [27].

Even if the original base data exist, the ability to reconstruct the original data from the summaries is of great value. In order to reconstruct the data, various assumptions have to be made about the statistical properties of the reduced data. Given the reconstructed and the original data at hand, we can test how valid 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 helpful in data mining. It may be used to detect correlations in the data and identify deviations, that is, values that do not conform to the underlying model. Such results are interesting 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. Our reconstruction technique is based on the well recognized and widely applicable information theoretic principle of *maximum entropy* [22]. We present algorithms for the efficient reconstruction of data from the aggregates. Moreover, using an information theoretic formalism, we identify and describe 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.

- *T. Palpanas is with IBM T.J. Watson Research Center, 19 Skyline Dr., Hawthorne, NY 10532. E-mail: themis@us.ibm.com.*
- *N. Koudas and A. Mendelzon are with the Department of Computer Science, Bahen Center for Information Technology, University of Toronto, 40 St. George Street, Rm BA5240, Toronto, ON Canada M5S 2E4. E-mail: :koudas, mendel}@cs.toronto.edu.*

Manuscript received 28 Oct. 2003; revised 27 Aug. 2004; accepted 18 May 2005; published online 19 Sept. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0215-1003.

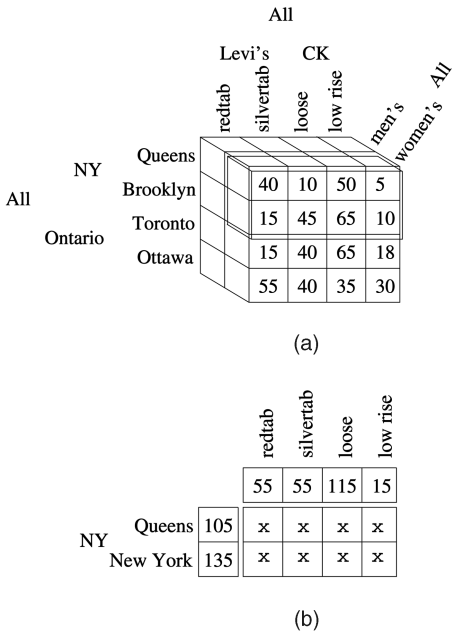


Fig. 1. Example of a three-dimensional data cube with sales data. (a) The entire data set. (b) Aggregated values for the selected portion of the data set (i.e., the sales of women's jeans in New York).

The contributions we make in this paper are as follows:

- We propose a method for reconstructing multidimensional values from aggregate data. In the OLAP environment, our technique uses the already computed aggregated values.
- We describe an extension to the above method in which we are able to provide quality guarantees (error bounds) for the reconstruction. Moreover, the quality of the reconstructed information can be controlled by the user, achieving any degree of desired accuracy at the cost of using more space.
- We present a method to identify and rank deviations in multidimensional data sets, that is, values that do not follow in general the underlying data distribution. These values are of particular interest to an analyst 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 experimental evaluation, using both synthetic and real data sets, as well as with a case study.

The outline of the rest of this paper is as follows: In Section 2, we present some background material necessary for the rest of the paper and Section 3 describes the reconstruction algorithm. In Section 4, we present experimental results evaluating the performance and the utility of the proposed algorithms. In Section 5, we demonstrate the applicability of the techniques discussed in this paper in a single case study. Section 6 reviews related work, and, finally, we present our conclusions in Section 7.

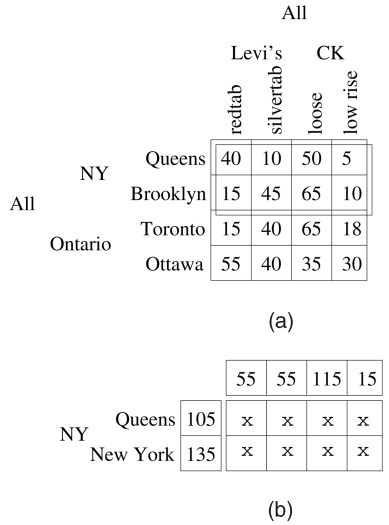


Fig. 2. Example dimension hierarchies on two-dimensional sales data. (a) The entire data set. (b) Aggregated values for the upper half of the data set.

2 Background

Consider a relation schema $R = (A_1, A_2, \ldots, A_n, Y)$ and r an instance of R. $A_1, \ldots, A_n$ are dimension and hierarchy attributes, and Y is a measure attribute. Attribute Y could represent volume of sales, dollar amount, number of calls, etc. Usually, each dimension of a datacube is associated with a set of hierarchically-related attribute values. A combination of attribute values from the leaves of the hierarchy specify a single data value in the data set. Any combination of attribute values in which at least one comes from a nonleaf level of its hierarchy specifies a (hyperrectangular) set of data values. For example, for the simplified OLAP table depicted in Fig. 2a the schema is R(STATE, CITY, BRAND, PRODUCT, SALES). Values of attribute CITY are hierarchically grouped into values of STATE, and values of attribute PRODUCT are hierarchically grouped into values of BRAND.

With a schema of n attributes, assume that h attributes define hierarchies on the remaining $d = n - h$ attributes. Following Jagadish et al. [20], we refer to the d-dimensional vector $(h_1, \ldots, h_d)$ as a *grid query*. When a grid query specifies a single data value in the data set, we refer to it as a *point* query. For example, viewing the table of Fig. 2a with its dimensions and hierarchies as a two-dimensional grid, the grid query *(NY, All)* is a range query referring to the entire upper half of the data set, while the grid query *(Queens, silvertab)* is a point query asking for the sales of silvertab Levi's in Queens. Grid queries are the most common queries in warehouse environments [20]. For the rest of this paper, we refer to grid queries simply as queries and, as we demonstrate later on, they can be easily represented using SQL.

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. We can derive from r all the distributions of P_r, where we have eliminated any of the attributes participating in P_r by summing over them. These distributions are called *marginal distributions* or simply *marginals*. More

formally, the marginal distribution of a set of random variables is the distribution obtained by summing the joint distribution over all values of the other variables.

Consider, for example, the following query:

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

where W is the sum of all Y values in the cube. The outcome of this query is one of the n marginal distributions of P_r of order $n-1$.

Reasoning similarly, one can draw an analogy between all the group-bys 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.

2.1 Maximum Entropy Distributions

For the following discussion, we will need the notion of *entropy*, which is defined with respect to a discrete random variable Z as follows: If variable Z takes on the values $(z_1, z_2, \ldots, z_n)$ with probabilities $(p_1, p_2, \ldots, p_n)$, then the entropy of variable Z, $H(Z)$, is

$$H(Z) = H(p_1, p_2, \ldots, p_n) = \sum_{i=1}^{n} p_i \log p_i.$$

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). The problem of *maximum entropy estimation of P* is defined as follows:

Problem 1: The Maximum Entropy Estimation Problem. *Given S, an arbitrary subset of the powerset of*

$$X = \{A_1, \ldots, A_n\},$$

find P such that it maximizes the entropy $H(P)$ of P, over all probability distributions that satisfy the following conditions:

- *every element in $P(X)$ has a nonnegative value,*
- $\sum P(X) = 1$, *and*
- $\forall i \in S, \sum_{j \in \{S-i\}} P(j) = P(i)$,

where $P(j)$ is a marginal distribution of P.

Note that j represents a set of attributes. Therefore, the dimensionality of $P(j)$ is equal to the cardinality of the set j.

The maximum entropy estimation of P is a model fitting technique. It has a unique solution [22] and it finds the model with the "least" information or fewest assumptions given the specified constraints, which are the marginal distributions in our case. The overriding principle in maximum entropy is that when nothing is known the distribution should be as uniform as possible, and the participating attributes independent. The constraints specify the regions where the estimated distribution should be minimally nonuniform, as well as the attribute correlations that should exist in the estimated distribution.

In the definition of the maximum entropy estimation problem (Problem 1), the only constraints that serve as input to the problem are the given marginals, to which the solution should conform. In the general case, these constraints can take other forms as well. For example, instead of using just the sums of the detailed values (i.e., the marginals), we could also use any of the higher order moments, such as variance and skew. In this work, we restrict our attention to the use of marginals only because these are readily available in a data cube.

The only exception to the above restriction is in order to take into consideration the values of particular elements of the sample space, for which the exact *real* values are known. Then, there is no need to produce estimates for these elements. We discuss cases where the above situation is applicable in Section 3.2.

2.2 Properties of Maximum Entropy

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

Theorem 1 [25]. *Let C be the set of all n-dimensional probability distributions that have the same marginals as those in M_i and assume that all distributions in C are equally probable to be the true distribution P. Then, the distribution $p = ME_i \in C$ minimizes the expected distance among p and $P' \in C$, where the distance function is defined as:*

$$D(p, P') = \sum_{X} p(X) \log \frac{p(X)}{P'(X)}. \quad (1)$$

The measure D is known in the literature as the relative entropy [14] and measures the similarity of two probability distributions. More precisely, D is a measure of the effort required to describe distribution P' based on the knowledge of distribution p. It has been shown that by minimizing $D(p, P')$ we also minimize the χ^2 test between p and P' [25].

Theorem 2 [25]. *Let $ME_i, 1 \leq i \leq n-1$ be the maximum entropy estimation of P using only marginals of order i. Then, the following inequality holds:*

$$D(ME_1, P) \geq D(ME_2, P) \ldots \geq D(ME_{n-1}, P). \quad (2)$$

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 Algorithmic Solution

Problem 1 is a constraint optimization problem that is not amenable to a general closed form solution. The standard technique for solving this maximization problem is the method of *Lagrange Multipliers* [6], which (in the multidimensional case) requires the solution of a rather compled, ad hoc (depending on the specified constraints) system of equations. This is not very appealing for automation.

We propose the use of an algorithmic approach. The technique is called *Iterative Proportional Fitting* (IPF) and was introduced by Deming and Stephan [15]. It is an iterative algorithm that converges to the maximum entropy solution. We can prove that IPF has the following properties [7]:

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

Objective: Compute the maximum entropy estimates $\hat{Y}_{A_1A_2...A_d}$ of a given d-dimensional dataset $Y_{A_1A_2...A_d}$.
Input: A set of k marginals $P_i(j), i = 1, \ldots, k$ and a stopping threshold δ.
Output: The maximum entropy estimates $\hat{Y}_{A_1A_2...A_d}$

let step t:=0;
initialize $\hat{Y}_{A_1A_2...A_d}$;
do {
  for i:=1 to k {
    $\forall l_1 \in [d_{1_s}, d_{1_e}], \ldots, l_d \in [d_{d_s}, d_{d_e}]$
    $\hat{Y}^{(t+1)}_{l_1l_2...l_d} = \hat{Y}^{(t)}_{l_1l_2...l_d} \frac{P_i(j)}{\hat{P}^{(t)}_i(j)}$
    for m:=1 to k {
      update $\hat{P}^{(t+1)}_m(j)$ based on $\hat{Y}^{(t+1)}_{l_1l_2...l_d}$;
    }
    t++;
  }
  let maxChange:=$\max\{\left|\hat{Y}^{(t)}_{A_1A_2...A_d} - \hat{Y}^{(t-k)}_{A_1A_2...A_d}\right|\}$,
    over all $l_1 \in [d_{1_s}, d_{1_e}], \ldots, l_d \in [d_{d_s}, d_{d_e}]$;
} while maxChange$\geq \delta$;

Fig. 3. The IPF algorithm.

2. A stopping rule may be used that ensures accuracy to any desired degree in the solution (i.e., in the asymptotic estimates).
3. The estimates depend only on the given set of marginals.
4. Convergence and its speed are not directly affected by the starting values.
5. In some cases, convergence is achieved after only a single iterative cycle.[1]
6. It does not require the values of the marginals (or, equivalently, of the data set) to be normalized in order to form a probability distribution.

3.1 Description of IPF

Let $r(A_1, A_2, \ldots, A_n, Y)$ be a relational instance. We specify a multidimensional region of interest $[d_{1_s}, d_{1_e}] \times \ldots \times [d_{d_s}, d_{d_e}]$, defined by a d-dimensional grid query $Q = (h_1, \ldots, h_d)$. Let S be the set containing all the marginals we are going to use for the reconstruction of Q, and k be the cardinality of S. To illustrate the above with an example, we use the data set of Fig. 2a. The highlighted box in the figure defines a range query; the two marginals based on which we will estimate this query are the row and column aggregates shown in Fig. 2b.

Let $P_i(j)$, $1 \leq i \leq k$, $j \in S$, be one of the k marginals of P. Note that by j we refer to a particular marginal. On each $P_i(j)$, the aggregation has been computed on all attributes not present in the marginal j.

We denote as $Y_{A_1A_2...A_n}$, the value of attribute Y for a specific combination of the A_i attributes (d of those are specified by the grid query and the remaining $n - d$ from the hierarchy). We denote as $\hat{Y}^{(t)}_{A_1A_2...A_n}$ the estimate of the value of $Y_{A_1A_2...A_n}$ during the tth 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, identically to 1. We refer to each element of the d-dimensional grid as a *cell*. In addition, it computes the k marginals in S for the initialized grid G. Let $\hat{P}^{(t)}_i(j)$, denote the marginals computed from G in the tth iteration of the algorithm. Denote $\hat{P}^{(0)}_i(j)$, the marginals after the initialization.

At each iteration, IPF loops over the k marginals $j \in S$, and over 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

1. A complete discussion of this topic may be found elsewhere [7].

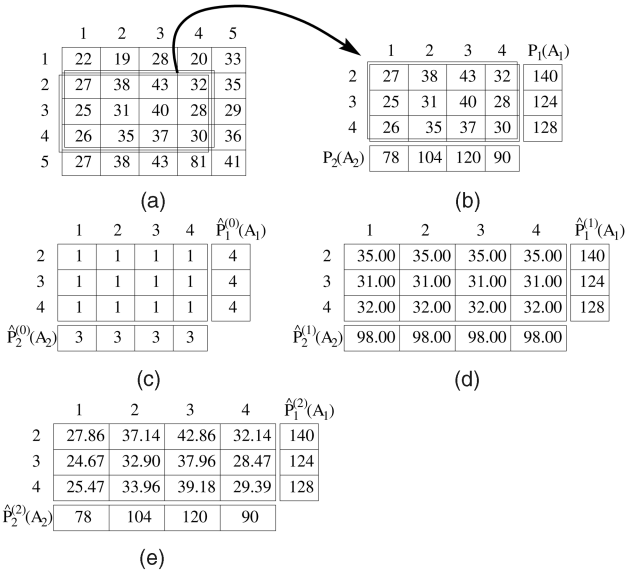


Fig. 4. An example of applying the IPF algorithm.

$$\hat{Y}^{(t+1)}_{l_1l_2...,l_d} = \hat{Y}^{(t)}_{l_1l_2...l_d} \frac{P_i(j)}{\hat{P}^{(t)}_i(j)}, \quad (3)$$

where $1 \leq i \leq k$.

This procedure is guaranteed to converge to the maximum entropy estimation of P given the specified collection of marginals. The estimates converge in a monotonic 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 in this chapter, we set δ to 10 percent 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 [29] and stored in the DBMS. In addition, as our experiments show, 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 Fig. 3.

Example 1. Assume that we have the data set depicted in Fig. 4a, and we want to focus on the highlighted subset. Fig. 4b shows the particular subset along with its marginals. The IPF algorithm starts by initializing the estimates $\hat{Y}^{(0)}_{l_1l_2} = 1$, $l_1 = 2, \ldots, 4; l_2 = 1, \ldots, 4$ (Fig. 4c). Then, in the subsequent steps, it fits the marginals one by one. First, it fits $P_1(A_1)$ using the formula $\hat{Y}^{(1)}_{l_1l_2} = \hat{Y}^{(0)}_{l_1l_2} \frac{P_1(A_1)}{\hat{P}^{(0)}_1(A_1)}$, $\forall l_1 \in [2, \ldots, 4], l_2 \in [1, \ldots, 4]$ and we obtain the table of Fig. 4d. Then, it fits marginal $P_2(A_2)$ using the formula $\hat{Y}^{(2)}_{l_1l_2} = \hat{Y}^{(1)}_{l_1l_2} \frac{P_2(A_2)}{\hat{P}^{(1)}_2(A_2)}$, $\forall l_1 \in [2, \ldots, 4], l_2 \in [1, \ldots, 4]$. The result is depicted in Fig. 4e, 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.

3.1.1 On the Selection of Marginals

An important issue is which marginals to choose in order to use them for the estimation process. Clearly, different sets of marginals will yield different estimates. This will, in turn, result in different estimation errors. Another, equally important, factor is the space required to store the marginals. If the gain in the approximation accuracy is negligible when using marginals of a high order, then we may as well restrict ourselves to a cheaper alternative that has almost the same benefit.

At this point, we know, based on Theorem 2, that when we use marginals of the same order for the estimation procedure, then the higher the order of the marginals is, the more accurate the estimation becomes. Nevertheless, what we would really need is a measure of the relative benefit of using different sets of marginals. Note that we should be allowed to choose marginals from different orders as well. This measure would serve as a means of comparison among the available choices of marginals, by taking into account both the space required to store each alternative, and the estimation accuracy that it achieves. Ideally, we would like to be able to compute this measure based only on the marginals.

Unfortunately, no measure that can effectively quantify the relative merit of using alternative sets of marginals is known. In the absence of such a measure, for the rest of this paper, we restrict our attention to marginals of the same order.

3.1.2 Algorithmic Complexity

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 hold the grid G then the algorithm only needs to read from disk the marginals. 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 k passes over G, where k is the number of 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. Therefore, the total cost is $[2kt]$ passes, where t is the number of iterations. In our implementation, we incorporate the update of the estimated marginals with the computation of the base values, reducing the cost of the algorithm to $[kt]$ passes over G. The results that we report in the experimental section illustrate the worst-case scenario, where the data set does not fit in main memory.

3.2 Quality 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 bounds for the reconstruction of individual values is imperative. For this procedure, 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 queries of interest. One approach to provide error bounds 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 incur 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[2] $N \cdot \max_{1 \leq i \leq N}\{d_i\}$. 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 applications may require tighter quality 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 estimation errors have been explicitly stored, the reconstruction algorithm uses these 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 bound per query should be specified by the user, we can choose k (the number of values to store) such that the overall error of reconstruction satisfies the error bound. As will become evident from the experiments, only a minor percentage of cells exhibit high errors and, thus, can be efficiently stored, with only a small storage overhead.

3.3 Mining Interesting Patterns

When 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. Such values can potentially be of great interest to the analyst, and we term them *deviations* because they deviate from the estimation model.

The basis for identifying 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, this metric may not always produce high quality results. An example of this case would be a data set with values drawn from a uniform distribution. Then, most of the values in this data set 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 data set $s = \frac{|\hat{Y}_i - Y_i|}{\sigma}$, 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 differentiate between the normal perturbations in the data set and the large deviations. The above technique splits the sorted set of deviations into two regions: it assigns the statistically large deviations to the

2. We assume that the error metric is the Root Mean Square Error. Similar arguments hold if we choose other error metrics as well.

TABLE 1
The Statistical Properties (Min, Max, Mean, Standard Deviation, and Skew) for the Synthetic Data Sets Used in the Experiments

dataset	*min*	*max*	*mean*	*std.dev.*	*skew*
uniform1000	0.0	1000.0	501.14	287.84	0.01
gauss_small	38.0	115.0	73.46	10.26	0.11
gauss_medium	4.0	100.0	42.90	16.24	0.40
gauss_large	0.0	104.0	15.60	18.34	1.65
gauss_combined	0.0	106.68	24.66	13.57	0.53

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 $s = 2$, which will prune 95 percent of the approximation errors as trivial, leaving only the largest 5 percent 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 deviations determined by the algorithm, for both synthetic and real data sets.

4 Experimental Evaluation

In order to test the IPF algorithm, we used both synthetic and real data sets. The synthetic data sets are produced by sampling uniform and Gaussian data distributions.

Uniform. We produced data sets of dimensionalities 2, 3, and 4. For each of the above data sets of different dimensionality, the values for the measure attribute were drawn uniformly from the range $[0, 10]$ (*uniform10*), $[0, 100]$ (*uniform100*), and $[0, 1000]$ (*uniform1000*). The size of the data sets varied from 1,000 to 20,000 tuples. The statistical properties of those data sets are reported in Table 1.

Gaussian. We produced data sets of dimensionalities 2, 3, and 4. The values for the measure attribute were sampled from independent Gaussian distributions. For σ we chose three different values that altered the distribution of the values in the data space. In the experiments, we refer to these data sets as *gauss_small*, *gauss_medium*, and *gauss_large*. Once more, the size of the data sets varied from 1,000 to 20,000 tuples. We also experimented with a Gaussian data set which had additional random noise coming from a uniform distribution. This data set, *gauss_combined*, was derived from a mixture of two multidimensional Gaussian distributions. The statistical properties of the data sets are reported in Table 1.

Real. The first two real data sets, *calls* and *calls3*, are derived from AT&T proprietary data. Their measure attribute represents the number of telephone calls in certain regions of North America over time. They are 2 and three-dimensional, and their size is 10,000 tuples. The dimension attributes are time, location of the call, and customer type.

Finally, we used *census*, a 4-dimensional data set from the US Census Bureau, containing information about the age, education, command of English, and number of children of individuals. The measure attribute records the income of the individuals. From this data set, we extracted instances of 10,000-50,000 tuples by uniform random sampling.

Table 2 summarizes the statistical properties of the real data sets we used in our experiments.

TABLE 2
The Statistical Properties (Min, Max, Mean, Standard Deviation, and Skew) for All the Real Data Sets

dataset	*min*	*max*	*mean*	*std.dev.*	*skew*
calls	1.0	729.00	18.01	37.79	5.37
calls3	0.0	729.00	9.01	9.32	22.77
census_10K	1000.00	196623.00	24735.73	23449.87	3.67

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

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

where Y_i represents the original values in the data set, $\hat{Y}_i$ the corresponding estimated values, and N is the total number of values in our data set. In the following experiments, we estimate the values of each data set and measure the error of the estimation for all N values of the data set.

4.1 Exploring the Properties of the Algorithm

4.1.1 Effect of Data Set Size and Dimensionality

Fig. 5a shows how the running time of the algorithm changes when the data set size increases. As expected, there exists a linear relationship between the size of the data set and the running time of the algorithm. Moreover, when dimensionality increases, the number of marginals that the algorithm uses increases as well. This explains the steeper slopes for the curves in Fig. 5a for the higher dimensions. As Fig. 5b depicts, the number of iterations increases with the dimensionality of the data set. For these experiments, we used the marginals of the highest possible order in order to exaggerate the differences. We also performed some experiments to test the effect of data set size on the convergence of the algorithm. These experiments indicate that there is no correlation between the data set size and the number of iterations that the algorithm needs to perform in order to converge.

These results on the scalability of the algorithm demonstrate that this approach can be effectively used by the analyst in real time, and in an interactive fashion, even for queries that do not fit in main memory.

The graph in Fig. 6a illustrates how the error changes when the data set size increases, for the three Gaussian distributions (the results for the uniform data sets are similar, and are omitted for brevity). All the data sets have three dimensions. The graph shows that the error of the reconstruction is related to the variance of the underlying data set and it increases as the variance increases, while data set size has a neglizible effect. The next graph, Fig. 6b, depicts how the data set dimensionality affects the accuracy of the estimations. It is evident that the reconstruction error increases with dimensionality; however, the increase seems correlated to the variance of the underlying data set since the increase of the error as the dimensionality increases is small.

4.1.2 Effect of Order of Marginals

Fig. 7 depicts the experimental verification of Theorem 2. We used four-dimensional data sets, and measured the error of the estimation when the algorithm operates with marginals of orders 1 to 3. Fig. 7a is an illustration of the fact that when we use higher order marginals for the recon-

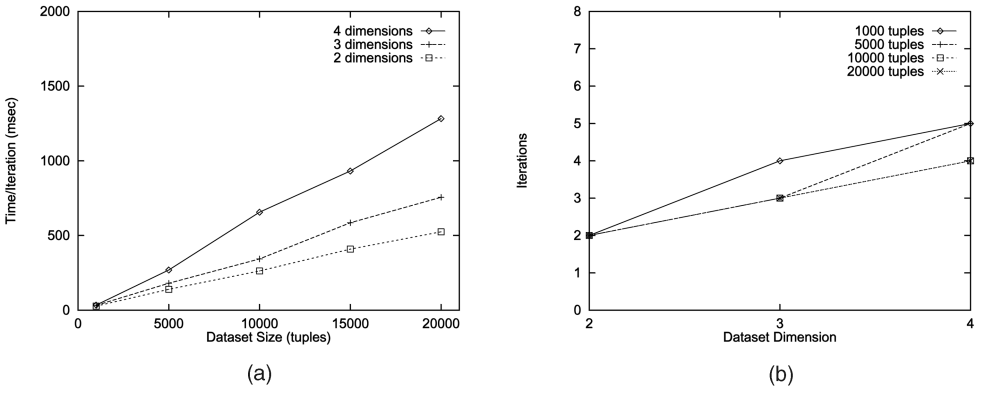


Fig. 5. The effect of data set size on the running time of the algorithm (time per iteration), and of data set dimensionality on the number of iterations. For these experiments we used the uniform data sets. (a) Time versus data set size. (b) Iterations versus data set dimensionality.

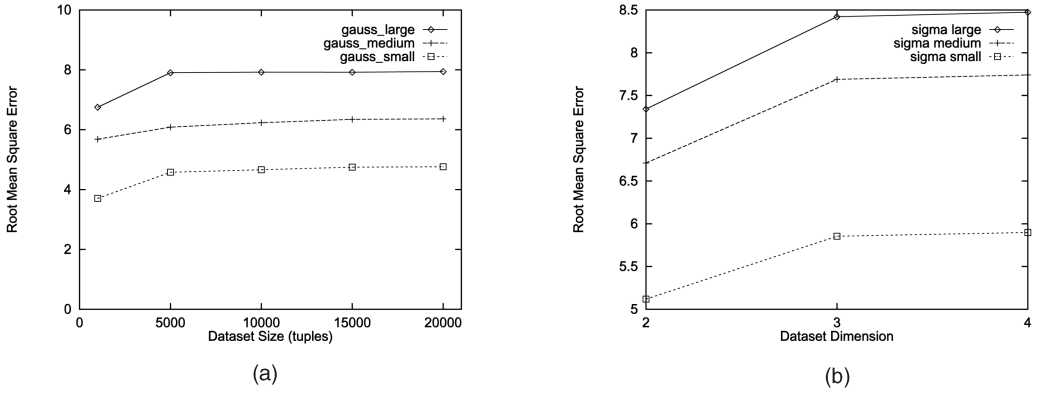


Fig. 6. The effect of data set size and dimensionality on error, for Gaussian data sets. Three Gaussian distributions with different sigma value are represented in the graphs. (a) Error versus data set size, three-dimensional data sets and (b) error versus data set dimensionality, 10,000 tuples.

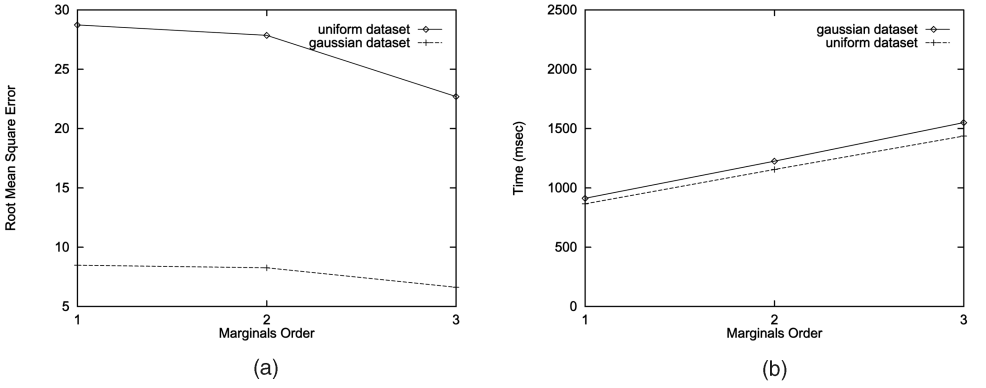


Fig. 7. The effect of the order of the marginals used on the error of reconstruction and the running time of the algorithm. The data sets used in these experiments are *uniform100* and *gauss_small*. (a) Error versus marginal order. (b) Time versus marginal order.

struction of same data set, the error is diminishing. The large difference in the errors shown in the graph of Fig. 7a is explained by the fact that the uniform data set has a much larger variance than the Gaussian one (see Table 1). It is interesting to note here that, for the data sets we used, the relative benefit of employing marginals of higher order is increasing. The reduction in error is larger in moving from order 2 to order 3 marginals, than it is for moving from order 1 to order 2 marginals. The greater accuracy that we are gaining by using marginals of high order comes at the expense of time (and, of course, space). The runtime of the algorithm increases with the order of the marginals (Fig. 7b).

4.2 Reconstruction with Quality Guarantees

In the following experiments, we assess the benefit of providing error bounds. 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 Fig. 8a depicts the distributions for the data sets *calls* and *calls3* after sorting into decreasing size. Both curves indicate that the error sizes follow a skewed distribution. This fact indicates that the choice to store the largest estimation errors as extra information is likely to pay off during reconstruction. Fig. 8b shows the same graph for different sizes of the *census* data set.

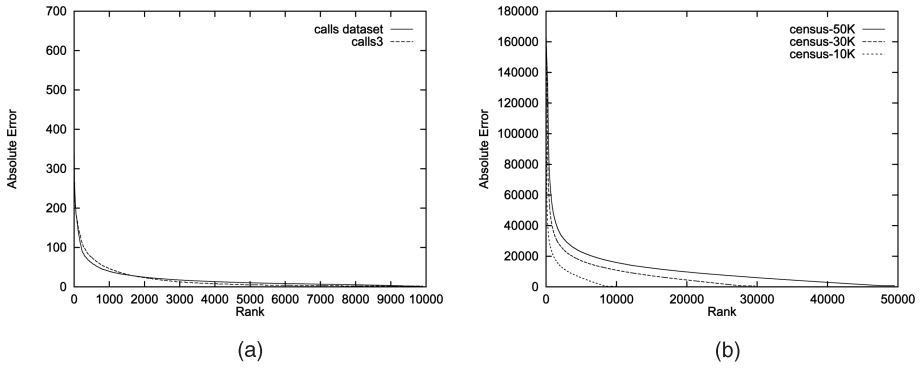


Fig. 8. The distribution of the absolute error for the real data sets. (a) *calls* data sets. (b) *census* data sets.

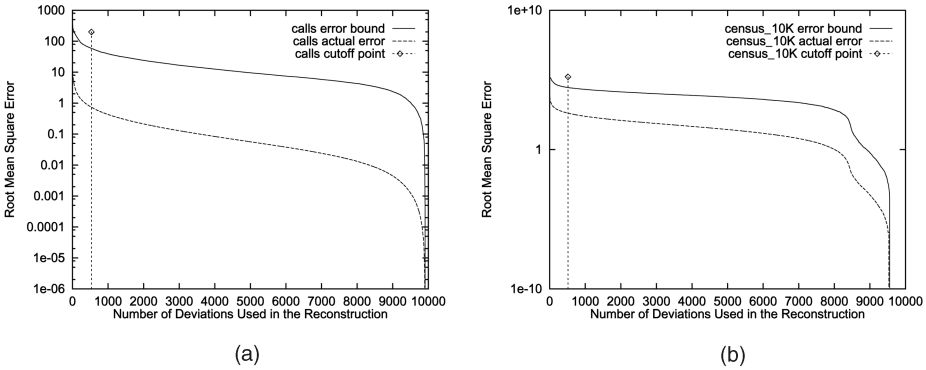


Fig. 9. The actual reconstruction error and an upper bound when a varying number of deviations is used by the algorithm. (a) *calls* data set. (b) *census* data set.

The next experiments evaluate the relative benefit of storing a number of deviating values per query in order to guarantee a specified reconstruction error bound. Fig. 9 shows the error of the reconstruction as the number of deviations that are stored increases from 0 to the size of the data set. In every case, storing only a very small number of deviations is enough to dramatically decrease the error. For both real data sets we used, storing only the few largest deviations decreased the error by two orders of magnitude.

Note the role that the cutoff point (see Section 3.3) can play in this situation. 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. It is clear that the cutoff point separates the initial region of dramatic decrease of the error from the plateau that follows.

In addition to the actual reconstruction error, we plot a theoretical upper-bound for the error, following the discussion in Section 3.2. Even though this bound is not tight, it may still be useful for certain kinds of applications.

In the following experiments, we evaluate the trade off between the accuracy of the estimation and the space needed to achieve this accuracy. Fig. 10 shows the reduction in the estimation error when we use marginals of successively higher order, and the increase in the space needed by these marginals. When we use marginals of higher order, the accuracy of the estimation increases. However, this increase in the accuracy comes at the expense of space since the space needed by the marginals of higher order increases as well. If we compare the graphs illustrated in Figs. 10a and 10b, we observe that in order to improve the estimation accuracy by a small amount, we have to use disproportionally more space. That is, there is small added benefit for the extra space needed by the marginals of higher order.

In the above experiments, we reduced the estimation error by employing marginals of higher order. Another way of reducing the estimation error is to explicitly store a number of the largest deviating values. For the following experiments, we measure the reduction in the estimation error when we use the marginals of some order i and a number of deviations. We ensure that the total space required by both the marginals of order i and the stored deviations, equals the space required by the marginals of order $i+1$. In this way, we can compare the benefit of explicitly storing some deviating values, against using marginals of higher order, for some fixed space. Fig. 10c depicts the outcome of these experiments. We observe that in this case, the reduction of the estimation error is considerable, and the reconstruction of the real values of the data sets is almost perfect.

Evidently, for the same amount of space, it is much more beneficial to store the marginals of some order i along with a number of the largest deviations, than to store the marginals of order $i+1$. Nevertheless, it is not always the case that we can explicitly store the deviating values. For example, consider the scenario of an online system, where only the marginals of some measures of interest are materialized, and the detailed values are not stored. Then, we will have to use just the marginals for the estimation process.

4.3 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

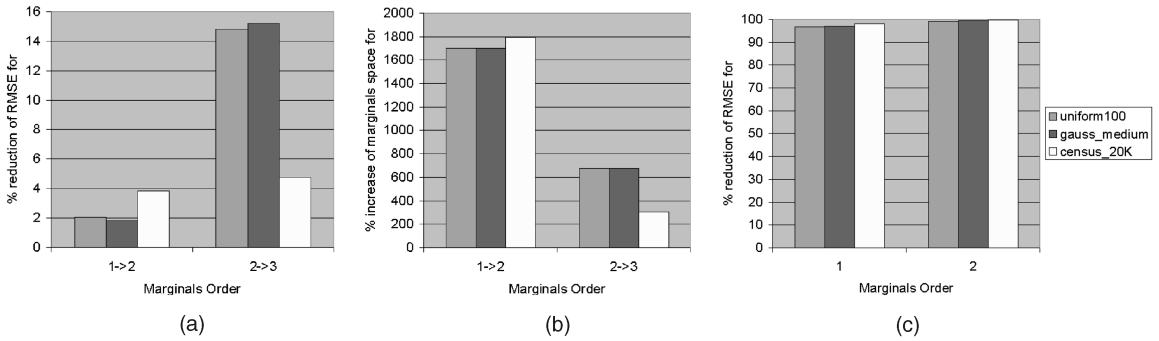


Fig. 10. The percentage of the (a) error reduction and increase in (b) space when we use the marginals of order $i+1$ instead of the marginals of order i, in the estimation process. (a) Error reduction (only marginals), (b) space increase (only marginals), and (c) error reduction (marginals and deviations).

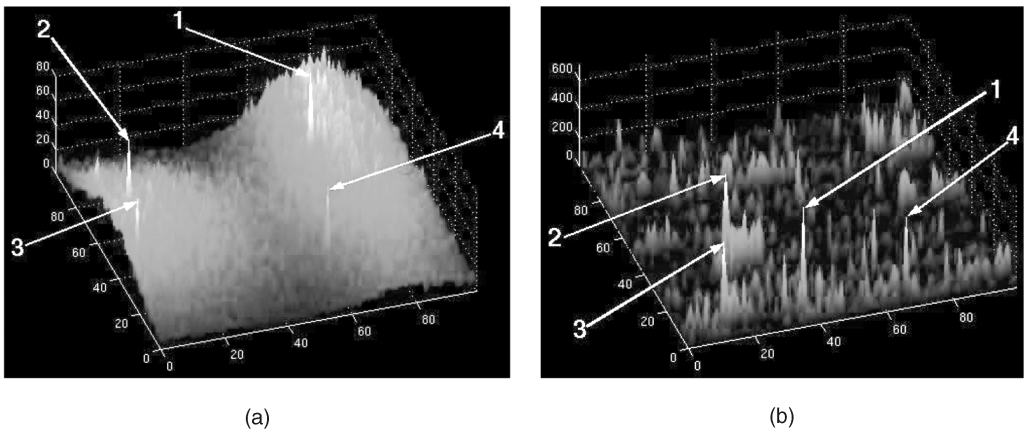


Fig. 11. Illustration of the largest deviations found in two of the data sets. (a) *gauss_combined* data set. (b) *calls* data set.

and real data sets. Note that the graphs we present involve data sets in two dimensions only, for illustration purposes.

4.3.1 Synthetic Data Set

The synthetic data set (*gauss_combined*) we tested is a combination of two multidimensional Gaussian distributions with different mean and sigma values, and some uniform noise added on top. It is depicted in Fig. 11a. The algorithm correctly singled out the most significant deviating values. 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. For example, consider the deviation number 4 reported by the algorithm, in Fig. 11a. This value is not the maximum value of the data set (its rank in the data set is 116th). Nevertheless, it is identified by the algorithm as a deviant, because it differs significantly from all the other values in its neighborhood.

4.3.2 Real Data Sets

In the following experiments, we used the algorithm to find the most deviating values in two of the real data sets, the *calls*, and the *census_10K* data set.

Fig. 11b depicts the *calls* data set along with the top-4 deviating values. All the marked values are instances of unusually high volume of calls. This information is important to the analyst since it indicates exceptional behavior which can either be fraudulent, or signify special cases in the data set.

The outcome of the second experiment, with *census_10K*, cannot be graphically depicted because the data set is four-dimensional. However, it is interesting to report some of the findings of the algorithm. The dimensional attributes of the data set are age, command of English, number of children, and level of education, while income is the measure attribute. As expected, the above attributes are not independent. For example, the income tends to increase with age and with the level of education. 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.

5 Case Study

In this section, we demonstrate how the techniques presented in this work can be applied in a single case study over a real data set. Namely, we start by considering a workload of queries. These queries specify the regions of the entire data set that need to be estimated with the maximum accuracy. To this end, we materialize the highest order marginals that correspond to each one of these regions, according to the discussion in Section 3. Then, in order to be able to answer any other query that asks for values in the data set that fall outside the selected regions,

we will also materialize the lowest order marginals (i.e., marginals of order one) for the entire data set. Evidently, since we are targeting the set of selected regions in the data set, we expect to achieve better accuracy in estimating any query that asks for values inside those regions. Nevertheless, by storing the marginals of order one for the entire data set as well, we are able to answer approximately any query, regardless of which values in the data set it involves.

5.1 Description of Experiments

For all the experiments in the case study described in this section, we used the *census_50K* data set, whose size is $50,000$ tuples, and we use synthetically generated query workloads in order to specify selected regions in the data set. All the regions defined by the queries in the workload are four-dimensional hyperrectangles. Note that in all cases, the number of tuples of the data set that belong to more than one selected region is negligible compared to the total number of tuples in the selected regions. Therefore, we can safely assume that this parameter of the problem does not affect the outcome of our experiments significantly.

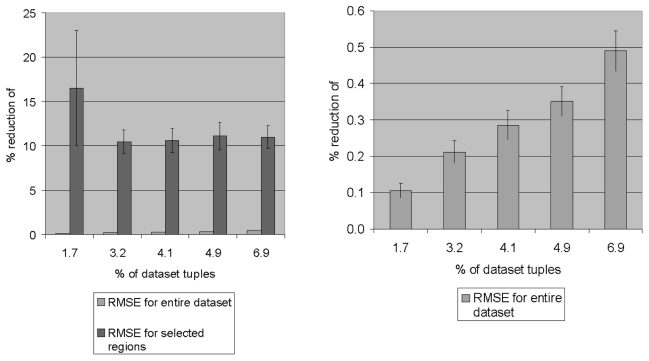
The error metric that we report in the experiments is the *Root Mean Square Error (RMSE)*. The error is computed as follows: We wish to estimate all the values in the data set. For each value, we check whether it is contained in any of the selected regions, for which we have stored the marginals of the highest order. If it is contained, then we estimate the value based on these marginals. Otherwise, we estimate the value based on the marginals of the lowest order, which we have stored for the entire data set. Note that we estimate the values and measure the error of the estimation for all N values of the data set. Each of the experiments was run 30 times and in the results, we report the mean values and confidence intervals for these runs. For the error metric, we also show in the graphs the corresponding confidence intervals with level of 95 percent confidence.

5.2 Queries on Detailed Values

In Fig. 12a, we depict the reduction in the error of the estimation for different sets of selected regions of increasing cardinality. The basis for computing the reduction of the error are the estimates provided by the marginals of order 1, which correspond to the independence assumption. The graph shows the observed error reduction as a percentage, for both the error over the entire data set, and the error over the values contained in the selected regions.

Evidently, the benefit of storing the marginals for the selected regions is entirely absorbed by the values contained in these selected regions. The rest of the values in the data set do not benefit. Thus, when the reduction in the estimation error of the values in the selected regions is spread over the entire data set, as depicted by the light colored bars in the graph of Fig. 12a, it becomes negligible.

We should note here, that the significantly higher error reduction for the first set of selected regions (leftmost dark bar in the graph) is caused by a small number of queries, whose values are approximated exceptionally well. The reduction of the error for the rest of the cases is slightly more than 10 percent. The fact that this error reduction remains steady across the different sets of selected regions, even though the space dedicated to the marginals is increasing, is not surprising. The additional marginals are related to a new, different, set of values in the data set. Therefore, we should not expect them to reduce any further the estimation errors of the old set of values.



(a) (b)

Fig. 12. The percentage of the error reduction when estimating all the values in the data set, and when estimating only the values contained in the selected regions. (a) Overall and selected regions. (b) Overall only .

Nevertheless, as we show in Fig. 12b, the overall error is steadily diminishing when we materialize more marginals. This graph illustrates the reduction of the error for the entire data set as a function of the tuples covered by the selected regions. This confirms the intuition that when we use more space to store marginals in order for the selected regions to cover more tuples of the data set, we should expect a reduction of the overall estimation error.

In the following experiments, we compare the error reduction in estimating the values contained in the selected regions (Fig. 13a), and all the values in the data set (Fig. 13b), when we employ the marginals of order 1 and order 3 of the selected regions. The marginals of order 1 are the lowest order marginals, and in the graphs are depicted with the light colored bars, while the marginals of order 3 are the highest order, and are depicted with the dark colored bars. Fig. 13a shows that the reduction in the estimation error when we use the marginals of order 3 is up to three times larger than when using the marginals of order 1, for values in the selected regions only. When we consider all the values in the data set, then the benefit is not as pronounced (see Fig. 13b). This is because only a small subset of the values can benefit from the use of the higher order marginals. Note also that in this case, the error reduction is approximately twice as large when we use the marginals of higher order. This improvement in the estimation accuracy comes at the expense of space, since the marginals of order 3 require about 50 times more space than those of order 1. Clearly, we have to take into account this tradeoff when deciding whether the added accuracy, that the marginals of higher order offer, is needed.

5.3 Aggregate Queries

In our discussion so far, we have only considered the case where we are interested in estimating all the detailed values specified by a query. The error metric has been calculated accordingly by measuring the difference between the estimate and the real value for each one of the detailed values individually.

In the next set of experiments, we consider another important class of queries, namely, the aggregate queries. For this class of queries, we are interested in a *single* value only: the *aggregate* of all the values specified by the query. Then, the error is calculated on the estimate of the aggregate and the real aggregate value.

For the following experiments, we produced a query workload of 5,000 queries, by employing the same method

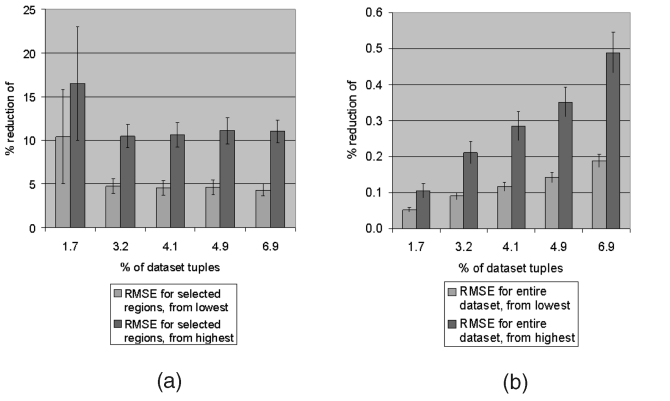


Fig. 13. The percentage of the error reduction when estimating only the values contained in the selected regions, and all the values in the data set, by using the marginals of order 1 and order 3. (a) Selected regions. (b) Overall.

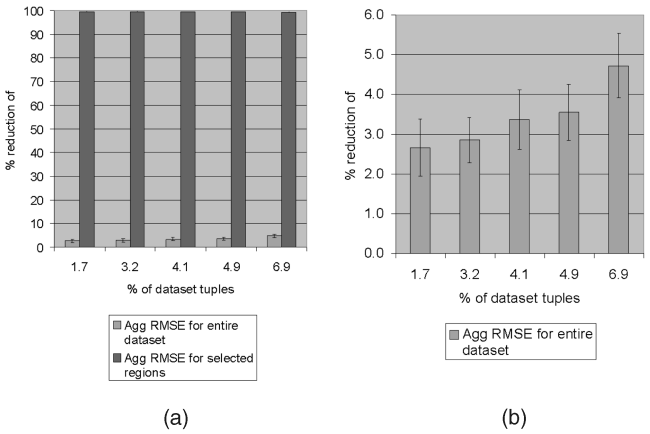


Fig. 14. The percentage of the error reduction when estimating the aggregate value of subsets of the entire data set and subsets of the selected regions only. (a) From entire data set and selected regions and (b) from entire data set only.

we used to produce the selected regions, as described earlier. We used the same sets of selected regions as before, and repeated each experiment 30 times. The results on the errors we report are over the 5,000 aggregate queries, and then averaged over the 30 repetitions of each experiment. The error metric we use is once again the RMSE.

Fig. 14a depicts the reduction in the error when estimating aggregate queries and when we use the marginals of order 3 for the selected regions. We measure the error reduction for aggregate queries over the entire data set (light colored bars), and over the selected regions only (dark colored bars). In the latter case, we simply consider the portion of each query that overlaps with some selected region, and we compute the aggregate value of this portion. (In each of the experiments we ran, there were approximately 1,000 queries overlapping with the selected regions.) The graph shows that the estimates produced for the aggregate queries over the selected regions is almost perfect (the error reduction is slightly less than 100 percent). The reason why the estimates are so accurate is because in the general case, when estimating the values of a data set, we expect to have an almost equal number of underestimates and overestimates, which will cancel each other out when we compute the aggregate value.

Even when we consider aggregate queries over the entire data set, we still get significantly more accurate estimates. As Fig. 14b shows, the error reduction for aggregate queries over the entire data set is approximately 2.5 percent to 5 percent for the selected regions we considered. This reduction is 10-25 times larger than for queries on the detailed values (see Fig. 12b).

In the last experiments, we evaluate what the added benefit of using marginals of order 3 for the selected regions is, as compared to marginals of order 1. Fig. 15a compares the reduction in the error of aggregate queries over the selected regions, when using marginals of order 1 (light colored bars) and order 3 (dark colored bars). The marginals of order 1 are able to reduce the estimation error by 75 percent, while the use of the marginals of order 3 contribute to the error reduction with an extra 25 percent, virtually eliminating the error in the estimation. Nevertheless, this added accuracy comes at the expense of space, since the marginals of order 3 require between one and two orders of magnitude more space than the marginals of order 1. The same error reduction is not observed when we consider aggregate queries over the entire data set. As Fig. 15b shows, the benefit of using marginals of higher order is in this case negligible.

5.4 Discussion

In this section, we presented a case study, where we applied the techniques described in this paper on a real data set. The outcome of the experiments show that the marginals can help in the estimation of the values toward which they are targeted. Nevertheless, the benefit in the overall estimation error is minimal. The above two distinct cases correspond to the following two extreme scenarios. The first is when the future query workload will consist of exactly the same queries as the selected regions. In this case, we will observe the error reduction bars depicted in dark color in Fig. 12. In the second scenario, the future query workload will consist of queries spread uniformly over the entire data set. This corresponds to the outcome of the experiments represented by the light color bars.

Obviously, the performance that we should expect from such a system lies somewhere in between those two extremes, according to how well we can predict the future query workload. The state-of-the-art commercial database management systems offer tools that, based on some query workload, can automatically generate the list of marginals that are most useful in answering the specified queries [38], [2], [26]. Then, these marginals are candidates for materialization. The assumption during this process is that the past workload is indicative of the future queries that will be posed on the system. Furthermore, queries that are similar to each other can be indentified [12], and the proposed solution can be generalized to suit classes of queries, rather than the specific queries present in the input workload.

Finally, we should note the case of aggregate queries, for which our techniques can provide estimates extremely close to the real values. In a real-life scenario, we expect that the query workload will be a mixture of queries asking for the detailed values, and aggregate queries that target subsets of the regions specified by the materialized marginals (i.e., the answer to these aggregate queries is not readily available). Then, the benefit of using our framework will be even greater than what is suggested in Section 5.2.

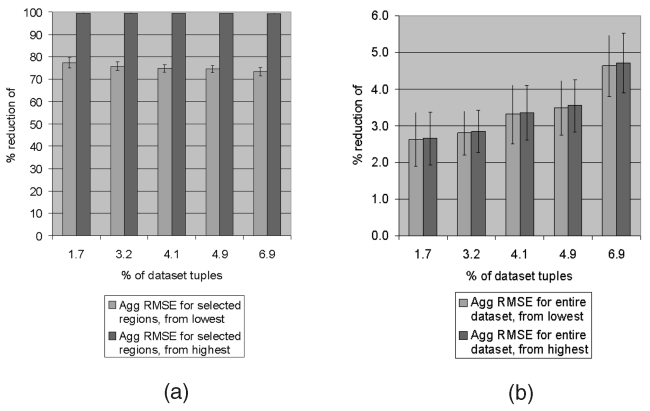


Fig. 15. The percentage of the error reduction when estimating the aggregate value of subsets of the selected regions only, and of subsets of the entire data set, by using the marginals of order 1 and order 3. (a) From selected regions and (b) from entire data set.

6 Related Work

The principle of maximum entropy [14] has been successfully applied in different domains, including linguistics [13] [9] and databases [16]. Faloutsos et al. apply maximum entropy, in addition to other techniques, for one-dimensional data reconstruction [16]. Our work generalizes the work of Faloutsos et al. to multiple dimensions.

There exists a sizeable bibliography in approximate query answering techniques [18], [30], [19], [37], [3], [35], [10], [11]. 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 there exist data that 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.

The problem of identifying interesting values in a data set is related to deviation detection. Arning et al. [1] try to identify the subset of a database that is most dissimilar to the rest of the data. Other approaches discuss algorithms specialized to metric spaces that scale to large data sets [23], [24], [32]. The drawback of these 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 require such input, making the whole procedure less cumbersome and more robust. A method that can identify outliers based on the density of data points is presented by Breunig et al. [8].

Our work is closer to the framework proposed by Sarawagi et al. [33]. They describe an algorithm that mines the data in a data cube for exceptions. We should note that in certain cases, the exceptions identified by this algorithm are the same as the deviations reported using our technique. This is because the log-linear models used in the above work produce the same model as the one derived using the maximum entropy principle, when the input is the same set of marginals [7]. However, their method is computationally expensive, and is tightly coupled with the computation of the entire data cube. Our solution does not have this requirement, which makes it more flexible than the above approach. In a subsequent study [34], Sarawagi describes a framework that automatically adapts the exploration of a datacube to the user's prior knowledge of the data. This framework is related to our work, employing maximum entropy and the aggregated information across different dimensions in order to select the dimension that holds the most unexpected values.

The maximum entropy principle has also been used for the problem of query selectivity estimation for binary data sets [29], [31]. The aim is to estimate a function of the joint probability distribution, given some of its marginals. However, it is not clear how the techniques developed in the above studies can be extended to nonbinary data sets. The work we present in this study is a detailed analysis of the benefits of using the maximum entropy principle and IPF for approximate querying answering and deviation detection, in the context of data warehouses.

7 Conclusions

In this work, we examine the problem of using only the information contained in the aggregates in order to extract estimates of the detailed values from which the aggregates were derived. Since this is an underspecified problem, we employ the information theoretic principle of *maximum entropy*. This principle allows us to produce estimates for the detailed values based only on the aggregated, without the need to make any additional assumptions about the detailed data distribution.

Based on this framework, we describe a technique for identifying deviations in multidimensional data sets. Deviations are those values that do not follow the general trends of the data set. Therefore, they may be interesting to the user analyst. The advantage of this technique is that it does not require any human intervention or domain specific knowledge. We also present a method that, based on the aggregates of a multidimensional data set, provides approximate answers to queries asking about the individual values of the data set. In this context, we show how we can extend the framework to offer quality guarantees for the reconstruction process.

An interesting application of the techniques presented in this study is in the area of selectivity estimation in the context of a Database Management System (DBMS). The nice property of these techniques is that they are bound to yield more accurate estimates than the ones used by the state-of-the-art databases, whose estimates are based on one-dimensional histograms. Moreover, the algorithms we describe in this paper can effectively use any information about the distributions they are trying to approximate that is available to the system. Hints, or even exact values for parts of the unknown distributions, may be available by examining the execution of the query workload [36], [5]. In such cases, we can use this information to improve the accuracy of the estimation of the unknown distribution.

Acknowledgments

The authors would like to thank Ken Sevcik and Renee Miller for useful discussions and many insightful comments. They are also grateful to Kyuseok Shim and the anonymous reviewers for numerous suggestions that helped improve the contents of the paper.

References

[1] A. Arning, R. Agrawal, and P. Raghavan, "A Linear Method for Deviation Detection in Large Databases," *Proc. Int'l Conf. Knowledge Discovery and Data Mining*, pp. 164-169, Aug. 1996.

[2] S. Agrawal, S. Chaudhuri, and V.R. Narasayya, "Automated Selection of Materialized Views and Indexes in SQL Databases," *Proc. Int'l Conf. Very Large Databases*, pp. 496-505, Sept. 2000.

[3] S. Acharya, P.B. Gibbons, V. Poosala, and S. Ramaswamy, "Join Synopses for Approximate Query Answering," *Proc. ACM SIGMOD Int'l Conf.,* pp. 275-286, June 1999.
[4] S. Abad-Mota, "Approximate Query Processing with Summary Tables in Statistical Databases," *Proc. Int'l Conf. Extending Database Technology,* pp. 499-515, Mar. 1992.
[5] N. Bruno and S. Chaudhuri, "Exploiting Statistics on Query Expressions for Optimization," *Proc. ACM SIGMOD Int'l Conf.,* pp. 263-274, June 2002.
[6] D. Bertsekas, *Constrained Optimization and 6 Multiplier Methods.* Academic Press, 1982.
[7] Y.M.M. Bishop, S.E. Fienberg, and P.W. Holland, *Discrete Multivariate Analysis: Theory and Practice.* MIT Press, 1975.
[8] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *Proc. ACM SIGMOD Int'l Conf.,* pp. 21-32, May 2000.
[9] A. Berger, S. Pietra, and V. Pietra, "A Maximum Entropy Approach to Natural Language Modelling," *Computational Linguistics,* vol. 22, no. 1, May 1996.
[10] D. Barbará and M. Sullivan, "Quasi-Cubes: Exploiting Approximations in Multidimensional Databases," *ACM SIGMOD Record,* vol. 26, no. 3, pp. 12-17, 1997.
[11] D. Barbará and X. Wu, "Using Loglinear Models to Compress Datacubes," *Web-Age Information Management,* pp. 311-322, June 2000.
[12] S. Chaudhuri, A. Gupta, and V.R. Narasayya, "Compressing SQL Workloads," *Proc. ACM SIGMOD Int'l Conf.,* pp. 488-499, June 2002.
[13] S.F. Chen and R. Rosenfeld, "A Gaussian Prior For Smoothing Maximum Entropy Models," Technical Report CMU-CS-99-108, Carnegie Mellon Univ., Feb. 1999.
[14] T. Cover and J. Thomas, *Elements of Information Theory.* Wiley, 1991.
[15] 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 Math. Statistics,* vol. 11, pp. 427-444, 1940.
[16] C. Faloutsos, H.V. Jagadish, and N. Sidiropoulos, "Recovering Information from Summary Data," *Proc. Very Large Databases Conf.,* pp. 36-45, Aug. 1997.
[17] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals." *Proc. Int'l Conf. Data Eng.,* pp. 152-159, Mar. 1996.
[18] Y. Ioannidis and V. Poosala, "Balancing Histogram Optimality and Practicality for Query Result Size Estimation," *Proc. ACM SIGMOD,* pp. 233-244, June 1995.
[19] H.V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K.C. Sevcik, and T. Suel, "Optimal Histograms with Quality Guarantees," *Proc. Int'l Conf. Very Large Data Bases,* pp. 275-286, Aug. 1998.
[20] H.V. Jagadish, L.V.S. Lakshmanan, and D. Srivastava, "Snakes and Sandwiches: Optimal Clustering Strategies for a Data Warehouse," *Proc. ACM SIGMOD ,* pp. 37-48, June 1999.
[21] H.V. Jagadish, I.S. Mumick, and A. Silberschatz, "View Maintenance Issues in the Chronicle Data Model," *Proc. ACM Symp. Principles of Database Systems,* pp. 113-124, June 1995.
[22] J.N. Kapur and H.K. Kesavan, *Entropy Optimization Principles with Applications.* Academic Press, Inc., 1992.
[23] E.M. Knorr and R.T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," *Proc. Int'l Conf. Very Large Data Bases,* pp. 392-403, Aug. 1998.
[24] E.M. Knorr and R.T. Ng, "Finding Intensional Knowledge of Distance-Based Outliers," *Proc. Int'l Conf. Very Large Data Bases,* pp. 211-222, Sept. 1999.
[25] S. Kullback, *Information Theory and Statistics.* John Wiley and Sons, 1968.
[26] G.M. Lohman and S.S. Lightstone, "SMART: Making DB2 (More) Autonomic," *Proc. Int'l Conf. Very Large Data Bases,* pp. 877-879, Aug. 2002.
[27] F. Malvestuto, "A Universal Scheme Approach to Statistical Databases Containing Homogeneous Summary Tables," *ACM Trans. Database Systems,* vol. 18, no. 4, pp. 678-708, Dec. 1993.
[28] H. Mannila, D. Pavlov, and P. Smyth, "Prediction with Local Patterns Using Cross-Entropy," *Proc. Int'l Conf. Knowledge Discovery and Data Mining,* pp. 357-361, Aug. 1999.
[29] G.S. Manku, S. Rajagopalan, and B.G. Lindsay, "Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets," *Proc. ACM SIGMOD,* pp. 251-262, June 1999.
[30] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita, "Improved Histograms for Selectivity Estimation of Range Predicates," *Proc. ACM SIGMOD,* pp. 294-305, June 1996.
[31] D. Pavlov, H. Mannila, and P. Smyth, "Probabilistic Models for Query Approximation with Large Sparse Binary Data Sets," *Proc. Conf. Uncertainty in Artificial Intelligence,* 2000.
[32] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Ouliers from Large Data Sets" *Proc. ACM SIGMOD,* pp. 427-438, May 2000.
[33] S. Sarawagi, R. Agrawal, and N. Megiddo, "Discovery-Sriven Exploration of OLAP Data Cubes," *Proc. Int'l Conf. Extending Database Technology,* pp. 68-182, Mar. 1998.
[34] S. Sarawagi, "User-Adaptive Exploration of Multidimensional Data," *Proc. Int'l Conf. Very Large Data Bases,* pp. 307-316, Sept. 2000.
[35] J. Shanmugasundaram, U.M. Fayyad, and P.S. Bradley, "Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions," *Proc. Int'l Conf. Knowledge Discovery and Data Mining,* pp. 223-232, Aug. 1999.
[36] M. Stillger, G.M. Lohman, V. Markl, and M. Kandil, "LEO - DB2's LEarning Optimizer," *Proc. Int'l Conf. Very Large Data Bases,* pp. 19-28, Sept. 2001.
[37] J.S. Vitter, M. Wang, and B. yer, "Data Cube Approximation and Histograms via Wavelets," *Proc. ACM Int'l. Conf. Information and Knowledge Management,* pp. 96-104, 1998.
[38] M. Zaharioudakis, R. Cochrane, G. Lapis, H. Pirahesh, and M. Urata, "Answering Complex SQL Queries Using Automatic Summary Tables," *Proc. ACM SIGMOD,* pp. 105-116, May 2000.

Themis Palpanas received the BSc degree in electrical and computer engineering from the National Technical University of Athens and the MSc and PhD degrees in computer science from the University of Toronto. He is currently a member of the IBM T.J. Watson Research Center. He has also worked at the University of California, Riverside, IBM Almaden Research Center, and Microsoft Research. He holds two US patents. His research interests include data analysis, data summarization, approximate answering, OLAP, streaming data, data mining, view maintenance, and caching.

Nick Koudas received the BTech degree from the University of Patras in Greece, the MSc degree from the University of Maryland at College Park, and the PhD degree from the University of Toronto. He is a faculty member at the University of Toronto, Department of Computer Science. He serves as an associate editor for the *Information Systems* journal and the *IEEE Transactions on Knowledge Data Engineering* journal. He is the recipient of the 1998 ICDE Best Paper award. His research interests include core database management, data quality, metadata management and its applications to networking. He is a member of the IEEE Computer Society.

Alberto Mendelzon received the MA, MSE, and PhD degrees from Princeton University. He spent a postdoctoral year at the IBM T.J. Watson Research Center and has been with the University of Toronto since 1980. He has been a visiting scientist at the IBM Centre for Advanced Studies, AT&T Bell Laboratories, NTT Basic Research Labs in Musashino, Japan, and the IASI in Rome. His research interests are in databases and knowledge-bases including database design theory, query languages, database visualization, query processing, belief revision and knowledge-base update, and global information systems. He has been an associate and acting chair of the Computer Systems Research Institute and chaired or cochaired the program committees of the ACM PODS and VLDB, as well as general chair of the ACM PODS. He has been a guest editor of the *Journal of Computer and System Sciences* and is on the editorial board of the *Journal of Digital Libraries*.