Regression Cubes with Lossless Compression and Aggregation

Yixin Chen, Guozhu Dong, *Senior Member*, *IEEE*, Jiawei Han, *Senior Member*, *IEEE*, Jian Pei, Benjamin W. Wah, *Fellow*, *IEEE*, and Jianyong Wang, *Member*, *IEEE*

Abstract—As OLAP engines are widely used to support multidimensional data analysis, it is desirable to support in data cubes advanced statistical measures, such as regression and filtering, in addition to the traditional simple measures such as count and average. Such new measures will allow users to model, smooth, and predict the trends and patterns of data. Existing algorithms for simple distributive and algebraic measures are inadequate for efficient computation of statistical measures in a multidimensional space. In this paper, we propose a fundamentally new class of measures, *compressible measures*, in order to support efficient computation of the statistical models. For compressible measures, we compress each cell into an auxiliary matrix with a size independent of the number of tuples. We can then compute the statistical measures for any data cell from the compressed data of the lower-level cells without accessing the raw data. Time- and space-efficient lossless aggregation formulae are derived for regression and filtering measures. Our analytical and experimental studies show that the resulting system, regression cube, substantially reduces the memory usage and the overall response time for statistical analysis of multidimensional data.

Index Terms—Aggregation, compression, data cubes, OLAP.

1 Introduction

Data warehouses provide online analytical processing (OLAP) tools for interactive analysis of multidimensional data. With years of research and development of data warehouse and OLAP technology [15], [7], [1], [34], a large number of data warehouses and data cubes have been successfully constructed and deployed in many applications.

The fast development of OLAP technology has led to high demand for more sophisticated data analyzing capabilities, such as prediction, trend monitoring, and exception detection of multidimensional data. Oftentimes, existing simple measures such as sum() and average() become insufficient, and more sophisticated statistical models, such as regression analysis, are desired to be supported in OLAP. Moreover, there are lots of applications with dynamically changing stream data generated continuously in a dynamic environment, with huge volume, infinite flow, and fast changing behavior. When collected, such data is almost always at a rather low level, consisting of various kinds of detailed temporal and other features. To find interesting or unusual patterns, it is essential to perform regression analysis at certain meaningful abstraction levels, discover critical changes of data, and drill down to some more detailed levels for in-depth analysis when needed.

To illustrate the multidimensional regression models, let us examine the following example.

Example 1. A power supply station collects infinite streams of power usage data, with the lowest granularity as individual user, location, and minute. Given a large number of users, it is only realistic to analyze the fluctuation of power usage at certain high levels, such as by city or district and by hour, in order to make timely power supply adjustments and handle unusual situations. For this application, simple measures like sum() are insufficient, and regression models for data at different levels are needed. A simplest regression in this case would be a linear model between time t and the power usage y as $y = \hat{\eta}_0 + \hat{\eta}_1 t$, where $\hat{\eta}_0$ and $\hat{\eta}_1$ are two parameters. The linear regression model describes the major trend of the power usage over a certain period.

Conceptually, for multidimensional analysis, one can view such a data warehouse as a virtual data cube, consisting of one measure and a set of dimensions, including one *regression dimension* and a few *standard dimensions* such as location and user-category. However, in practice, it is impossible to materialize such a stream data cube, since the materialization requires a huge (and potentially infinite) amount of data to be computed and stored in a multidimensional space. Some efficient methods

- *Y. Chen is with the Computer Science Department, Washington University in St. Louis, One Brookings Dr., St. Louis, MO 63130. E-mail: chen@cse.wustl.edu.*
- *G. Dong is with the Computer Science Department, Wright State University, 3640 Colonel Glenn Hwy., Dayton, OH 45435. E-mail: gdong@cs.wright.edu.*
- *J. Han is with the Computer Science Department, University of Illinois, 202 N. Goodwin St., Urbana, IL 61801. E-mail: hanj@uiuc.edu.*
- *J. Pei is with the Computer Science Department, Simon Fraser University, 8888 University Drive, Burnaby, BC Canada V5A 1S6. E-mail: jpei@cs.sfu.ca.*
- *B.W. Wah is with the Electrical and Computer Engineering Department, University of Illinois, 1308 Main St., Urbana, IL 61801. E-mail: wah@uiuc.edu.*
- *J. Wang is with the Computer Science Department, Tsinghua University, Beijing University, Beijing, China 100008. E-mail: jianyong@mail.tsinghua.edu.cn.*

Manuscript received 5 Feb. 2006; revised 31 May 2006; accepted 18 July 2006; published online 18 Oct. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0055-0206.

must be developed for systematic regression analysis over such data.

The example shows that multidimensional analysis of regression measures offers an analytical modeling engine to generate smoothed trend summarizations, prediction, and exception detection at different view levels. Such a regression-enabled OLAP engine is a powerful data analysis tool as well as a user-friendly environment for interactive data analysis. In this paper, we examine the issue of supporting advanced statistical measures, including regression and filtering measures, in a multidimensional space. In the rest of this section, we discuss why this is a challenging problem and outline our contributions.

1.1 Regression Cubes and Compressible Measures

Data warehouses and OLAP tools are based on a multidimensional data model. The model views data in the form of a data cube. A data cube is defined by dimensions and facts. In an n-dimensional data cube, the n dimensions are the perspectives or entities with respect to which an organization wants to keep records. A multidimensional data model is typically organized around a central theme such as *power usage*. The theme is represented by a fact table. Facts are numerical measures by which we want to analyze relationships between dimensions. In the multidimensional data cube, data are organized into multiple dimensions, and each dimension contains multiple levels of abstraction defined by concept hierarchies. Measures in a data cube can be classified into three categories based on the difficulty of aggregation:

- **Distributive.** An aggregate function is *distributive* if it can be computed in a distributed manner as follows: Suppose the data is partitioned into n sets. The computation of the function on each partition derives one aggregate value. If the result derived by applying the function to the n aggregate values is the same as that derived by applying the function on all the data without partitioning, the function can be computed in a distributive manner. count(), sum(), min(), and max() are distributive aggregate functions. A measure is *distributive* if it is obtained by applying a distributive aggregate function.
- **Algebraic.** An aggregate function is *algebraic* if it can be computed by an algebraic function with several arguments, each of which is obtained by applying a distributive aggregate function. For example, avg() (average) can be computed by sum()/count() where both sum() and count() are distributive aggregate functions. min_N(), max_N(), and stand_dev() are algebraic aggregate functions. A measure is *algebraic* if it is obtained by applying an algebraic aggregate function.
- **Holistic.** An aggregate function is *holistic* if there is no constant bound on the storage size needed to describe a subaggregate. That is, there does not exist an algebraic function with M arguments (where M is a constant) that characterize the computation. Common examples of holistic functions include median(), mode(), and rank(). A measure is *holistic* if it is obtained by applying a holistic aggregate function.

If we characterize the regression measure using the above classification, it seems to be a holistic measure because it requires the information of all the data points in a cuboid in order to compute the regression model. It is impossible to compute the regression model distributively and compose the high-level regression model merely from the low-level models.

Thus, the requirement for multilevel, multidimensional online analysis of advanced statistical measures, though desirable, raises a challenging research issue: *"Is it feasible to perform OLAP analysis for advanced statistical measures on huge volumes of data since a data cube is usually much bigger than the original data set, and its construction may take multiple database scans?"*

Our main idea in this paper is to compress the raw data for each cell, store only a minimum number of measures that are just sufficient to support the online analysis, and compute high-level measures from the corresponding low-level cells without accessing the raw data. Such a compression technique leads to the definition of a new category of measures:

- **Compressible.** An aggregation function is *compressible* if it can be computed by a procedure with a number of arguments from lower level cells, and the number of arguments is *independent* of the number of tuples in the data cuboid. In other words, for compressible aggregate functions, we can compress each cuboid, regardless of its size (i.e., the number of tuples), into a constant number of arguments, and aggregate the function based on the compressed representation. The data compression technique should satisfy the following requirements: 1) the compressed data should support efficient lossless or nearly lossless aggregation of regression measures in a multidimensional data cube environment and 2) the space complexity of compressed data should be low and be independent of the number of tuples in each cell, as the number of tuples in each cell may be huge. A measure is *compressible* if it is obtained by applying a compressible aggregate function.

In this paper, we will show that certain advanced statistical measures, including regression models and filters, are compressible measures. Note that compressible measures are different from holistic measures in that the number of arguments is a constant for compressible aggregate functions, but not for holistic aggregate functions. The compressible measures are different from algebraic measures in that the arguments for algebraic aggregate functions are distributive measures, while the arguments for compressible aggregate functions are still compressible measures. We will explain this in detail later.

One important assumption of this paper is that the measure and the attributes of regression dimensions have to be numerical and not categorical. Regression and filtering analysis require the measures of regression dimensions to be continuous or discrete numbers, but not unordered, discrete symbolic values. In the future, we plan to extend the theory to logistic regression which can handle categorical data.

1.2 Applications of Regression Cubes

We believe that supporting advanced statistical analysis is an important feature for the next-generation data cube technology and OLAP engines. In addition to performing the analysis at the lowest or the highest abstraction level, it is more important that the users would like to see regression models anywhere they want in a data cube space. Specifically, there are several important applications of regression cubes:

1. *Trend analysis*. A user can explore the data cube to find interesting cuboids guided by the regression models at different levels. For example, a sales manager might want to first examine the data at a higher abstraction level, find out a year when the sales drop down quickly, and drill down in that particular year to find out which months and states are main causes for the sales decrease. Such analysis will allow the users to interact with the data repository to find out certain views and levels that can provide key insights to their marketing.
2. *Exception detection*. In a regression cube, a user is able to find cuboids with an exceptional change rate (out of certain thresholds). In a typical case, an exception only occurs for cuboids at a particular view of a certain combination of the dimension levels, and a user cannot observe the exception at levels too low or too high. Therefore, it is an essential requirement that a user can explore the data cube space and generate regression models for different cuboids efficiently.
3. *Incremental cubing*. Using the regression cube technique, the user can scan the data only once, compute and store the regression measures for the cells at the lowest level, and discard the raw data. For any newly generated data, a user only needs to generate the regression model for the new part. The regression models for all the cuboids after any update can be efficiently recovered using the compression and aggregation techniques proposed in this paper. Without compression, a user has to maintain all historical raw data and update the relevant data models from scratch for every update.
4. *Partial materialization*. Due to space and time limitations, it is usually desirable to materialize only a subset of the cuboids in the data cube space, and there have been an extensive study for partial materialization schemes of data cubes. When such partially materialized cubes are used, it is necessary to be able to derive the regression model of any cuboid not materialized on the fly from other materialized cuboids. The proposed regression cube will support this operation.

In summary, the proposed regression cube technique will provide users insights to the trend of data and the correlations among dimensions, allow users to explore the data cube space efficiently, and support efficient regression computation when the data cube is constructed incrementally and/or partially.

1.3 Research Contributions

In this paper, we propose the concept of regression cubes and a data cell compression technique NCR (*nonlinear compression representation*) to support efficient OLAP operations in the regression cubes. Our study shows that to support multidimensional regression analysis, only a small number of *compressed measures* need to be registered. We develop lossless aggregation formulae based on the NCRs.

The space complexity of the compressed NCR in data cells is *independent* of the number of tuples in those cells and is only *quadratic* to the number of regression coefficients. Therefore, we make the regression measure a *compressible measure* and the space consumption is significantly reduced using NCR compression.

In addition to regression analysis, we have extended the results to filtering analysis of time-series data. We have shown that filters are also compressible measures by showing that we can compress each data cell into a small, fixed-size data block while still allowing lossless aggregation of the measures.

The rest of the paper is organized as follows: In Section 2, we define the basic concepts and introduce the research problem. In Section 3, we illustrate the key concepts and results of this paper using linear regression, the simplest special case of general multiple linear regression analysis. In Section 4, we present the theoretical foundation for computing regression models in data cubes. We propose the NCR compression technique to support online regression analysis of stream data in data cubes, and present lossless NCR aggregation formulae in regression and standard dimensions. In Section 5, we develop lossless compression schemes for predictive filters. We present performance studies in Section 6, discuss related work in Section 7, and give conclusions in Section 8.

2 Problem Definition

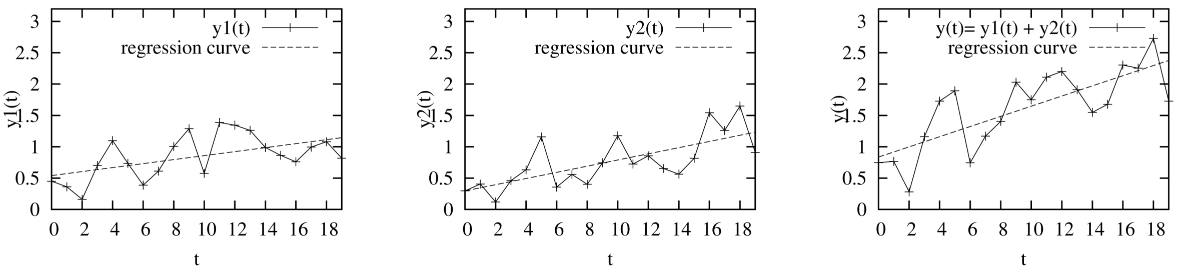
In this section, we introduce the basic concepts related to regression analysis in data cubes.

2.1 Data Cubes

To perform multidimensional, multilevel analysis, we need to introduce some basic terms related to data cubes. Let $\mathcal{D}$ be a relational table, called the base table, of a given cube. The set of all *attributes* $\mathcal{A}$ in $\mathcal{D}$ are partitioned into two subsets, the *dimensional attributes* DIM and the *measure attributes* M (so $DIM \cup M = \mathcal{A}$ and $DIM \cap M = \emptyset$). The measure attributes functionally depend on the dimensional attributes in $\mathcal{D}$ and are defined in the context of data cube using some typical aggregate functions, such as count(), sum(), avg(), or some regression related measures to be studied here.

A key feature of data warehouses is that they are time-variant. Data are stored to provide information from a historical perspective. Every structure in the data warehouse contains, either implicitly or explicitly, a dimension of time. In this paper, we assume that the time dimension is always a dimensional attributes for stream data analysis.

Example 2. For our power supply analysis in Example 1, the dimensional attributes may include time, user location,



(a) (b) (c)

Fig. 1. An example showing aggregation in the *location* dimension, a standard dimension. $y(t)$ represents the amount of power usage in billion kilowatts during the $t\mathrm{th}$ minute. We aggregate $k = 2$ component cells in time $t \in [0, 19]$. (a) $y_1(t)$: Eastern US power in [0, 19]. (b) $y_2(t)$: Western US power in [0, 19]. (c) $y(t)$: US power in [0, 19].

and user category, and the measure attributes include amount of power usage and the regression model parameters.

A tuple with schema $\mathcal{A}$ in a multidimensional space is called a **cell**. Given three distinct cells c_1, c_2, and c_3, c_1 is an **ancestor** of c_2, and c_2 a **descendant** of c_1 iff on every dimensional attribute, either c_1 and c_2 share the same value, or c_1's value is a generalized value of c_2's in the dimension's concept hierarchy. A tuple $c \in \mathcal{D}$ is called a **base cell**. A base cell does not have any descendant. A cell c is an **aggregated cell** iff it is an ancestor of some base cells. For each aggregated cell, the values of its measure attributes are derived from the set of its descendant cells.

2.2 Regression and Standard Dimensions

In a regression analysis environment, the dimensions can be divided into *regression dimensions*, which are those involved in regression, and other dimensions, which are called *standard dimensions*. We denote the dimensions as $(x_1, x_2, \cdots, x_p, s_1, s_2, \cdots, s_q)$, where x_is are the regression dimensions (or regression attributes) and s_is are the standard dimension (or standard attributes). The regression dimensions should be represented in numerical attributes. Typically, the time dimension is used in regression and we assume it to be x_1 in this paper. As usual, there is a hierarchy for each dimension. Conceptually, the base facts of interest is a base relational table $T(x_1, x_2, \cdots, x_p, s_1, s_2, \cdots, s_q, y_1, y_2, \cdots, y_l)$, where y_1 to y_l are measure attributes. Without loss of generality, to simplify our presentation, we assume there is only one measure attribute y to be modeled by regression in this paper.

In such a data cube, each primitive aggregation is either over a standard dimension, or over a regression dimension. For the former, aggregation means to sum up the measure y over all corresponding base cells. For the latter, aggregation means to merge all base cells into the resulting cell.

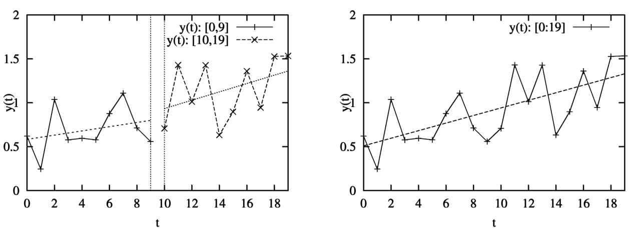
Example 3. Continuing from Example 1, for our power supply analysis, we assume that the regression dimension is the time dimension t, and that the user location L is a standard dimension. We assume that linear least square error (LSE) regression is used, i.e., for each cell, we use the least square error as the standard to model a linear relationship between time t and the power usage y as $y = \hat{\eta}_0 + \hat{\eta}_1 t$.

Figs. 1 and 2 show some examples to illustrate the regression models and aggregation of regression models over standard and regression dimensions. Fig. 1 shows the aggregation over the standard dimension of user locations. Figs. 1a and 1b show two data cells for the amount of power usage of eastern and western US, respectively. Suppose the cell for the overall US power usage is the ancestor cell of these two cells, Fig. 1c plots this cell aggregated from Figs. 1a and 1b. It is obvious that the measure attribute y (amount of power usage) in Fig. 1c are obtained by summing up the corresponding descendants.

Fig. 2a shows two cells for power usage over two time intervals [0,9] and [10,19], while Fig. 2b shows the resulting cell after aggregating the two base cells along the regression (time) dimension. We see that aggregation along a regression dimension means to merge all base cells into the resulting cell.

2.3 Multidimensional Regression Analysis

In this study, we assume that the data is collected at the most detailed level in a multidimensional space, which may represent time, location, user, theme, and other semantic information. Thus, the direct regression of data at the most detailed level may generate a large number of regression lines, but still cannot tell the general trends contained in the data.



(a) (b)

Fig. 2. An example showing aggregation in the time dimension. $y(t)$ represents the power usage in billion kilowatts during the $t\mathrm{th}$ minute. We aggregate time series over two time intervals, namely, [0, 9] and [10, 19]. (a) Power in [0, 9] and [10, 19]. (b) Power in [0, 19].

As we have discussed before, data measures are classified by the difficulty in aggregating them. In this paper, we want to study how to aggregate the regression measures. In our running example, we want to compute the regression measures of the aggregated curve $y(t)$ from the regression measures of base-cell curves $y_1(t)$ and $y_2(t)$. An aggregation function is *lossless* if the values of the regression measures obtained from the aggregating computation are identical to the original regression measures computed from the raw data, and is *lossy* otherwise.

Our task is to *perform time and space-efficient high-level, on-line, multidimensional regression analysis in a data cube environment.* There are two issues to be resolved: 1) what to store for each cell and 2) how to aggregate the data stored. Our objective is to compress the data to minimize the space requirement and support lossless aggregation in both standard and regression dimensions.

3 Multidimensional Linear Regression Analysis

To give a gentle introduction to the theory, we first work on linear regression, a simple special case. After reviewing the basics of linear regression, we illustrate the research problem and our solutions for linear regression measures.

3.1 Linear Regression Analysis for Data Cubes

Linear regression is one of the most common methods to model the overall trend of a series of data. We now briefly review the linear regression analysis.

Suppose we have n tuples in a cell: $(x_i, y_i), i = 1..n$, where x_i is the regression dimension attribute of the ith tuple, and each scalar y_i is the measure attribute of the ith tuple. The linear regression function is defined as $E(y_i|x_i) = \eta_0 + \eta_1 x_i$, where $\eta = (\eta_0, \eta_1)^T$ is a 2×1 vector of *regression parameters*.

Remember we have n samples in the cell, so we can write all the measure attribute values into an $n \times 1$ vector $\mathbf{y} = (y_1, \cdots, y_n)^T$ and we collect the terms x_i into an $n \times 2$ *model matrix* $\mathbf{U}$ as follows:

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & . & . & 1 \\ x_1 & x_2 & . & . & x_n \end{pmatrix}^T, \tag{1}$$

and we can now write the regression function in matrix form as $E(\mathbf{y}|\mathbf{U}) = \mathbf{U}\eta$.

We consider the *ordinary least square* (OLS) estimates of regression parameters η in this paper. The OLS estimate $\hat{\eta}$ of η is the argument that minimizes the *residual sum of squares function*

$$RSS(\eta) = (\mathbf{y} - \mathbf{U}\eta)^T(\mathbf{y} - \mathbf{U}\eta). \tag{2}$$

For the linear regression, we have

$$RSS(\eta) = \sum_{i=1}^{n} (y_i - (\eta_0 + \eta_1 x_i))^2.$$

Differentiating (2) and setting the result to be zero: $\frac{\partial}{\partial \eta} RSS(\eta) = 0$, we get the following result about $\hat{\eta}$:

$$\hat{\eta} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y}. \tag{3}$$

Example 4. Continuing from Example 3, in Figs. 1 and 2, a linear least square error(LSE) regression is used, i.e., for each cell, we model a linear relationship between time t and the power usage y as $y = \hat{\eta}_0 + \hat{\eta}_1 t$.

The data cell in Fig. 1a has $n = 20$ tuples, where

$$\begin{aligned}\mathbf{y} = (y_1, \cdots, y_{20})^T =& (0.450629, 0.361298, 0.161426, 0.702031, \\ & 1.09753, 0.734187, 0.388058, 0.611386, \\ & 1.00416, 1.28791, 0.574888, 1.38808, \\ & 1.34544, 1.26086, 0.987224, 0.862096, \\ & 0.759747, 0.994751, 1.0792, 0.818197)^T,\end{aligned}$$

and

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & . & . & 1 \\ t_1 & t_2 & . & . & t_n \end{pmatrix}^T = \begin{pmatrix} 1 & 1 & . & . & 1 \\ 0 & 1 & . & . & 19 \end{pmatrix}^T. \tag{4}$$

We can compute the regression parameters using (3):

$$\hat{\eta} = \begin{pmatrix} \hat{\eta}_0 \\ \hat{\eta}_1 \end{pmatrix} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y} = \begin{pmatrix} 0.540995 \\ 0.0318379 \end{pmatrix}. \tag{5}$$

Fig. 1a plots the resulting regression curve of $E(y_i|t) = \hat{\eta}_0 + \hat{\eta}_1 t$. Other regression curves in Figs. 1 and 2 are obtained following the same way:

3.1.1 Aggregation in Standard Dimensions

We now consider aggregation (roll-up) in a standard dimension. Suppose that c_a is a cell aggregated from a number of component cells $c_1, \ldots, c_m$, and that we want to compute the linear model of c_a's data. The component cells can be base cells at the bottom level, or derived descendant cells at higher levels.

In this roll-up, the stream data for c_a is defined to be the summation of the stream data for the component cells. More precisely, all component cells are supposed to have same number of tuples and they only differ in one standard dimension. Suppose that each component cell has n tuples, then c_a also has n tuples; let $y_{a,i}$, where $i = 1..n$, be the measure attribute of the ith tuple in c_a, and let $y_{j,i}$, where $j = 1..m$, be the measure attribute of the ith tuple in c_j. Then, $y_{a,i} = \sum_{j=1}^{m} y_{j,i}$ for $i = 1..n$. In matrix form, we have $\mathbf{y}_a = \sum_{j=1}^{m} \mathbf{y}_j$ where $\mathbf{y}_a$ and $\mathbf{y}_j$ are vectors of measure attributes for corresponding cells, respectively.

Suppose we have the linear regression measures but not the raw data of all the descendant cells. We wish to derive the regression measures of the aggregated cell from the those of the component cells.

Example 5. Continuing from Example 4, Fig. 1 illustrates an example of aggregation in the standard dimension of user locations. The ancestor cell c_a in Fig. 1c is aggregated from the two component cells c_1 and c_2 in Figs. 1a and 1b, respectively.

We have $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.540995, 0.0318379)^T$ for $y_1(t)$, $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.294875, 0.0493375)^T$ for $y_2(t)$, and $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.83587, 0.0811754)^T$ for $y(t)$.

Now, the problem is, in multidimensional analysis for large-scale data, especially stream data, we cannot afford to store the raw data. Also, it is infeasible due to time and space costs to materialize and store all cuboids. Therefore, it is necessary to aggregate the analytical measures from lower levels.

Specifically, we want to compute the regression measures $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.83587, 0.0811754)^T$ of the aggregated cell $y(t)$ from the data of $y_1(t)$ and $y_2(t)$. Since we do not store the raw data, we need to compress the data. It will be a lossless compression if $(\hat{\eta}_0, \hat{\eta}_1)$ of $y(t)$ reconstructed from compressed data are identical to those obtained from the raw data, and will be a lossy compression otherwise.

3.1.2 Aggregation in Regression Dimensions

We now consider aggregation (roll-up) in regression dimensions. Still, we suppose that c_a is a cell aggregated from a number of component cells $c_1, \ldots, c_m$, which differ in one regression dimension, and that we want to compute the linear regression model of c_a's stream data. The component cells can be base cells or derived descendant cells.

In this roll-up, the set of tuples of c_a is defined to be the union of tuples of the component cells. Since c_a contains the union of tuples of all component cells, in matrix form, we have $\mathbf{y}_a = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_m)^T$, where $\mathbf{y}_a$ and $\mathbf{y}_j$ are vectors of measure attributes for corresponding cells, respectively.

Example 6. Continuing from Example 4, Fig. 2 illustrates an example of aggregation in the regression dimension of time. The ancestor cell c_a in Fig. 2b is aggregated from the two component cells c_1 and c_2 in Fig. 2a. We have $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.582995, 0.0240189)^T$ for $y_1(t)$, $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.459046, 0.047474)^T$ for $y_2(t)$, and $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.509033, 0.0431806)^T$ for $y(t)$. Again, in this example, we want to compute the regression measures $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.509033, 0.0431806)^T$ of the aggregated cell $y(t)$ from the data of $y_1(t)$ and $y_2(t)$.

3.2 SA Compression: A Lossy Compression

When faced with the problem of multidimensional aggregation of regression measures, a simple and intuitive compression technique would be to store just $(\hat{\eta}_0, \hat{\eta}_1)$ for each cell, and using the following formulae during aggregation:

$$\begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_r = \begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_1 + \begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_2 \quad \text{for standard dimensions}$$
$$\begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_r = \frac{1}{2}\left[\begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_1 + \begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_2\right] \quad \text{for regression dimensions,}$$

where $(\hat{\eta}_0, \hat{\eta}_1)_r^T$ are the regression measures of the aggregated cell, and $(\hat{\eta}_0, \hat{\eta}_1)_1^T$ and $(\hat{\eta}_0, \hat{\eta}_1)_2^T$ are regression measures of two component cells.

Here, we use summation to aggregate the regression measures in standard dimensions, and use averaging to aggregate the regression measures in regression dimensions. It is called sum/avg (SA) compression. Let us explain the intuition of SA compression from a geometry point of view. For a regression line $E(y|x) = \eta_0 + \eta_1 x$, η_0 is the intercept and η_1 is the slope. For aggregations in a standard dimension, we sum up the measure y over a same region of x. Therefore, the slope and intercept of the regression lines should accumulate. This calculation is actually lossless along standard dimensions. For aggregations in a regression dimension, suppose we have two regression lines for two cells with the same length of regression region, and we should expect that the overall slope and intercept will get averaged out.

Example 7. Let us check if the SA compression is lossy or lossless. Continuing from Example 6, if we use the SA compression to aggregate the measures along the regression (time) dimension, we have:

$$\begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_r = \frac{1}{2} \times \left[\begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_1 + \begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix}_2\right] = \begin{pmatrix}0.5210205\\ 0.03574645\end{pmatrix}.$$

We can see that the reconstructed regression measures are different from the original regression measures $(\hat{\eta}_0, \hat{\eta}_1)^T = (0.509033, 0.0431806)^T$ of the aggregated cell. Hence, the SA compression is lossy.

We note that the SA compression is applicable to complex aggregations over both standard and regression dimensions, but is lossy due to the errors incurred in regression dimensions.

3.3 NCR: A Lossless Compression

In this paper, we will present a lossless compressed representation of the regression models of cells of data cubes called the *Nonlinear Compression Representation (NCR)*. The compressed NCR for the materialized base cells will be sufficient for deriving NCR and regression models of all other cells losslessly. Moreover, the size of NCR is independent of the number of tuples in each cell. Since the number of tuples in a cell could be very huge at higher levels, it is desirable that the size of the compressed data for each cell is *independent* of the number of tuples.

We will define the NCR compression in the next section. Here, we use the result to illustrate the application of NCR. The NCR for each cell consists of two matrices: $\hat{\eta}$ for the regression measures, and $\mathbf{\Theta}$ to store some auxiliary information needed for lossless aggregation. For linear regression in the running examples, the two matrices in NCR will be:

$$\hat{\eta} = \begin{pmatrix}\hat{\eta}_0\\ \hat{\eta}_1\end{pmatrix} \quad \text{and} \quad \mathbf{\Theta} = \begin{pmatrix}\theta_{00} & \theta_{01}\\ \theta_{10} & \theta_{11}\end{pmatrix},$$

where $\theta_{00} = n$, $\theta_{01} = \theta_{10} = (t_0 + t_1)n/2$,

$$\theta_{11} = (t_1(t_1+1)(2t_1+1) - (t_0-1)t_0(2t_0-1))/6,$$

n is the number of tuples, and t_0 and t_1 are starting and ending time of the cell being compressed, respectively. Therefore, for linear regression, we need to store five numerical values in its NCR for each cell c: $NCR(c) = (\hat{\eta}_0, \hat{\eta}_1, n, \theta_{01}, \theta_{11})$. Note that the NCR includes the necessary information about regression measures $(\hat{\eta}_0, \hat{\eta}_1)^T$ and some additional measures n, θ_{01}, θ_{11}.

We have designed the NCR compression in such a way that the regression models of any data cell can be losslessly reconstructed from the NCRs of its descendant cells without accessing the raw data. We demonstrate the lossless aggregation of the NCR compression in both standard and regression dimensions using the running example. In our running example, there are two base cells c_1 and c_2, and one aggregated cell c_a. Let

$$NCR(c_1) = (\hat{\eta}_1, \mathbf{\Theta}_1),$$
$$NCR(c_2) = (\hat{\eta}_2, \mathbf{\Theta}_2), \text{ and}$$
$$NCR(c_a) = (\hat{\eta}_a, \mathbf{\Theta}_a),$$

using the aggregation formulae we will develop in the next section, $NCR(c_a)$ can be derived from $NCR(c_1)$ and $NCR(c_2)$ as follows:

- For standard dimension aggregation:

$$\hat{\eta}_a = \hat{\eta}_1 + \hat{\eta}_2, \quad (6)$$

$$\mathbf{\Theta}_a = \mathbf{\Theta}_1 = \mathbf{\Theta}_2. \quad (7)$$

- For regression dimension aggregation:

$$\hat{\eta}_a = (\mathbf{\Theta}_1 + \mathbf{\Theta}_2)^{-1}(\mathbf{\Theta}_1\hat{\eta}_1 + \mathbf{\Theta}_2\hat{\eta}_2), \quad (8)$$

$$\mathbf{\Theta}_a = \mathbf{\Theta}_1 + \mathbf{\Theta}_2. \quad (9)$$

Example 8. Continuing from Example 5, Fig. 1 illustrates an example of aggregation in the standard dimension of user locations. The ancestor cell c_a in Fig. 1c is aggregated from the two component cells c_1 and c_2 in Figs. 1a and 1b, respectively. Their NCRs are as follows:

$$\hat{\eta}_1 = \begin{pmatrix} 0.540995 \\ 0.0318379 \end{pmatrix} \quad \mathbf{\Theta}_1 = \begin{pmatrix} 20 & 190 \\ 190 & 2,470 \end{pmatrix}$$
$$\hat{\eta}_2 = \begin{pmatrix} 0.294875 \\ 0.0493375 \end{pmatrix} \quad \mathbf{\Theta}_2 = \begin{pmatrix} 20 & 190 \\ 190 & 2,470 \end{pmatrix}$$
$$\hat{\eta}_a = \begin{pmatrix} 0.83587 \\ 0.0811754 \end{pmatrix} \quad \mathbf{\Theta}_a = \begin{pmatrix} 20 & 190 \\ 190 & 2,470 \end{pmatrix}.$$

It can be verified that $NCR(c_a) = (\hat{\eta}_a, \mathbf{\Theta}_a)$ can be losslessly reconstructed from $NCR(c_1) = (\hat{\eta}_1, \mathbf{\Theta}_1)$ and $NCR(c_2) = (\hat{\eta}_2, \mathbf{\Theta}_2)$ using (6) and (7).

Example 9. Continuing from Example 6, Fig. 2 illustrates an example of aggregation in the regression dimension of time. The ancestor cell c_a in Fig. 2b is aggregated from the two component cells c_1 and c_2 in Fig. 2a. Their NCRs are as follows:

$$\hat{\eta}_1 = \begin{pmatrix} 0.582995 \\ 0.0240189 \end{pmatrix} \quad \mathbf{\Theta}_1 = \begin{pmatrix} 10 & 45 \\ 45 & 285 \end{pmatrix}$$
$$\hat{\eta}_2 = \begin{pmatrix} 0.459046 \\ 0.047474 \end{pmatrix} \quad \mathbf{\Theta}_1 = \begin{pmatrix} 10 & 145 \\ 145 & 2,185 \end{pmatrix}$$
$$\hat{\eta}_a = \begin{pmatrix} 0.509033 \\ 0.0431806 \end{pmatrix} \quad \mathbf{\Theta}_1 = \begin{pmatrix} 20 & 190 \\ 190 & 2,470 \end{pmatrix}.$$

It can be verified that $NCR(c_a) = (\hat{\eta}_a, \mathbf{\Theta}_a)$ can be losslessly reconstructed from $NCR(c_1) = (\hat{\eta}_1, \mathbf{\Theta}_1)$ and $NCR(c_2) = (\hat{\eta}_2, \mathbf{\Theta}_2)$ using (8) and (9).

4 Multidimensional General Multiple Linear Regression Analysis

In this section, we review the theory of general multiple linear regression (GMLR) and propose our compression technique to support the construction of regression cubes.

4.1 Theory of GMLR

We now briefly review the theory of GMLR (See [10], for example, for more details). Suppose we have n tuples in a cell: $(\mathbf{x}_i, y_i), i = 1..n$, where $\mathbf{x}_i^T = (x_{i1}, x_{i2}, \cdots, x_{ip})$ are the p regression dimensions of the ith tuple, and each scalar y_i is the measure of the ith tuple. To apply multiple linear regression, from each $\mathbf{x}_i$, we compute a vector of k terms $\mathbf{u}_i$:

$$\mathbf{u}_i = \begin{pmatrix} u_0 \\ u_1(\mathbf{x}_i) \\ \cdots \\ u_{k-1}(\mathbf{x}_i) \end{pmatrix} = \begin{pmatrix} 1 \\ u_{i,1} \\ \cdots \\ u_{i,k-1} \end{pmatrix}. \quad (10)$$

The first element of $\mathbf{u}_i$ is $u_0 = 1$ for fitting an intercept, and the remaining $k-1$ terms $u_j(\mathbf{x}_i), j = 1..k-1$ are derived from the regression attributes $\mathbf{x}_i$ and are often written as $u_{i,j}$ for simplicity. $u_j(\mathbf{x}_i)$ can be any kind of function of $\mathbf{x}_i$. It could be as simple as a constant, or it could also be a complex nonlinear function of $\mathbf{x}_i$. For example, for time series regression (where $\mathbf{x} = (t)$), we have $k = 2$ and $u_1 = t$; for 2D spatial regression (where $\mathbf{x} = (x_1, x_2)$) used in spatial and moving-object databases, we can have $k = 4$, $u_1 = x_1$, $u_2 = x_2$, and $u_3 = x_1x_2$.

The nonlinear regression function is defined as follows:

$$E(y_i|\mathbf{u}_i) = \eta_0 + \eta_1 u_{i1} + \cdots + \eta_{k-1}u_{i,k-1} = \eta^T\mathbf{u}_i, \quad (11)$$

where $\eta = (\eta_0, \eta_1, \cdots, \eta_{k-1})^T$ is a $k \times 1$ vector of *regression parameters*, and k is the *number of regression terms*.

Remember we have n samples in the cell, so we can write all the measure attribute values into an $n \times 1$ vector $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$. and we collect the terms $u_{i,j}$ into an $n \times k$ *model matrix* $\mathbf{U}$ as follows, where $u_{i,j} = u_j(x_i)$:

$$\mathbf{U} = \begin{pmatrix} 1 & u_{11} & u_{12} & \cdots & u_{1,k-1} \\ 1 & u_{21} & u_{22} & \cdots & u_{2,k-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & u_{n1} & u_{n2} & \cdots & u_{n,k-1} \end{pmatrix}. \quad (12)$$

We can now write the regression function as $E(\mathbf{y}|\mathbf{U}) = \mathbf{U}\eta$.

Definition 1. *The OLS estimate $\hat{\eta}$ of η is the argument that minimizes the residual sum of squares function $RSS(\eta) = (\mathbf{y} - \mathbf{U}\eta)^T(\mathbf{y} - \mathbf{U}\eta)$. If the inverse of $(\mathbf{U}^T\mathbf{U})$ exists, OLS estimates $\hat{\eta}$ of regression parameters are unique and are given by:*

$$\hat{\eta} = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y}. \quad (13)$$

In the rest of this paper, without loss of generosity, we only consider the case where the inverse of $(\mathbf{U}^T\mathbf{U})$ exists. If the inverse of $(\mathbf{U}^T\mathbf{U})$ does not exist, then the matrix $(\mathbf{U}^T\mathbf{U})$ is of less than full rank and we can always use a subset of the u terms in fitting the model so that the reduced model matrix has full rank. Our major results will remain valid in this case.

The memory size of $\mathbf{U}$ in (12) is nk, and the size of $\mathbf{U}^T\mathbf{U}$ is k^2, where n is the number of tuples of a cell and k is the number of regression terms which is usually a very small constant independent of the number of tuples. For example, k is two for linear regression, and three for quadratic regression. Therefore, the overall space complexity is $O(n)$ and only linear to n.

4.2 Nonlinear Compression Representation of a Data Cell

We propose a compressed representation of data cells to support multidimensional GMLR analysis. The compressed information for the materialized cells will be sufficient for deriving regression models of all other cells.

Definition 2. *For multidimensional online regression analysis in data cubes, the Nonlinear Compression Representation (NCR) of a data cell c is defined as the following set:*

$$NCR(c) = \{\hat{\eta}_i | i = 0, \cdots, k-1\} \cup \{\theta_{ij} | i, j = 0, \cdots, k-1, i \leq j\}, \tag{14}$$

where

$$\theta_{ij} = \sum_{h=1}^{n} u_{hi} u_{hj}. \tag{15}$$

It is useful to write NCR in the form of matrices. In fact, elements in an NCR can be arranged into two matrices: $\hat{\eta}$ and $\mathbf{\Theta}$, where $\hat{\eta} = (\hat{\eta}_0, \hat{\eta}_1, \cdots \hat{\eta}_{k-1})^T$ and

$$\mathbf{\Theta} = \begin{pmatrix} \theta_{00} & \theta_{01} & \theta_{02} & \cdots & \theta_{0,k-1} \\ \theta_{10} & \theta_{11} & \theta_{12} & \cdots & \theta_{1,k-1} \\ . & . & . & . & . \\ . & . & . & . & . \\ \theta_{k-1,0} & \theta_{k-1,1} & \theta_{k-1,2} & \cdots & \theta_{k-1,k-1} \end{pmatrix}.$$

We can write an NCR in a matrix form as $NCR = (\hat{\eta}, \mathbf{\Theta})$.

Note that, since $\theta_{ij} = \theta_{ji}$ and, thus, $\mathbf{\Theta}^T = \mathbf{\Theta}$, we only need to store the upper triangle of $\mathbf{\Theta}$ in an NCR. Therefore, the size of an NCR is $S(k) = (k^2 + 3k)/2$. The following property of NCRs indicates that this representation is economical in space and scalable for large data cubes:

Theorem 4.1. *The size $S(k)$ of an NCR of a data cell is quadratic in k, the number of regression terms, and is independent of n, the number of tuples in the data cell.*

$\hat{\eta}$ in NCR provides the regression parameters one expects to see in the description of a regression model. $\mathbf{\Theta}$ is an auxiliary matrix that facilitates the aggregation of regression models in a data cube environment. As we will show in the next two sections, by storing such a compressed representation in the base cells, we can compute the regression models of all other cells in a data cube. We have illustrated the use of NCRs in Examples 8 and 9.

We have designed the NCR compression in such a way that the compressed data contains sufficient information to support the desired lossless aggregation. In fact, the regression models of any data cell can be losslessly reconstructed from the NCRs of its descendant cells without accessing the raw data.

Theorem 4.2. *If we materialize the NCRs of the base cells at the lowest level of a data cube, then we can compute the NCRs of all other cells in the data cube losslessly.*

We will prove this theorem in the rest of this section. We consider the aggregation in standard dimensions in Section 4.3 and in regression dimensions in Section 4.4.

4.3 Lossless Aggregation in Standard Dimensions

We now consider aggregation (roll-up) in standard dimensions. Suppose that c_a is a cell aggregated from a number of component cells $c_1, \ldots, c_m$, and that we want to compute the regression model of c_a's stream data. The component cells can be base cells, or derived descendant cells.

In this roll-up, the measure attribute of c_a is defined to be the summation of the measure attributes of the component cells. More precisely, all component cells are supposed to have the same number of tuples and they only differ in one standard dimension. Suppose that each component cell has n tuples, then c_a also has n tuples; let $y_{a,i}$, where $i = 1..n$, be the measure attribute of the ith tuple in c_a, and let $y_{j,i}$, where $j = 1..m$, be the measure attribute of the ith tuple in c_j. Then, $y_{a,i} = \sum_{j=1}^{m} y_{j,i}$ for $i = 1..n$. In matrix form, we have:

$$\mathbf{y}_a = \sum_{j=1}^{m} \mathbf{y}_j, \tag{16}$$

where $\mathbf{y}_a$ and $\mathbf{y}_j$ are vectors of measure attributes for corresponding cells, respectively.

Suppose we have the NCRs but not the raw data of all the descendant cells we wish to derive $NCR(c_a)$ from the NCRs of the component cells. The following theorem shows how this can be done.

Theorem 4.3 (Lossless aggregation in standard dimensions). *For aggregation in a standard dimension, suppose $NCR(c_1) = (\hat{\eta}_1, \mathbf{\Theta}_1)$, $NCR(c_2) = (\hat{\eta}_2, \mathbf{\Theta}_2), \cdots, NCR(c_m) = (\hat{\eta}_m, \mathbf{\Theta}_m)$ are the NCRs for the m component cells, respectively, and suppose $NCR(c_a) = (\hat{\eta}_a, \mathbf{\Theta}_a)$ is the NCR for the aggregated cell, then $NCR(c_a)$ can be derived from that of the component cells using the following equations:*

a. $\hat{\eta}_a = \sum_{i=1}^{m} \hat{\eta}_i$ *and*
b. $\mathbf{\Theta}_a = \mathbf{\Theta}_i, \quad i = 1..m.$

Proof. a) Since all c_i and c_as differ only in a standard dimension, they are identical in the regression dimensions. We see from (10) that all $u_{i,j}$ terms used in the regression model are only related to the regression dimensions. Therefore, all cells have identical $u_{i,j}$ terms, where $i = 1..n, j = 0..k-1$, and we have:

$$\mathbf{U}_a = \mathbf{U}_i, \quad i = 1..m, \tag{17}$$

where $\mathbf{U}_a$ is the model matrix (defined in (12)) of c_a, and $\mathbf{U}_i$ is the model matrix of c_i.

Therefore, from (13), (16), and (17), we have:

$$\begin{aligned} \hat{\eta}_a &= (\mathbf{U}_a^T\mathbf{U}_a)^{-1}\mathbf{U}_a^T\mathbf{y}_a = (\mathbf{U}_a^T\mathbf{U}_a)^{-1}\mathbf{U}_a^T \sum_{i=1}^{m} \mathbf{y}_i \\ &= \sum_{i=1}^{m} (\mathbf{U}_i^T\mathbf{U}_i)^{-1}\mathbf{U}_i^T\mathbf{y}_i = \sum_{i=1}^{m} \hat{\eta}_i. \end{aligned} \tag{18}$$

b) From (18), we see that $\theta_{i,j}$ terms only depend on $u_{i,j}$ terms. Since all cells have identical $u_{i,j}$ terms, we see that all cells also have identical $\theta_{i,j}$ terms, for $i, j = 0..k-1$. Therefore, we have:

$$\mathbf{\Theta}_a = q\mathbf{\Theta}_i, \quad i = 1..m. \tag{19}$$

□

An assumption for aggregation in standard dimensions is that each component cell has the same number of tuples. We have found that this assumption is satisfied in most applications. Moreover, since aggregation in a standard dimension implies that the component cells share a same region along the regression dimension, it is usually not meaningful to aggregate two cells with different numbers of tuples. For example, for the location dimension, we can aggregate eastern and western parts into the United States for a same month or year, but it is not meaningful to aggregate one month of the eastern part with two months of the western part.

4.4 Lossless Aggregation in Regression Dimensions

We now consider aggregation (roll-up) in regression dimensions. Still, we suppose that c_a is a cell aggregated from a number of component cells $c_1, \ldots, c_m$, which differ in one regression dimension, and that we want to compute the GMLR model of c_a.

In this roll-up, the set of tuples of c_a is defined to be the union of tuples of the component cells. More precisely, all component cells only differ in one regression dimension and do not necessarily have the same number of tuples. Suppose the component cell c_i has n_i tuples, and c_a has n_a tuples, then we have $n_a = \sum_{i=1}^{m} n_i$. Since c_a contains the union of the tuples from all component cells, we have:

$$\mathbf{y}_a = (\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_m)^T, \tag{20}$$

where $\mathbf{y}_a$ and $\mathbf{y}_j$ are vectors of measure attributes for corresponding cells, respectively. The following theorem shows how to aggregate NCRs along a regression dimension without accessing the raw data.

Theorem 4.4. (Lossless aggregation in regression dimensions). *For aggregation in a regression dimension, suppose $NCR(c_1) = (\hat{\eta}_1, \mathbf{\Theta}_1)$, $NCR(c_2) = (\hat{\eta}_2, \mathbf{\Theta}_2), \cdots, NCR(c_m) = (\hat{\eta}_m, \mathbf{\Theta}_m)$ are the NCRs for the m component cells, respectively, and suppose $NCR(c_a) = (\hat{\eta}_a, \mathbf{\Theta}_a)$ is the NCR for the aggregated cell, then $NCR(c_a)$ can be derived from that of the component cells using the following equations:*

a. $\hat{\eta}_a = \left(\sum_{i=1}^{m} \mathbf{\Theta}_i\right)^{-1}\left(\sum_{i=1}^{m} \mathbf{\Theta}_i\hat{\eta}_i\right)$ *and*
b. $\mathbf{\Theta}_a = \sum_{i=1}^{m} \mathbf{\Theta}_i$.

Proof. We will prove item b, first and then prove item a based on item b.

Let $\mathbf{U}_a$ be the model matrix of c_a and let $\mathbf{U}_i$ be the model matrix of c_i for $i = 1..m$, from (12), we see that $\mathbf{U}_a$ has n_a rows and $\mathbf{U}_i$ has n_i rows, respectively. Similar to $\mathbf{y}_a$, we can write $\mathbf{U}_a$ in matrix form as:

$$\mathbf{U}_a = (\mathbf{U}_1, \mathbf{U}_2, \cdots, \mathbf{U}_m)^T. \tag{21}$$

From the definition of matrix multiplication, and from the definition of $\mathbf{\Theta}$, we see that

$$\mathbf{\Theta}_a = \mathbf{U}_a^T\mathbf{U}_a \quad \text{and} \quad \mathbf{\Theta}_i = \mathbf{U}_i^T\mathbf{U}_i, \quad \text{for } i = 1\cdots m. \tag{22}$$

Therefore, from (21), we have:

$$\begin{aligned}\mathbf{\Theta}_a &= \begin{pmatrix}\mathbf{U}_1^T & \mathbf{U}_2^T & \cdots & \mathbf{U}_m^T\end{pmatrix}\begin{pmatrix}\mathbf{U}_1 & \mathbf{U}_2 & \cdots & \mathbf{U}_m\end{pmatrix}^T \\ &= \sum_{i=1}^{m} \mathbf{U}_i^T\mathbf{U}_i = \sum_{i=1}^{m} \mathbf{\Theta}_i.\end{aligned} \tag{23}$$

Thus, we proved item b. We now turn to prove item a. From (13), we know that $\hat{\eta}_i = (\mathbf{U}_i^T\mathbf{U}_i)^{-1}\mathbf{U}_i^T\mathbf{y}_i$, for $i = 1\cdots m$, therefore, we have

$$(\mathbf{U}_i^T\mathbf{U}_i)\hat{\eta}_i = \mathbf{U}_i^T\mathbf{y}_i, \quad \text{for } i = 1\cdots m. \tag{24}$$

Substituting (22) into (24) gets:

$$\mathbf{\Theta}_i\hat{\eta}_i = \mathbf{U}_i^T\mathbf{y}_i, \quad \text{for } i = 1\cdots m. \tag{25}$$

By combining (20), (21), and (25), we have:

$$\begin{aligned}\sum_{i=1}^{m} \mathbf{\Theta}_i\hat{\eta}_i &= \sum_{i=1}^{m} \mathbf{U}_i^T\mathbf{y}_i \\ &= (\mathbf{U}_1^T \ \mathbf{U}_2^T \ \cdots \ \mathbf{U}_m^T)(\mathbf{y}_1 \ \mathbf{y}_2 \ \cdots \ \mathbf{y}_m)^T \\ &= \mathbf{U}_a^T\mathbf{y}_a.\end{aligned} \tag{26}$$

From (22) and from item b of this theorem, we get:

$$\sum_{i=1}^{m} \mathbf{\Theta}_i = \mathbf{\Theta}_a = \mathbf{U}_a^T\mathbf{U}_a. \tag{27}$$

From (13), we have:

$$\hat{\eta}_a = (\mathbf{U}_a^T\mathbf{U}_a)^{-1}\mathbf{U}_a^T\mathbf{y}_a. \tag{28}$$

By substituting (26) and (27) into (28), we get:

$$\hat{\eta}_a = \left(\sum_{i=1}^{m} \mathbf{\Theta}_i\right)^{-1}\left(\sum_{i=1}^{m} \mathbf{\Theta}_i\hat{\eta}_i\right). \tag{29}$$

Thus, we proved item a and the proof is completed. □

4.5 Remarks

Theorems 4.3 and 4.4 show lossless aggregation properties of our NCR compression to support multidimensional regression analysis. For aggregations in the standard and regression dimensions, we see from Theorems 4.3 and 4.4 that all the parameters needed to compute the regression model of the aggregated cell can be obtained from the NCRs of the component cells. Also, Theorem 4.1 shows that the space complexity of NCR is low and is independent of the number of base tuples in each cell. These properties of the proposed NCR compression technique make aggregations in a data cube efficient and scalable.

Our method is general for multidimensional data cells, since any aggregation involving multiple dimensions can be decomposed into several single-dimension aggregations. Since each aggregation is lossless, the combined aggregation is still lossless. If we need to compute multiple regression models, it is true that we need to save the NCR auxiliary information for each regression model. For example, if we have both a linear regression and a quadratic regression model, then we need to keep NCRs for both of them.

Our work is applicable to multidimensional data cubes no matter if the hierarchies of data cubes are shallow or deep. Typically, in older applications, the basic unit of time is large and dimension hierarchies are shallow, while in many recent "monitoring" type of applications such as homeland security and power usage monitoring, the basic unit of time can be very small and the hierarchies are lengthy. Our theory is general for these data cubes with different characteristics.

The proposed technique can save computational time under incremental updates of the raw data. There are several cases of data updates. If all of the raw data are updated, then all the regression models need to be updated at the bottom level. If only a few records of the raw data are updated, which is more often, then we only need to recompute the cells that are modified and propagate the changes to higher level cells using the aggregation formulae. If new records of data are added into the data set, then we only need to recompute the new data cells and propagate the changes.

5 Multidimensional Filtering Analysis

Filtering is a popular technique that is widely used to describe the relation of one output, or system variable, to one or more inputs. Filters have ample applications in statistical analysis of time-series and stream data, such as weather prediction and financial market modeling. We extend our theory for GMLR to support multidimensional aggregation of filter measures in a multidimensional data cube.

5.1 Autoregressive Filters

Given an $n \times 1$ vector of time-series data $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$, the autoregressive filter models the dynamics of the the time-series using the following model [6]:

$$E(y_i) = \eta_1 y_{i-1} + \eta_2 y_{i-2} + \cdots + \eta_p y_{i-p}, \tag{30}$$

where $\eta = (\eta_1, \eta_2, \cdots, \eta_p)$ is a $p \times 1$ vector of *autoregression parameters*. We can see that the autoregressive filter is a model that predicts the future data using the p previous historical data, where $p > 1$ is the *number of lag steps* for the filter. For all the data points, the autoregression can be written as:

$$\begin{pmatrix} E(y_{p+1}) \\ E(y_{p+2}) \\ \cdots \\ E(y_n) \end{pmatrix} = \begin{pmatrix} y_1 & y_2 & \cdots & y_p \\ y_2 & y_3 & \cdots & y_{p+1} \\ . & . & . & . \\ . & . & . & . \\ y_{n-p} & u_{n-p-1} & \cdots & y_{n-1} \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \\ \cdots \\ \eta_p \end{pmatrix}. \tag{31}$$

The autoregression analysis can be viewed as a special case of GMLR regression discussed in the last section, with the following model matrix $\mathbf{U}_a$:

$$\mathbf{U}_a \begin{pmatrix} y_1 & y_2 & \cdots & y_p \\ y_2 & y_3 & \cdots & y_{p+1} \\ . & . & . & . \\ . & . & . & . \\ y_{n-p} & u_{n-p+1} & \cdots & y_{n-1} \end{pmatrix}. \tag{32}$$

We can now write the regression function in matrix form as $E(\mathbf{y}|\mathbf{U}_a) = \mathbf{U}_a \eta$. The ordinary least square(OLS) estimates of the autoregression parameters η is defined as:

$$\hat{\eta} = (\mathbf{U}_a^T \mathbf{U}_a)^{-1} \mathbf{U}_a^T \mathbf{y}. \tag{33}$$

We show that autoregression filter parameters are compressible measures. Like the regression analysis, we propose the following compressed representation of a data cell to support multidimensional aggregation of autoregresssion filter measures.

Definition 3. *For multidimensional online autoregression filtering analysis in data cubes, the Nonlinear Compression Representation* (NCR_a) *of a data cell* c *is defined as* $NCR_a(c) = \{\hat{\eta}_i | i = 1, \cdots, p\} \cup \{\theta_{ij} | i, j = 1, \cdots, p, i \le j\}$, *where* $\theta_{ij} = \sum_{h=1}^{n-p} y_{i+h-1} \cdot y_{j+h-1}$.

Intuitively, θ_{ij} is a compressed measure that records the sum of the autocorrelations among the variables with different distances defined by i and j.

In the matrix form, the elements in NCR_a can be arranged into two matrices: $\hat{\boldsymbol{\eta}}$ and $\boldsymbol{\Theta}$, where

$$\hat{\eta} = \begin{pmatrix} \hat{\eta}_1 \\ \hat{\eta}_2 \\ \cdots \\ \hat{\eta}_p \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Theta} = \begin{pmatrix} \theta_{11} & \theta_{11} & \theta_{12} & \cdots & \theta_{1,p} \\ \theta_{21} & \theta_{21} & \theta_{22} & \cdots & \theta_{2,p} \\ . & . & . & . & . \\ . & . & . & . & . \\ \theta_{p,1} & \theta_{p,2} & \theta_{p,2} & \cdots & \theta_{p,p} \end{pmatrix}.$$

Since Θ is symmetric, we only need to store the upper triangle of Θ in NCR_a. The size of NCR_a is $S(p) = p + \frac{p(p+1)}{2} = \frac{p^2+3p}{2}$, where p is the number of lag steps in the filter. Therefore, similar to NCRs for regression analysis, the size $S(p)$ of the NCR_a for autoregressive filtering analysis is quadratic in the number of lag steps p and is independent of the number of tuples in the data cell.

Since autoregressive filter is used to analyze time-series data, typically only aggregation in the regression dimension is performed to combine multiple time segments together. For aggregation in a regression dimension, suppose $NCR_a(c_1) = (\hat{\eta}_1, \Theta_1)$, $NCR_a(c_2) = (\hat{\eta}_2, \Theta_2), \cdots, NCR_a(c_m) = (\hat{\eta}_m, \Theta_m)$ are the NCR_as for the m component cells, respectively, and suppose $NCR_a(c_a) = (\hat{\eta}_a, \Theta_a)$ is the NCR_a for the aggregated cell, then $NCR_a(c_a)$ can be derived from that of the component cells using the following equations:

$$\hat{\eta}_a = \left(\sum_{i=1}^{m} \boldsymbol{\Theta}_i \right)^{-1} \left(\sum_{i=1}^{m} \boldsymbol{\Theta}_i \hat{\eta}_i \right) \quad \text{and} \quad \boldsymbol{\Theta}_a = \sum_{i=1}^{m} \boldsymbol{\Theta}_i \ . \tag{34}$$

The correctness of the above lossless aggregation formula can be proved as a corollary of Theorem 4.5.

5.2 Linear Prediction Filters

A general linear prediction filter [33] usually includes two parts, one for an autoregressive model, and one for moving average of input values. Suppose we have n tuples in a cell: $(I_i, y_i), i = 1 \ldots n$, where each of $I_i, i = 1 \ldots n$ is an *input variable* at step i, and each scalar y_i is the measure attribute of the ith tuple. A linear prediction filter defines the

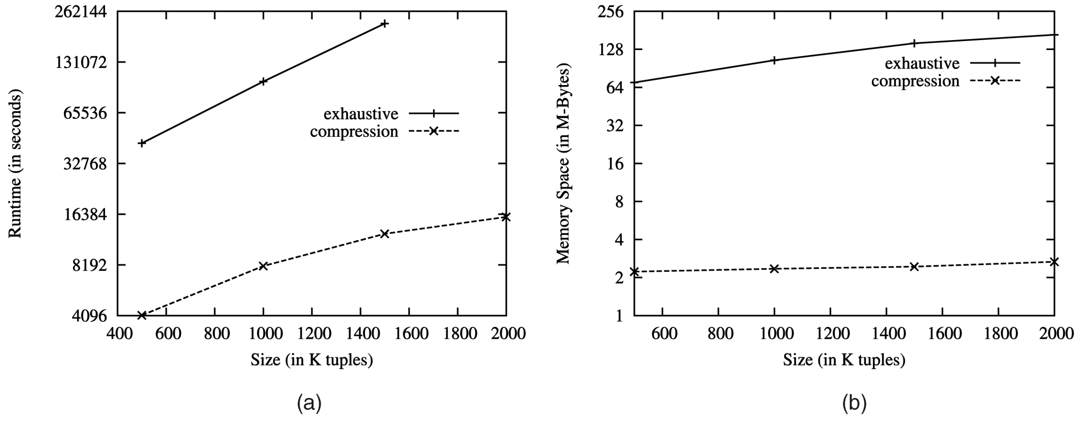


Fig. 3. Time and space versus the number of tuples for data sets $D3L3C5$. (a) Time versus size. (b) Space versus size.

following model relating the output measure and input variables:

$$E(y_i) = \sum_{j=1}^{p} \eta_j y_{i-j} + \sum_{k=1}^{m} \alpha_k I_{i-k}, \tag{35}$$

where $\eta = (\eta_1, \eta_2, \cdots, \eta_p)^T$ is a $p \times 1$ vector, and $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_m)^T$ is an $m \times 1$ vector. (η, α) is the parameter measures for a linear prediction filter.

The linear prediction filter is a model that predicts the future data using the autoregression of p previous historical data and the moving average of m previous input data. Typically, the autoregressive part models the internal dynamics of a systems and the moving average part models the dynamics of the interaction between a system and its external inputs.

The ordinary least square (OLS) estimates of linear prediction parameters (η, α) is defined as:

$$\begin{pmatrix} \hat{\eta} \\ \hat{\alpha} \end{pmatrix} = (\mathbf{U}_L^T \mathbf{U}_L)^{-1} \mathbf{U}_L^T \mathbf{y}, \tag{36}$$

where

$$\mathbf{U}_L = \begin{pmatrix} y_1 & \cdots & y_p & I_1 & \cdots & I_m \\ y_2 & \cdots & y_{p+1} & I_2 & \cdots & I_{m+1} \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & & & \\ y_{n-p} & \cdots & y_{n-1} & I_{n-p} & \cdots & I_{n-p+m-1} \end{pmatrix}.$$

Fitting into the GMLR regression framework, the linear prediction filter parameters are also compressible measures under the following compression:

Definition 4. *For multidimensional online linear prediction filter analysis in data cubes, the Nonlinear Compression Representation* (NCR_L) *of a data cell c is*

$$NCR_L(c) = \{\hat{\eta}_i | i = 1, \cdots, p\} \cup \{\hat{\alpha}_i | i = 1, \cdots, m\} \cup \{\theta_{ij} | i, j = 1, \cdots, p+m, i \leq j\},$$

where $\theta_{ij} = \sum_{h=1}^{n-p} \mathbf{U}_{L_{h,i}} \cdot \mathbf{U}_{L_{h,j}}$.

The size of NCR_L is only quadratic to p and m: $S(p,m) = (p^2 + m^2 + 2pm + 3p + 3m)/2$ and is independent of the number of tuples in the data cell. The lossless aggregation formula along the regression dimension can be derived straightforwardly from Theorem 4.4.

6 Performance Study

To evaluate the effectiveness and efficiency of the proposed NCR compression technique, we perform a performance study on synthetic data sets. Our results show that the memory and time taken by the proposed algorithms are small in comparison to the exhaustive methods.

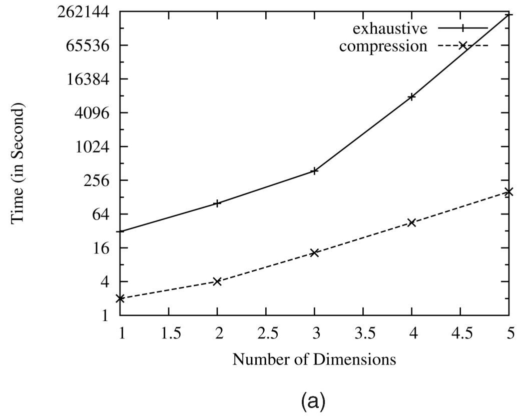
In our experiments, we need to generate Θ and η for the NCRs. For a cell with n tuples, the time complexity is $O(n)$ to compute θ_{ij} using (15), $O(nk^2)$ to generate Θ, and $O(nk)$ to generate η. The overall time complexity to compute the NCR for a cell is $O(nk^2)$. There are k^2 numbers in Θ and k elements in η, where k, the number of regression terms, is a small constant independent of n. In our experiments, we have used linear regression where $k = 2$. Increasing k will only increase the complexity by a constant factor.

In our experiments, we use synthetic data sets generated by a data generator similar in spirit to the IBM data generator [3] designed for testing data mining algorithms. The convention for the data sets is as follows: $D3L3C10T4000K$ means there are three dimensions, each dimension contains three levels, the node fan-out factor (cardinality) is 10 (i.e., 10 children per node), and there are in total 4000K tuples in the lowest level. All experiments were conducted on a 1.0GHz AMD PC with $1G$ megabytes memory, running Microsoft Windows-2000 Server.

We compare the proposed compression-based regression cube computation technique with the exhaustive method in which the regression models are generated from scratch for all cuboids. The performance results of data cubing (cube computation) are reported in Figs. 3, 4, and 5.

Fig. 3 shows the processing time and memory usage for the two methods to generate regression measures for all data cells in data cubes with increasing size, where the size is measured as the number of tuples at the lowest level. Fig. 4 shows the time and memory usage for the two methods with increasing number of dimensions, and Fig. 5 shows the results with varying number of levels for each dimension.

Note that our compression technique saves the time to compute η for all the cells. For data $D3L3C5$ with linear



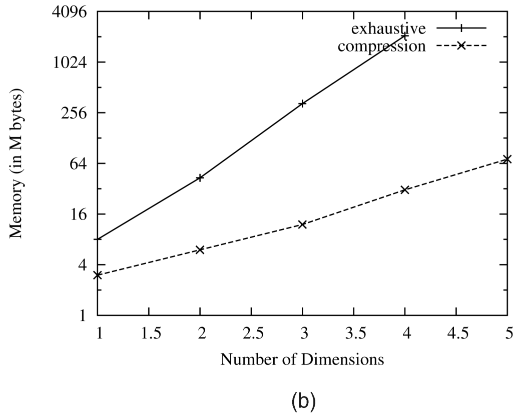
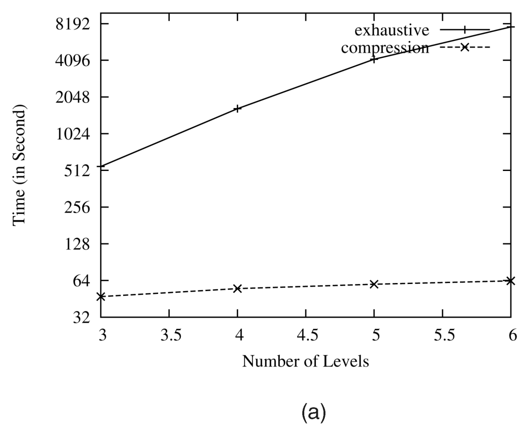


Fig. 4. Time and space versus the number of dimensions for data sets $L2C5T4000K$. (a) Time versus the number of dimensions. (b) Space versus the number of dimensions.



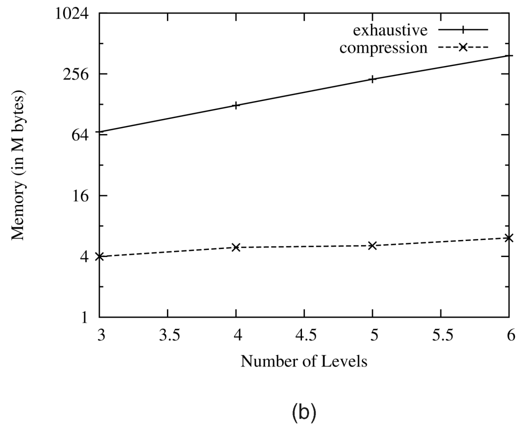


Fig. 5. Time and space versus the number of hierarchy levels for data sets $D3C5T50K$. (a) Time versus the number of levels. (b) Space versus the number of levels.

regression, we need to generate 54,692 η values for the 27,346 data cells in the data cube. For the exhaustive method, the time complexity to compute η is $O(n)$ which increases as the dimension levels go up. Using our compression technique, the time complexity to compute η is significantly reduced for high-level cells as we can reuse the results of descendent cells.

We can see that, in all the cases, the proposed compressed regression cube is significantly more efficient in time and space than the exhaustive method without using compression. Note that the CPU time and space usage on the vertical axes in these figures are in a logarithmic scale, so that a small difference along the vertical axes typically represents a large difference in running time or space usage. The techniques in Section 5 will have similar performance as the regression models in Section 4, since they use similar compression and aggregation techniques.

Our work can also be extended to handle massive stream data. In practice, stream data are generated in real time and one can only scan the raw data once. Also, the data cube space is usually so huge that it is impossible to materialize all the cuboids. We have studied the performance of the regression cube in a more sophisticated and more realistic setting in which the data cuboids are partially and selectively materialized. Our implementation is based on the Stream Cube architecture [18], a general and efficient framework for online processing of large-scale stream data. Our experimental study showed that compression-based regression cubing can significantly improve the efficiency for computing multidimensional, multilevel regression cubes for stream data in the Stream Cube environment.

7 Discussion of Related Work

In this section, we compare our study with some related work and point out the differences from our work.

In 2002, we proposed to support simple linear regression in a multidimensional space [9] by compressing each cuboid to four arguments to support lossless aggregation of linear regression models. In this paper, we have generalized the concept of regression cubes and extended the compression technique to the general GMLR and filtering analysis.

A highly related work to ours is the tool of *prediction cubes* proposed in 2005 [8], which supports OLAP of prediction models including probability-based ensemble, naive Bayesian classifier, and kernel-density classifier. The prediction cubes bear similar ideas as regression cubes in that both of them aim at deriving high-level models from lower-level models instead of accessing the raw data and rebuilding the models from scratch. A key difference is that, the prediction cube only supports models that are distributively decomposable or algebraically decomposable (i.e., the models are distributive or algebraic measures) [8], whereas the regression models in our study are not distributively or algebraically decomposable. We have overcome the mathematical difficulty with regression models by venturing the concept of compressible measures and developing lossless compression techniques for regression models. Also, the theory developed in prediction cubes deals with the prediction accuracy of nonparametric statistical models such as naive Bayesian classifiers, whereas our compression theory is developed for parameter reconstruction of parametric models such as regression models and filters.

Our paper considers efficient aggregation operations without accessing the raw data. Palpanas et al. [29] have considered the reverse problem, which is to derive original raw data from the aggregates. An approximative estimation algorithm based on maximum information entropy is proposed [29]. It will be interesting to study the interactions of these two complimentary approaches. We believe that using regression cubes will further improve the quality of data reconstruction, and estimated raw data can lead to enhanced statistical models at high levels.

Statistical time series analysis has been extensively studied [10]. A common assumption of these studies is that users are responsible for choosing the time series to be analyzed, including the scope of the object of the time series and the level of granularity. These studies and tools (including longitudinal studies [11]) do not provide the capabilities of relating the time series to the associated multidimensional multilevel characteristics, and they do not provide adequate support for online analytical processing and mining of the time series. In contrast, the framework established in this paper provides efficient support to help users form, select, analyze, and mine time series in a multidimensional and multilevel manner.

Similarity search and efficient retrieval of time series has been the major focus for time series-related research in the database community [25], [28]. Previous data mining research also paid attention to time series data, including shape-based patterns [2], representative trends [19], periodicity [24], and using time warping for data mining [26], management and querying of stream data [4], [13], [14], [16], and data mining (classification and clustering) on stream data [17], [22]. These works do not relate the multidimensional, multilevel characteristics with time series and stream data and do not seriously consider the aggregation of statistical measures.

Dimension hierarchies, cubes, and cube operations are formally introduced by Vassiliadis [32]. Lenz and Thalheim [27] proposed to classify OLAP aggregation functions into distributive, algebraic, and holistic ones. In data warehousing and OLAP, much progress has been made on the efficient support of standard and advanced OLAP queries in data cubes, including selective cube materialization [21], iceberg cubing [5], [20], cube gradients analysis [12], [23], exception [30], and intelligent roll-up [31]. However, the measures studied in OLAP systems are usually single values, and previous studies do not consider the support for regression analysis. In contrast, our work studies OLAP of complex regression measures in data cubes.

In statistics, regression and filtering are parametric models with fixed functions. In practice, parametric models are most useful when the users have prior domain knowledge of the applications and know how to choose the functions. For applications where users have no knowledge about the data, nonparametric models are preferable. We are currently working on supporting nonparametric models in the regression cube environment and will report the results in a later publication.

8 Conclusions

In this paper, we have promoted online analytical processing of advanced statistical measures in multidimensional data cubes, and proposed a general theory for efficiently compressing and aggregating the regression and filtering measures. We have developed the NCR compression technique for aggregations of linear and nonlinear regression parameters in data cubes, so that only a small number of numerical values instead of the complete raw data need to be registered for multidimensional regression analysis. Lossless aggregation formulae are derived based on the compressed NCR representation. The aggregation is efficient in terms of time and space complexities. We have also extended the results to filtering analysis of time-series data.

We believe that this study is the first one that explores online regression analysis of multidimensional data. There are a lot of issues to be explored further. For example, we can extend the method to support other statistical models that may bring new computational power and user flexibility to online multidimensional statistical analysis. Moreover, the results developed here are confined to numerical data. It is an interesting topic to study the extension to other regression models, such as logistic regression, that can be applied to categorical data. Finally, we believe that an important direction is to develop data mining methods to utilize the advanced statistical measures provided by the regression cubes.

Acknowledgments

The authors would like to thank the reviewers for their comments to improve this paper. The work is supported by research grants from the Washington University in St. Louis, the US Department of Energy, US National Science Foundation, the University of Illinois, and Microsoft Research.

References

[1] S. Agarwal, R. Agrawal, P.M. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan, and S. Sarawagi, "On the Computation of Multidimensional Aggregates," *Proc. Int'l Conf. Very Large Data Bases (VLDB '96)*, pp. 506-521, Sept. 1996.

[2] R. Agrawal, G. Psaila, E.L. Wimmers, and M. Zait, "Querying Shapes of Histories," *Proc. Int'l Conf. Very Large Data Bases (VLDB '95)*, pp. 502-514, Sept. 1995.

[3] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. Int'l Conf. Data Eng. (ICDE '95)*, pp. 3-14, Mar. 1995.

[4] S. Babu and J. Widom, "Continuous Queries Over Data Streams," *SIGMOD Record*, vol. 30, pp. 109-120, 2001.

[5] K. Beyer and R. Ramakrishnan, "Bottom-Up Computation of Sparse and Iceberg Cubes," *Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '99)*, pp. 359-370, June 1999.

[6] G. Box and F.M. Jenkins, *Time Series Analysis: Forecasting and Control*, second ed. Holden-Day, 1976.

[7] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology," *SIGMOD Record*, vol. 26, pp. 65-74, 1997.

[8] B.C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan, "Prediction Cubes," *Proc. Very Large Data Bases Conf.*, 2005.

[9] Y. Chen, G. Dong, J. Han, B.W. Wah, and J. Wang, "Multi-dimensional Regression Analysis of Time-Series Data Streams," *Proc. Very Large Data Bases Conf.*, 2002.

[10] D. Cook and S. Weisberg, *Applied Regression Including Computing and Graphics.* John Wiley, 1999.

[11] P. Diggle, K. Liang, and S. Zeger, *Analysis of Longitudinal Data.* Oxford Science Publications, 1994.

[12] G. Dong, J. Han, J. Lam, J. Pei, and K. Wang, "Mining Multi-Dimensional Constrained Gradients in Data Cubes," *Proc. Int'l Conf. Very Large Data Bases (VLDB '01)*, pp. 321-330, Sept. 2001.

[13] J. Gehrke, F. Korn, and D. Srivastava, "On Computing Correlated Aggregates over Continuous Data Streams," *Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '01)*, pp. 13-24, May 2001.

[14] A.C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss, "Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries," *Proc. Int'l Conf. Very Large Data Bases (VLDB '01)*, pp. 79-88, Sept. 2001.

[15] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals," *Data Mining and Knowledge Discovery*, vol. 1, pp. 29-54, 1997.

[16] M. Greenwald and S. Khanna, "Space-Efficient Online Computation of Quantile Summaries," *Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '01)*, pp. 58-66, May 2001.

[17] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams," *Proc. Symp. Foundations of Computer Science (FOCS '00)*, pp. 359-366, 2000.

[18] J. Han, Y. Chen, G. Dong, J. Pei, B.W. Wah, J. Wang, and D. Cai, "Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams," *Distributed and Parallel Databases J.*, 2005.

[19] J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," *Proc. Int'l Conf. Data Eng. (ICDE '99)*, pp. 106-115, Apr. 1999.

[20] J. Han, J. Pei, G. Dong, and K. Wang, "Efficient Computation of Iceberg Cubes With Complex Measures," *Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '01)*, pp. 1-12, May 2001.

[21] V. Harinarayan, A. Rajaraman, and J.D. Ullman, "Implementing Data Cubes Efficiently," *Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '96)*, pp. 205-216, June 1996.

[22] G. Hulten, L. Spencer, and P. Domingos, "Mining Time-Changing Data Streams," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery in Databases (KDD '01)*, Aug. 2001.

[23] T. Imielinski, L. Khachiyan, and A. Abdulghani, "Cubegrades: Generalizing Association Rules," technical report, Dept. of Computer Science, Rutgers Univ., Aug. 2000.

[24] P. Indyk, N. Koudas, and S. Muthukrishnan, "Identifying Representative Trends in Massive Time Series Data Sets Using Sketches," *Proc. Int'l Conf. Very Large Data Bases (VLDB '00)*, pp. 363-372, Sept. 2000.

[25] T. Kahveci and A.K. Singh, "Variable Length Queries for Time Series Data," *Proc. Int'l Conf. Data Eng. (ICDE '01)*, Mar. 2001.

[26] E.J. Keogh and M.J. Pazzani, "Scaling Up Dynamic Time Warping to Massive Dataset," *Proc. European Symp. Principle of Data Mining and Knowledge Discovery (PKDD '99)*, pp. 1-11, Sept. 1999.

[27] H. Lenz and B. Thalheim, "OLAP Databases and Aggregation Functions," *Proc. 13th Int'l Conf. Scientific and Statistical Database Management*, pp. 91-100, 2001.

[28] Y.-S. Moon, K.-Y. Whang, and W.-K. Loh, "Duality-Based Subsequence Matching in Time-Series Databases," *Proc. Int'l Conf. Data Eng. (ICDE '01)*, pp. 263-272, Apr. 2001.

[29] T. Palpanas, N. Koudas, and A.O. Mendelzon, "Using Datacube Aggregates for Approximate Querying and Deviation Detection," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 11, pp. 1465-1477, Nov. 2005.

[30] S. Sarawagi, R. Agrawal, and N. Megiddo, "Discovery-Driven Exploration of OLAP Data Cubes," *Proc. Int'l Conf. Extending Database Technology (EDBT '98)*, pp. 168-182, Mar. 1998.

[31] G. Sathe and S. Sarawagi, "Intelligent Rollups in Multidimensional OLAP Data," *Proc. Int'l Conf. Very Large Data Bases (VLDB '01)*, pp. 531-540, Sept. 2001.

[32] P. Vassiliadis, "Modeling Multidimensional Databases, Cubes and Cube Operations," *Proc. 10th Int'l Conf. Scientific and Statistical Database Management*, pp. 53-62, 1998.

[33] A.B. Williams and F.J. Taylor, *Electronic Filter Design Handbook.* McGraw-Hill, 1995.

[34] Y. Zhao, P.M. Deshpande, and J.F. Naughton, "An Array-Based Algorithm for Simultaneous Multidimensional Aggregates," *Proc. ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '97)*, pp. 159-170, May 1997.

Yixin Chen received the PhD degree in computing science from the University of Illinois at Urbana-Champaign in 2005. He is an assistant professor of computer science at the Washington University in St Louis, Missouri. His research interests including nonlinear optimization, constrained search, planning and scheduling, data mining, and data warehousing. His work on constraint partitioning and planning has won First Prizes in optimal and satisficing tracks in the International Planning Competitions (2004 and 2006) and the Best Paper Award at the International Conference on Tools for AI (2005). His work on data clustering has won the Best Paper Award at the International Conference on Machine Learning and Cybernetics (2004) and the Best Paper nomination at the International Conference on Intelligent Agent Technology (2004). He is partially funded by a Early Career Principal Investigator Award (2006) from the US Department of Energy.

Guozhu Dong received the PhD degree from the University of Southern California in 1988. He is a professor at Wright State University. His main research interests are databases, data mining, and bioinformatics. He has published around 100 articles in journals and conferences, holds three US patents, and was a recipient of the Best Paper Award from ICDM 2005. Representative research includes results on first-order incremental maintenance of transitive closure and shortest paths, expressive power of first-order queries with arithmetic constraints, mining of emerging patterns and their use for classification and bioinformatics, and iceberg cubing, gradient cubing, regression cubing, and bound-prune cubing for OLAP. He has served on the program committee of ICDE, ICDM, ICDT, ACM PODS, ACM SIGKDD, and VLDB. He was a program commitee cochair of the International Conference on Web-Age Information Management 2003, is on the steering committee of the same conference series, is on the editorial board of the *International Journal of Information Technology*, and will serve as a program cochair of the Joint International Conference of APWeb and WAIM in 2007. His research has been funded by US NSF, ARC, AFRL, and private corporations.

Jiawei Han is a professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He has been working on research into data mining, data warehousing, stream data mining, spatiotemporal and multimedia data mining, biological data mining, social network analysis, text and Web mining, and software bug mining, with more than 300 conference and journal publications. He has chaired or served in many program committees of international conferences and workshops. He also served or is serving on the editorial boards for *Data Mining and Knowledge Discovery*, the *IEEE Transactions on Knowledge and Data Engineering*, the *Journal of Computer Science and Technology*, and the *Journal of Intelligent Information Systems*. He is founding editor-in-chief of the *ACM Transactions on Knowledge Discovery from Data* (*TKDD*), and on the board of directors for the Executive Committee of ACM SIGKDD. He has received ACM SIGKDD Innovation Award (2004) and IEEE Computer Society Technical Achievement Award (2005). He is an ACM Fellow (2004).

Jian Pei received the PhD degree in computing science from Simon Fraser University in 2002. He is an assistant professor of computing science at Simon Fraser University. His research focuses on effective and efficient data analysis techniques for novel data intensive applications. His current research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the US National Science Foundation (NSF), Hewlett-Packard Company (HP), and the Canadian Imperial Bank of Commerce (CIBC). He has published prolifically in refereed journals, conferences, and workshops, has served regularly in the organization committees and the program committees of many international conferences and workshops, and has been a reviewer for the leading academic journals in his fields. He is a member of the ACM.

Benjamin W. Wah received the PhD degree in computer science from the University of California, Berkeley, in 1979. He is currently the Franklin W. Woeltge Endowed Professor of Electrical and Computer Engineering and a professor of the Coordinated Science Laboratory of the University of Illinois at Urbana-Champaign, Urbana, IL. Previously, he had served on the faculty of Purdue University (1979-1985), as a program director at the US National Science Foundation (1988-89), as Fujitsu Visiting Chair Professor of Intelligence Engineering, University of Tokyo (1992), and McKay Visiting Professor of Electrical Engineering and Computer Science, University of California, Berkeley (1994). In 1989, he was awarded a university scholar at the University of Illinois; in 1998, he received the IEEE Computer Society Technical Achievement Award; in 2000, the IEEE Millennium Medal; in 2003, the Raymond T. Yeh Lifetime Achievement Award from the Society for Design and Process Science; and in 2006, the IEEE Computer Society W. Wallace-McDowell Award and the Pan Wen-Yuan Outstanding Research Award. Dr. Wah's current research interests are in the areas of nonlinear search and optimization, multimedia signal processing, and computer networks. He cofounded the *IEEE Transactions on Knowledge and Data Engineering* in 1988 and served as its editor-in-chief between 1993 and 1996, and is the honorary editor-in-chief of *Knowledge and Information Systems*. He currently serves on the editorial boards of *Information Sciences*, the *International Journal on Artificial Intelligence Tools*, the *Journal of VLSI Signal Processing*, *World Wide Web*, and *Neural Processing Letters*. He had chaired a number of international conferences, including the 2000 IFIP World Congress and the 2006 IEEE/WIC/ACM International Conferences on Data Mining and Intelligent Agent Technology. He has served the IEEE Computer Society in various capacities, including as vice president of Publications (1998 and 1999) and president (2001). He is a fellow of the AAAS, the ACM, and the IEEE.

Jianyong Wang received the PhD degree in computer science in 1999 from the Institute of Computing Technology, the Chinese Academy of Sciences. Since then, he has worked as an assistant professor in the Department of Computer Science and Technology, Peking University, and visited the School of Computing Science at Simon Fraser University, the Department of Computer Science at the University of Illinois at Urbana-Champaign, and the Department of Computer Science and Engineering at the University of Minnesota, working in data mining. He is currently an associate professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He is a member of the IEEE Computer Society and the ACM SIGKDD.