The VLDB Journal (2000) 8: 254–266

The VLDB Journal

Quantifiable data mining using ratio rules

Flip Korn[1], **Alexandros Labrinidis**[2], **Yannis Kotidis**[2], **Christos Faloutsos**[3]

[1] AT&T Labs - Research, Florham Park, NJ 07932, USA; E-mail: flip@research.att.com
[2] University of Maryland, College Park, MD 20742, USA; E-mail: {labrinid,kotidis}@cs.umd.edu
[3] Carnegie Mellon University, Pittsburgh, PA 15213, USA; E-mail: christos@cs.cmu.edu

Edited by J. Widom. Received: March 15, 1999 / Accepted: November 1, 1999

Abstract. Association Rule Mining algorithms operate on a data matrix (e.g., customers $\times$ products) to derive association rules [AIS93b, SA96]. We propose a new paradigm, namely, *Ratio Rules*, which are quantifiable in that we can measure the "goodness" of a set of discovered rules. We also propose the "guessing error" as a measure of the "goodness", that is, the root-mean-square error of the reconstructed values of the cells of the given matrix, when we pretend that they are unknown. Another contribution is a novel method to guess missing/hidden values from the Ratio Rules that our method derives. For example, if somebody bought \$10 of milk and \$3 of bread, our rules can "guess" the amount spent on butter. Thus, unlike association rules, Ratio Rules can perform a variety of important tasks such as forecasting, answering "what-if" scenarios, detecting outliers, and visualizing the data. Moreover, we show that we can compute Ratio Rules in a *single* pass over the data set with small memory requirements (a few small matrices), in contrast to association rule mining methods which require multiple passes and/or large memory. Experiments on several real data sets (e.g., basketball and baseball statistics, biological data) demonstrate that the proposed method: (a) leads to rules that make sense; (b) can find large itemsets in binary matrices, even in the presence of noise; and (c) consistently achieves a "guessing error" of up to 5 times less than using straightforward column averages.

Key words: Data mining – Forecasting – Knowledge discovery – Guessing error

1 Introduction

Data mining has received considerable interest [FU96], of which the quintessential problem in database research has been association rule mining [AIS93b]. Given a data matrix with, for example, customers for rows and products for columns, association rule algorithms find rules that describe frequently co-occurring products, and are of the form

$$\{bread, milk\} \Rightarrow butter \quad (90\%),$$

meaning that customers who buy "bread" and "milk" also tend to buy "butter" with 90% confidence. What distinguishes database work from that of artificial intelligence, machine learning, and statistics is its emphasis on large data sets. The initial association rule mining paper [AIS93b], as well as all the follow-up database work [AS94], proposed algorithms to minimize the time to extract these rules through clever record-keeping to avoid additional passes over the data set.

The major innovation of this work is the introduction of *Ratio Rules* of the form

$$\textit{Customers typically spend } 1:2:5 \textit{ dollars } on \textit{ bread} : \textit{milk} : \textit{butter}.$$

The above example of a rule attempts to characterize purchasing activity: 'if a customer spends \$1 on bread, then s/he is likely to spend \$2 on milk and \$5 on butter'. What is also novel about this work is that, in addition to proposing a fully automated and user-friendly form of quantitative rules, it attempts to assess *how good* the derived rules are, an issue that has not been addressed in the database literature. We propose the "guessing error" as a measure of the "goodness" of a given set of rules for a given data set. The idea is to pretend that a data value (or values) is (are) "hidden", and to estimate the missing value(s) using the derived rules; the root-mean-square guessing error (averaged over all the hidden values) indicates how good the set of rules is. Towards this end, we provide novel algorithms for accurately estimating the missing values, even when multiple values are simultaneously missing.

As a result, Ratio Rules can determine (or recover) unknown (equivalently, hidden, missing, or corrupted) data, and can thus support the following applications:

This material is based upon work supported by the National Science Foundation under Grants No. IRI-9625428, DMS-9873442, IIS-9817496, and IIS-9910606, and by the Defense Advanced Research Projects Agency under Contract No. N66001-97-C-8517. Additional funding was provided by donations from NEC and Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, DARPA, or other funding parties.

– Data cleaning: reconstructing lost data and repairing noisy, damaged, or incorrect data (perhaps as a result of consolidating data from many heterogeneous sources for use in a data warehouse).
– Forecasting: '*If a customer spends £1 on bread and £2.50 on ham, how much will s/he spend on mayonnaise?*'
– "What-if" scenarios: '*We expect the demand for Cheerios to double; how much milk should we stock up on?*'[1]
– Outlier detection: '*Which customers deviate from the typical sales pattern?*'
– Visualization: Each Ratio Rule effectively corresponds to an eigenvector of the data matrix, as we discuss later. We can project the data points on the 2- or 3-D hyperplane defined by the first 2 or 3 Ratio Rules, and plot the result, to reveal the structure of the data set (*e.g.*, clusters, linear correlations, *etc.*).

This paper is organized as follows: Section 2 reviews the related work. Section 3 defines the problem more formally. Section 4 introduces the proposed method. Section 5 presents the results from experiments. Section 6 provides a discussion. Finally, Sect. 7 lists some conclusions and gives pointers to future work.

2 Related work

[AIS93a] distinguish between three data mining problems: identifying classifications, finding sequential patterns, and discovering association rules. We review only material relevant to the latter, since it is the focus of this paper. See [CHY96] for an excellent survey of all three problems.

The seminal work of [AIS93b] introduced the problem of discovering association rules and presented an efficient algorithm for mining them. Since then, new serial algorithms [AS94, PCY95, SON95] and parallel algorithms [AS96] have been proposed. In addition, generalized association rules have been the subject of recent work [SA95, HF95].

The vast majority of association rule discovery techniques are Boolean, since they discard the quantities of the items bought and only pay attention to whether something was bought or not. A notable exception is the work of [SA96], where they address the problem of mining quantitative association rules. Their approach is to partition each quantitative attribute into a set of intervals which may overlap, and to apply techniques for mining Boolean association rules. In this framework, they aim for rules such as

$$\textit{bread} : [3-5] \;\; \textit{and} \;\; \textit{milk} : [1-2] \Rightarrow \textit{butter} : [1.5-2] \;\; (90\%)$$

The above rule says that customers that spend between 3–5 dollars on bread and 1–2 dollars on milk, tend to spend 1.5–2 dollars on butter with 90% confidence.

Traditional criteria for selecting association rules are based on the support-confidence framework [AIS93b]; recent alternative criteria include the chi-square test [BMS97a] and probability-based measures [ST96]. Related issues include outlier detection and forecasting. See [Jol86] for a textbook treatment of both, and [AAR96, Hou96, BMS97b] for recent developments.

[1] We can also use Ratio Rules for "what-if" scenarios on the aggregate level, e.g., given the rule *Cheerios* : *milk* = 1 : 1 and the scenario that '*We expect the demand for Cheerios to double for the average customer; how much milk should we stock up on?*', we get that '*The demand for milk should double.*'

3 Problem definition

The problem we tackle is as follows. We are given a large set of N customers and M products organized in an $N \times M$ matrix $\mathbf{X}$, where each row corresponds to a customer transaction (for example, market basket purchase), and entry x_{ij} gives the dollar amount spent by customer i on product j. To make our discussion more concrete, we will use rows and "customers" interchangeably, and columns and "products" interchangeably[2]. The goal is to find Ratio Rules of the form $v_1 : v_2 : \cdots : v_M$ such that the rules can be used to accurately estimate missing values ("holes") when one or many *simultaneous* holes exist in any given row of the matrix. As previously mentioned, this will enable Ratio Rules to support the applications enumerated in Sect. 1.

Next, we give more intuition behind Ratio Rules and discuss a method for computing them efficiently.

4 Proposed method

The proposed method detects Ratio Rules using *eigensystem analysis*, a powerful tool that has been used for several settings, and is similar to Singular Value Decomposition, SVD, [PTVF92], Principal Component Analysis, PCA, [Jol86], Latent Semantic Indexing, LSI, [FD92], and the Karhunen-Loeve Transform, KLT, [DH73]. Eigensystem analysis involves computing the eigenvectors and eigenvalues of the covariance matrix of the given data points (see Sect. 4.1 for intuition and Sect. 4.2 for more formal treatment). In Sect. 4.3, we present an efficient, *single-pass* algorithm to compute the k best Ratio Rules. A fast algorithm is extremely important for database applications, where we expect matrices with several thousands or millions of rows. Section 4.4 presents one of the major contributions of this paper: the introduction of a measure for the "goodness" of a given set of rules. Section 4.5 presents another major contribution: how to use the Ratio Rules to predict missing values.

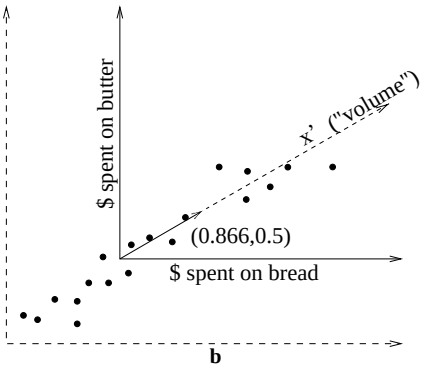
4.1 Intuition behind Ratio Rules

Figure 1a lists a set of N customers and M products organized in an $N \times M$ matrix. Each row vector of the matrix can be thought of as an M-dimensional point. Given this set of N points, eigensystem analysis identifies the axes (orthogonal directions) of greatest variance, after centering the points about the origin. Figure 1b illustrates an example of an axis that this analysis finds. In Fig. 1 we have M=2 products, and so our customers can be represented by 2-D points. The direction x' suggested by this analysis is shown, and its

[2] Of course, the proposed method is applicable to any $N \times M$ matrix, with a variety of interpretations for the rows and columns, e.g, patients and medical test measurements (blood pressure, body weight, etc.); documents and terms, typical in IR [SM83], etc

customer	*bread* ($)	*butter* ($)
Billie	.89	.49
Charlie	3.34	1.85
Ella	5.00	3.09
...	...	...
John	1.78	.99
Miles	4.02	2.61

a



b

Fig. 1. A data matrix **a** in table form and **b** its counterpart in graphical form, after centering (original axis drawn with dotted lines). As the graph illustrates, eigensystem analysis identifies the vector $(0.866, 0.5)$ as the "best" axis to project along

Table 1. Symbols and their definitions

symbol	*definition*
N	number of records (matrix rows)
M	number of attributes (matrix columns)
k	number of Ratio Rules retained
$\mathscr{R}$	set of Ratio Rules
$\mathscr{H}$	set of hidden values ("holes") in a given row
h	number of hidden values ("holes")
GE_1	guessing error in reconstructing one hole using $\mathscr{R}$
GE_h	guessing error in reconstructing h simultaneous holes using $\mathscr{R}$
$\mathbf{X}$	$N \times M$ data matrix
$\mathbf{X}_c$	column-centered version of $\mathbf{X}$
$\mathbf{X}^t$	transpose of $\mathbf{X}$
$x_{i,j}$	value at row i and column j of the matrix $\mathbf{X}$
$\hat{x}_{i,j}$	reconstructed (estimated) value at row i and column j
$\bar{x}$	mean cell value of $\mathbf{X}$
$\mathbf{C}$	$M \times M$ covariance matrix ($\mathbf{X}_c^t \times \mathbf{X}_c$)
$\mathbf{V}$	$M \times r$ day-to-concept similarity matrix
$\mathbf{U}$	$N \times r$ customer-to-concept similarity matrix
Λ	$r \times r$ eigenvalue matrix

meaning is that, if we are allowed only one rule, the best direction to project on is the direction of x'. The direction x' is a *Ratio Rule* (RR) that governs the correlations between money spent on the products, based on the customer purchasing activity in the matrix. In this case, the projection of a data point on the x' axis gives the overall "volume" of the purchase. The coordinates of the first RR = (0.866, 0.5) are those of the unit vector in the direction x'. They imply the rule "*bread* : *butter* $\Rightarrow$ \$0.866 : \$0.5"; that is, for most of our customers, the relative spendings bread-to-butter are close to the ratio 0.866:0.5. As we shall discuss later, these Ratio Rules can be used for forecasting, "what-if" scenarios, outlier detection, and visualization. In addition, they are often amenable to interpretation as underlying factors that describe, in this case, purchasing behavior.

4.2 Formal treatment of Ratio Rules

We shall use the following notational conventions from linear algebra:

- Bold capital letters denote matrices, for e.g., **U**, **X**.
- Bold lower-case letters denote *column* vectors, e.g., **u**, **v**.
- The "$\times$" symbol indicates the multiplication of two matrices, two vectors, or a matrix and a vector.

Ratio Rules are based on the concepts of *eigenvalues* and *eigenvectors* and are closely related to the Singular Value Decomposition from matrix algebra. These concepts are defined below. Table 1 gives a list of symbols and their definitions.

Definition 1. *For a square $n \times n$ matrix* **S**, *a unit vector* **u** *and a scalar λ that satisfy*

$$\mathbf{S} \times \mathbf{u} = \lambda \times \mathbf{u} \tag{1}$$

are called an eigenvector *and its corresponding* eigenvalue, *respectively, of the matrix* **S**.

In order to proceed, we must explain the Singular Value Decomposition (SVD). The formal definition for SVD is as follows:

Theorem 1 (SVD). *Given an $N \times n$ real matrix* **X** *we can express it as*

$$\mathbf{X} = \mathbf{U} \times \Lambda \times \mathbf{V}^t \tag{2}$$

where **U** *is a column-orthonormal $N \times r$ matrix, r is the rank of the matrix* **X**, *Λ is a diagonal $r \times r$ matrix of the eigenvalues λ_i of* **X**, *and* **V** *is a column-orthonormal $n \times r$ matrix.*

Proof: See [PTVF92, p. 59]. □

Recall that a matrix **U** is called *column-orthonormal* if its columns $\mathbf{u}_i$ are mutually orthogonal unit vectors. Equivalently, $\mathbf{U}^t \times \mathbf{U} = \mathbf{I}$, where **I** is the identity matrix. Also, recall that the rank of a matrix is the highest number of linearly independent rows (or columns).

Equation 2 equivalently states that a matrix **X** can be brought in the following form, the so-called *spectral decomposition* [Jol86, p. 11]:

$$\mathbf{X} = \lambda_1 \mathbf{u}_1 \times \mathbf{v}_1^t + \lambda_2 \mathbf{u}_2 \times \mathbf{v}_2^t + \ldots + \lambda_r \mathbf{u}_r \times \mathbf{v}_r^t \tag{3}$$

where $\mathbf{u}_i$, and $\mathbf{v}_i$ are column vectors of the **U** and **V** matrices respectively, and λ_i the diagonal elements of the matrix Λ. Without loss of generality, we can assume that the eigenvalues λ_i are sorted in decreasing order. Figure 2 illustrates the rotation of axes that SVD implies for an example where M=2 (i.e., 2-D points). The corresponding two directions (x' and y') that SVD suggests are shown, meaning that, if we are allowed only k=1 axis, the best direction to project

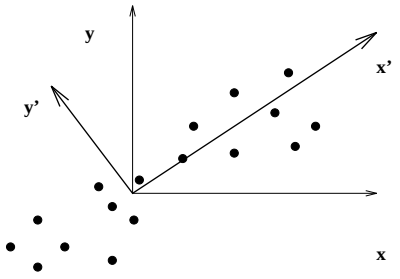


Fig. 2. Illustration of the rotation of axis that SVD implies: the "best" axis to project is x'

onto is the direction of x'; if we are allowed k=2, the best directions are x' and y'.

One of the by-products of SVD is that it reduces the dimensionality of a data set while retaining as much variation as possible. This is done by identifying the direction of maximum variance (given by the largest eigenvalue/vector) and then identifying the orthogonal direction with maximum variance (the second eigenvalue/vector), and so forth. In the end, only the eigenvectors associated with the k largest eigenvalues are kept while the remaining ones are truncated. These k largest eigenvectors give the Ratio Rules.

In order to choose a good cutoff k of rules to retain, the simplest textbook heuristic (and the one used in this paper) is to retain enough eigenvectors so that the sum of their eigenvalues covers 85% of the grand total [Jol86, p. 94]. That is, choose the cutoff k such that

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{M} \lambda_j} \approx 85\% \tag{4}$$

In addition to being a method for performing axis rotation and truncation, another intuitive way to view the SVD is that it tries to identify "rectangular blobs" of related values in the matrix **X**. This is best illustrated through an example.

Example: In the "toy" matrix of Table 2, we have two "blobs" of values, while the rest of the entries are zero. This is confirmed by the SVD, which identifies them both:

$$\mathbf{X} = \mathbf{U} \times \Lambda \times \mathbf{V}^t = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix} \tag{5}$$

or, in "spectral decomposition" form:

$$\mathbf{X} = 9.64 \times \begin{bmatrix} 0.18 \\ 0.36 \\ 0.18 \\ 0.90 \\ 0 \\ 0 \\ 0 \end{bmatrix} \times [0.58,\ 0.58,\ 0.58,\ 0,\ 0] +$$

Table 2. Example of a customer-product matrix

Item / Customer	*milk*	*bread*	*butter*	*tire*	*oil*
Smith	**1**	**1**	**1**	0	0
Doe	**2**	**2**	**2**	0	0
Johnson	**1**	**1**	**1**	0	0
Lee	**5**	**5**	**5**	0	0
Taylor	0	0	0	**2**	**2**
Sullivan	0	0	0	**3**	**3**
Thompson	0	0	0	**1**	**1**

$$+5.29 \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.53 \\ 0.80 \\ 0.27 \end{bmatrix} \times [0,\ 0,\ 0,\ 0.71,\ 0.71]$$

Notice that the rank of the **X** matrix is r=2; there are effectively two types of customers and, respectively, two types of rules: food shoppers and automotive shoppers, and two concepts (i.e., groups-of-products): the "food concept" (i.e., the group $\{$*milk, bread, butter*$\}$), and the "automotive concept" (i.e., the group $\{$*tire, oil*$\}$). The intuitive meaning of **U** and **V** is as follows:

Observation 1. **U** *can be thought of as the* customer-to-concept *similarity matrix.*

Observation 2. *Symmetrically,* **V** *is the* product-to-concept *similarity matrix.*

For example, $v_{1,2}$ = 0 means that the first product (*milk*) has zero similarity with the second concept (the "automotive concept"). **V** contains Ratio Rules in its columns; in the above example, **V** contains the following two rules:

1. *Customers typically spend* .58 : .58 : .58 : 0 : 0 *dollars on* *bread* : *milk* : *butter* : *tire* : *oil*.

2. *Customers typically spend* 0 : 0 : 0 : .71 : .71 *dollars on* *bread* : *milk* : *butter* : *tire* : *oil*.

The interpretation is that a customer will either spend an equal amount on all food products and none on automotive products, or vice versa.

Lemma 1. *The matrix* $\mathbf{X}^t \times \mathbf{X}$ *is a symmetric matrix, whose eigenvalues are the squares of the* λ_i *elements of* Λ *of the SVD of* **X**. *Moreover, the columns of* **V** *are the eigenvectors of* $\mathbf{X}^t \times \mathbf{X}$.

$$\mathbf{X}^t \times \mathbf{X} = \mathbf{V} \times \Lambda^2 \times \mathbf{V}^t \tag{6}$$

Proof: See [Fal96]. □

The intuitive meaning of the $M \times M$ matrix $\mathbf{X}^t \times \mathbf{X}$ is that it gives the product-to-product similarities. In our example, we have the following product-to-product similarities:

$$\mathbf{X}^t \times \mathbf{X} = \begin{bmatrix} 31 & 31 & 31 & 0 & 0 \\ 31 & 31 & 31 & 0 & 0 \\ 31 & 31 & 31 & 0 & 0 \\ 0 & 0 & 0 & 14 & 14 \\ 0 & 0 & 0 & 14 & 14 \end{bmatrix}$$

```
/* input: training set X on disk */
/* output: covariance matrix C */
for j := 1 to M do
   colavgs[j] ← 0;
   for l := 1 to M do
      C[j][l] ← 0;
for i := 1 to N do
   Read ith row of X from disk;
   for j := 1 to M do
      colavgs[j] += X[i][j];
      for l := 1 to M do
         C[j][l] += X[i][j]*X[i][l];
for j := 1 to M do
   colavgs[j] /= N;
for j := 1 to M do
   for l := 1 to M do
      C[j][l] -= N * colavgs[j] * colavgs[l];
```

a Single-pass over data matrix

```
input:
    covariance matrix C in main memory

output:
    eigenvectors v_1,...,v_k (i.e., the RRs)

compute eigensystem:
    {v_1,...,v_M} ← eigenvectors(C);
    {λ_1,...,λ_M} ← eigenvalues(C);
    sort v_j according to the eigenvalues;
    choose k based on Eq. 4;
    return the k largest eigenvectors;

complexity:
    O(M^3) main memory
```

b Eigensystem computation

Fig. 3. Pseudocode for efficiently computing Ratio Rules

Next we present a method for computing Ratio Rules by eigensystem analysis in a single pass.

4.3 A single-pass algorithm for Ratio Rules

A covariance matrix $\mathbf{C} = [c_{ij}]$ is a "column-to-column" similarity matrix, which has a high c_{ij} value if the columns i and j are correlated. Mathematically, it is defined as

$$\mathbf{C} \equiv \mathbf{X}_c^t \times \mathbf{X}_c \tag{7}$$

where $\mathbf{X}_c$ is derived from $\mathbf{X}$ by subtracting the respective column average from each and every cell. That is, $\mathbf{X}_c$ is a zero-mean, or "centered", matrix in the sense that its column averages are all zero. Thus, the covariance matrix $\mathbf{C}$ is a real, symmetric square matrix of side M. The computation of Ratio Rules involves determining the eigenvectors of $\mathbf{C}$, which, by Lemma 1, can be transformed into those of $\mathbf{X}_c$.

The following steps will compute the Ratio Rules in an I/O-efficient way: (a) zero-mean the input matrix to derive $\mathbf{X}_c$ and simultaneously compute $\mathbf{C}$ from Eq. 7 by updating partial sums; (b) compute the eigenvalues/vectors of $\mathbf{C}$ and pick the first k. We assume that $\mathbf{C}$ can fit in memory: it needs M^2 cells, where M is the number of columns, which should typically be on the order of one thousand for real applications [AIS93b]. Under this assumption, we can compute the column averages and the covariance matrix with a single pass over the N ($\approx$ millions) rows of the given $\mathbf{X}$ matrix, using the algorithm of Fig. 3a. Once we have $\mathbf{C}$ in main memory, we can use any off-the-shelf eigensystem package to determine its eigenvalues and eigenvectors, as shown in Fig. 3b.[3]

The proposed algorithm requires a *single pass* to compute the covariance matrix. In more detail, it requires O(N) I/Os to read the matrix $\mathbf{X}$ from disk, during which partial sums are maintained and zero-mean centering is performed, and O(NM^2) CPU operations to build the corresponding covariance matrix $\mathbf{C}$. It then requires O(M^3) CPU operations to compute the eigensystem. Since the number of rows N is typically in the hundreds of thousands (e.g., sales, or customers), and the number of columns M in the hundreds (e.g., products, or patient symptoms), the algorithm of Fig. 3 is very efficient. It should be noted that the algorithms of [AS96] require more than one pass over the data set in an attempt to find large itemsets. Also note that the O(M^3) factor for the eigensystem computation is negligible compared to the O(NM^2) operations needed to build the covariance matrix, since we assume that $N \gg M$.

[3] If the number of columns is much greater than one thousand, as potentially might be the case in some market basket data analyses, then the methods from [BDO95] could be applied to efficiently compute the eigensystem of the resulting sparse matrix.

4.4 Measuring the goodness of a rule-set: the "guessing error"

Let $\mathscr{R}$ be a given set of rules. We would like to be able to assess how good $\mathscr{R}$ is. The association rule mining literature has not defined a criterion to assess the "goodness", or accuracy, of a set of discovered rules. We propose a remedy, namely, the "guessing error". The fundamental requirement is that $\mathscr{R}$ must allow for estimations of missing values in a given record/row.

Let's consider a specific row (customer) $\mathbf{x}_i$ of the matrix, and pretend that the j-th attribute is hidden from us (i.e., the amount spend on the j-th product, say, bread). Given $\mathscr{R}$ and the rest of the values $x_{i,m}$ ($m \neq j$), we should be able to estimate the missing value as $\hat{x}_{ij}$. The *guessing error* for this specific cell (i, j) is $\hat{x}_{ij} - x_{ij}$.

Definition 2. *The "single-hole guessing error", or simply the "guessing error", for a set of rules $\mathscr{R}$ on a data matrix $\mathbf{X}$ is defined as the root-mean-square of the guessing errors of the individual cells, that is,*

$$GE = \sqrt{\frac{1}{NM}\sum_{i}^{N}\sum_{j}^{M}(\hat{x}_{ij} - x_{ij})^2} \tag{8}$$

More specifically, we also define it as the *single-hole guessing error* GE_1 because we allowed only a single hole at a time. The generalization to the h-hole guessing error GE_h is straightforward.

```
/* input: b_H, a 1 × M row vector with holes */
/* output: b̂, a 1 × M row vector with holes filled */
1.  V' ← E_H × V;                          /* ''RR-hyperplane'' */
2.  b' ← E_H × b^t_H;                      /* ''feasible sol'n space'' */
3.  solve V' × x_concept = b' for x_concept  /* solution in k-space */
4.  d ← V × x_concept;                     /* solution in M-space */
5.  b̂ ← b × [E_Hc]^t + d × [E_H]^t;
```

Fig. 4. Pseudocode for filling holes based on Ratio Rule matrix **V**

Definition 3. *The "h-hole guessing error" for a set of rules $\mathscr{R}$ on a data matrix* **X** *is defined as the root-mean-square of the guessing errors of configurations of h* simultaneous *cells, that is,*

$$GE_h = \sqrt{\frac{1}{Nh|\mathscr{H}_h|}\sum_{i}^{N}\sum_{\mathscr{H}\in\mathscr{H}_h}\sum_{j\in\mathscr{H}}(\hat{x}_{ij}-x_{ij})^2} \qquad (9)$$

where $\mathscr{H}_h$ contains some subset of the $\binom{M}{h}$ combinations of sets $\mathscr{H}$ with h "holes".

The way that $\mathscr{R}$ is derived is independent of the definition of the "guessing error". We expect that the typical practice in machine learning will be followed: we can use a portion $\mathbf{X}_{train}$ of the data set **X** to derive the rules $\mathscr{R}$ ("training set"), and some other portion $\mathbf{X}_{test}$ of the data set **X** to compute the guessing error ("testing set"). The details of the choice of training and testing sets is orthogonal to our definition, and outside the scope of this paper, since they have been extensively examined in the machine learning and classification literature [Qui93]. A reasonable choice is to use 90% of the original data matrix for training and the remaining 10% for testing. Another possibility is the use of the entire data matrix for both training and testing. In this paper, we report only the results for the former choice because the two choices above gave very similar results.

The ability to measure the goodness of a set of rules for a given testing data set is very important, for developers of data-mining products and for end-users alike:

- For developers, it allows benchmarking and comparison with competing products and designs: a low "guessing error" over a variety of input matrices indicates a good product.
- For end-users that use a given product on a specific data set, a low "guessing error" implies that the derived rules have captured the essence of this data set, and that they can be used for estimation of truly unknown values with more confidence.

It should be highlighted that the definition of the "guessing error" can be applied to *any* type of rules, as long as they can do estimation of hidden values. In the next section we focus on the proposed Ratio Rules, and show how to use them to obtain such estimates.

4.5 Determining hidden and unknown values

Here we present an algorithm for determining unknown values of a data matrix both algebraically and geometrically. If we can reconstruct these so-called "holes", then we can find hidden values or forecast future values. This framework is also applicable to "what-if" scenarios where we can specify some of the values ('*What if the demand for Cheerios doubles?*') and then forecast the effect on other attributes ('*Then the demand for milk will double.*'). In addition, it can be used to discover outliers by hiding a cell value, reconstructing it, and comparing the reconstructed value to the hidden value. A value is an outlier when the value predicted is significantly different (e.g., two standard deviations away) from the existing hidden value.

We begin by developing some notation necessary for formulating the problem algebraically. We show how the problem leads to an algebraic system of equations. Figure 4 gives the pseudocode. Figures 5–7 illustrate the solution geometrically.

Definition 4. *An h-hole row vector $\mathbf{b}_{\mathscr{H}}$ is defined as a vector with holes (denoted with "?"s) at indices given in $\mathscr{H}$, where $\mathscr{H}$ is the set of "holes".*

An example of a 1×5 2-hole row vector is the following:

$$\mathbf{b}_{\{2,4\}} = [b_1, ?, b_3, ?, b_5]$$

Definition 5. *An $(M-h)\times M$* **elimination matrix** $\mathbf{E}_{\mathscr{H}}$ *is defined as an $M \times M$ identity matrix with $h = |\mathscr{H}|$ rows removed, where the row indices are given in the set $\mathscr{H}$.*

An example of a 3×5 elimination matrix is the following:

$$\mathbf{E}_{\{2,4\}} = \begin{bmatrix} 1&0&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&0&1 \end{bmatrix}$$

An elimination matrix is very useful in helping us pick and choose entries from vectors. For example, we can eliminate the "?"s from $\mathbf{b}_{\{2,4\}}$ as follows:

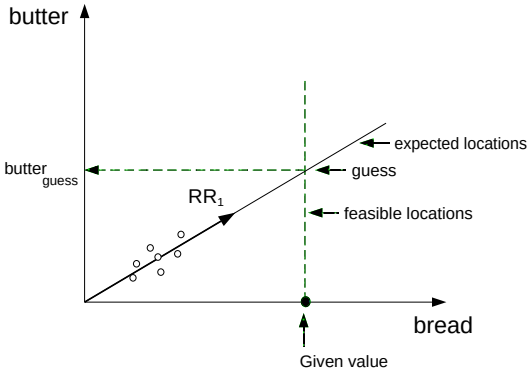


Fig. 5. The exactly-specified case

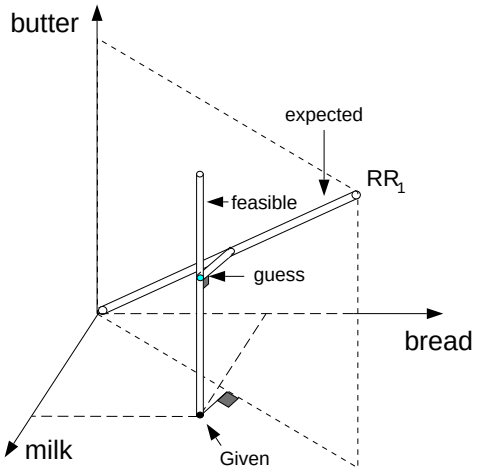


Fig. 6. The over-specified case

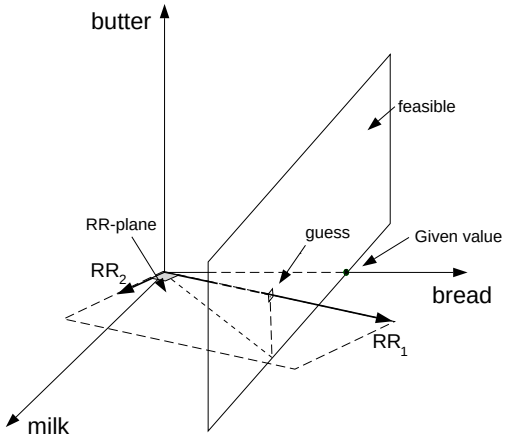


Fig. 7. The under-specified case

$$\mathbf{E}_{\{2,4\}} \times \mathbf{b}^t_{\{2,4\}} = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix} \times \begin{bmatrix} b_1 \\ ? \\ b_3 \\ ? \\ b_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_3 \\ b_5 \end{bmatrix}$$

Once the user has specified partial knowledge from a transaction $\mathbf{b}_{\mathscr{H}}$ (e.g., the dollar amounts spent by a new customer, for some products), the set of unknowns $\mathscr{H}$ are determined by the k Ratio Rules that have been kept, and are reported as $\hat{\mathbf{b}}$, that is, $\mathbf{b}_{\mathscr{H}}$ with the holes $\mathscr{H}$ filled in. The geometric intuition is the following: the rules form a k-dimensional hyper-plane $\mathbf{V}'$ (= $\mathbf{E}_{\mathscr{H}} \times \mathbf{V}$) in M-space, the "RR-hyperplane", on or close to which the data points lie. The h holes result in an h-dimensional hyper-plane $\mathbf{b}'$ (= $\mathbf{E}_{\mathscr{H}} \times \mathbf{b}^t_{\mathscr{H}}$) in M-space, the "feasible solution space", on which the solution is constrained. We want to find a point that agrees with our given partial data ("feasible solution space"), and is as close to (or exactly on) the RR-hyperplane.

Figure 5 illustrates the case in the simplest possible form: we have M=2 products (say, amount spent on "bread" for the x-axis, and amount spent on "butter" for the y-axis), k=1 rule, and h=1 hole. We know (a) that a customer spends the given amount on bread and (b) that most of our previous customers fall on or close to the line defined by the first rule (RR_1). We want to find the amount spent on butter (the hole). The intersection of "feasible locations" (vertical dashed line) and "expected locations" (solid diagonal line) gives our best prediction for the 2-D point that corresponds to that sale; the value on the "butter" axis, labeled as "guess" is our proposed estimate for the required amount spent on butter.

The intersection of the two hyper-planes corresponds to a system of linear equations $\mathbf{V}' \times \mathbf{x}_{concept} = \mathbf{b}'$, from which the solution of $\mathbf{x}_{concept}$ determines the unknowns.

Recall that the intersection of "feasible locations" and "expected locations" gives our best prediction. There are three possibilities regarding the intersection of the two hyperplanes, which are illustrated in Figs. 5–7. Respectively, there are three possibilities regarding the equation from step 3 of the pseudocode,

$$\mathbf{V}' \times \mathbf{x}_{concept} = \mathbf{b}' \tag{10}$$

given that there are $(M - h)$ equations and k unknowns.

CASE 1: (EXACTLY-SPECIFIED) The two hyper-planes intersect at a point. This occurs when $(M - h) = k$. The respective linear equations have an exact solution determined by

$$\mathbf{x}_{concept} = (\mathbf{V}')^{-1} \times \mathbf{b}' \tag{11}$$

Figure 5 illustrates an example in M = 2 dimensions, for h = 1 hole and cutoff k = 1 ratio rule.

CASE 2: (OVER-SPECIFIED) The two hyper-planes do not intersect. This occurs when $(M-h) > k$. The respective equations are over-determined, and the closest distance between them is chosen for the solution to $\mathbf{x}_{concept}$ based on the Moore-Penrose pseudo-inverse of $\mathbf{V}'$ [PTVF92]. This uses the singular value decomposition of $\mathbf{V}'$:

$$\mathbf{V}' = \mathbf{R} \times diag(\mu_j) \times \mathbf{S}^t \tag{12}$$

Since $\mathbf{V}'$ is singular, no inverse exists, but we can find a pseudo-inverse:

$$[\mathbf{V}']^{-1} = \mathbf{S} \times diag(1/\mu_j) \times \mathbf{R}^t \tag{13}$$

and, thus,

$$\mathbf{x}_{concept} = [\mathbf{V}']^{-1} \times \mathbf{b}' \tag{14}$$

Figure 6 illustrates an example in M = 3 dimensions, for h = 1 hole and cutoff k = 1.

CASE 3: (UNDER-SPECIFIED) The intersection of the two hyper-planes forms a $(min(k,h)-1)$-dimensional hyperplane. This occurs when $(M - h) < k$. The respective equations are under-determined. Among the infinite solutions, we propose to keep the one that needs the fewest eigenvectors. Thus, we ignore $(k+h) - M$ rules to make the system exactly-specified, and then solve it using CASE 1. Figure 7 illustrates an example in M = 3 dimensions, for h = 2 holes and cutoff k = 2.[4]

[4] We also experimented with the Moore-Penrose pseudo-inverse to find a least-squares estimation for this case, but the solution presented turned out to give more accurate estimations.

5 Experiments

We ran four sets of experiments. The first was to investigate the prediction accuracy achieved by the proposed method; the second was to examine the robustness of Ratio Rules in estimating more than one simultaneous hole (i.e., that the relative accuracy does not diminish as the number of holes is increased); the third was to examine our method on binary data; the fourth was to see how our method scales up for large data sets.

Methods: Since the literature on association rules has not addressed the issue of reconstructing missing/hidden values, there is no way to do an objective comparison with them. While it may be possible that current AR methods can be adapted for interpolations (e.g., by choosing the "centroid" of the most similar rule), it is an open problem as to how well such techniques would work. Should a clever scheme for reconstruction based on AR be proposed in the future, we have set forth a framework for a fair comparison against Ratio Rules, using our guessing error framework[5]. In any event, note that AR are inherently unable to give extrapolations. Thus, we compared Ratio Rules with a straightforward technique for predicting values, named `col-avgs`: *for a given hole, use the respective column average from the training set.* Note that `col-avgs` is identical to the proposed method with $k = 0$ eigenvalues.

Notice that linear regression, for example, [Hou96], is unable to fill in arbitrary holes. The reason is that we need one regression model for each combination of holes. Specifically, linear regression has n *independent* variables, that have to be given to us, and one *dependent* variable. The regression model will then express the value of the dependent variable as a linear combination of the values of the n independent ones. That is, if we want to predict the amount spent on "bread", given the amount spent on "milk" and "butter", we have to build one regression model; if we want to regress "bread" on "butter" only, we have to build *another* regression model, and so on. Thus, if we want to predict a given attribute, for any combination of holes in the other $M - 1$ attributes, we clearly need the power set: 2^{M-1} different regression models. In order to predict *any* attribute, given an arbitrary set of the other attributes, we need an exponential number of regression models, namely $M \times 2^{M-1}$, which is clearly impractical even for moderate values of M. In contrast, Ratio Rules can predict any attribute, given any subset of the other attributes.

We cannot compare Ratio Rules with any association-based methods because, as we argue in Sect. 6.3, association-based methods do not lead to prediction of missing values.

Error Measure: We use the GE_h "guessing error" which was described in Sect. 4.4.

Data sets: We ran our experiments on a variety of real data sets (see Sect. 6.1 which displays scatter-plots of them), described as follows:

- **'nba'** (459 × 12) - basketball statistics from the 1991–92 NBA season, including minutes played, field goals, rebounds, and fouls;
- **'baseball'** (1574 × 17) - batting statistics from Major League Baseball for four seasons; fields include batting average, at-bats, hits, home runs, and stolen bases;[6]
- **'abalone'** (4177 × 7) - physical measurements of an invertebrate animal, including length, diameter, and weights.[7]

Preliminary to running these experiments, for each data set we chose 90% of the matrix rows for the training matrix; the remaining 10% were used as the testing matrix. We computed the Ratio Rules from the training matrix, along with the column averages of the training matrix for use as the competitor (`col-avgs`).

5.1 Reconstruction accuracy

Figure 8 shows the GE_1 guessing error for the **'nba'**, **'baseball'**, and **'abalone'** data sets, normalized by the guessing error attained by `col-avgs`. As a frame of reference, we also present the normalized GE_1 of `col-avgs`, which is, of course, 100%. Note that the proposed method method was the clear winner for all data sets we tried and gave as low as one-fifth the guessing error of `col-avgs`.

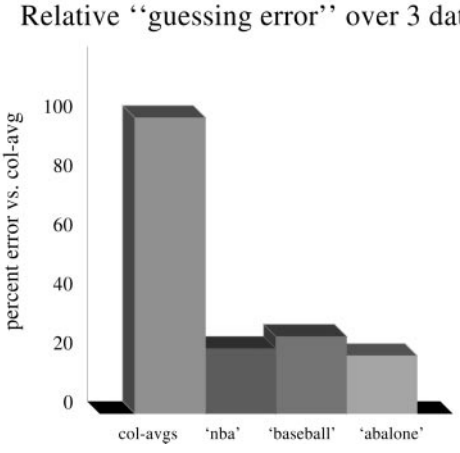


Fig. 8. Ratio of guessing error between RR and `col-avgs`, for **'nba'**, **'baseball'**, and **'abalone'**

5.2 Guessing error for simultaneous holes

In Fig. 9, we show GE_h for the **'nba'** and **'baseball'** data sets, for $1 \leq h \leq 5$ holes. The results for the **'abalone'** data set were similar, and are omitted for brevity. Note that the guessing error is rather insensitive for up to several simultaneous holes. Note that GE_h is constant with respect to h for `col-avgs` since the computation of GE_h turns out to be the same for all h, for that method.

[5] The ability to have an objective, numerical estimate of the goodness of a set of rules was exactly the motivation behind this paper.

[6] **'baseball'** is available at `www.usatoday.com/sports/baseball/sbstats.htm`.

[7] **'abalone'** is available at `www.ics.uci.edu/~mlearn/MLSummary.html`.

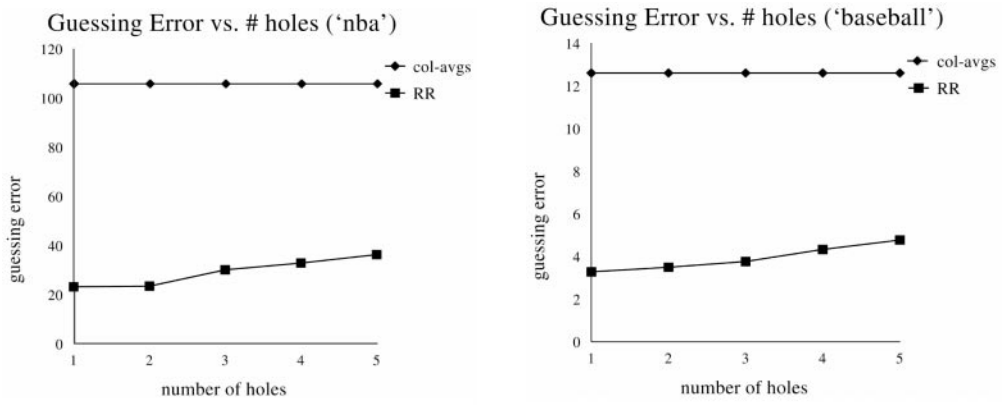


Fig. 9. Guessing error vs. number of holes (1–5) for the **'nba'** and **'baseball'** data sets

Table 3. Binary matrix of customers and products and its first three Ratio Rules

	milk	*bread*	*butter*	*tire*	*bulb*	*oil*	*shirt*	*pants*	*shoes*	*hat*
Smith	**1**	**1**	**1**	0	0	0	0	0	0	0
Doe	**1**	**1**	**1**	0	0	0	0	0	0	0
...					...					
Johnson	0	0	0	**1**	**1**	**1**	0	0	0	0
Taylor	0	0	0	**1**	**1**	**1**	0	0	0	0
...					...					
Lee	0	0	0	0	0	0	**1**	**1**	**1**	**1**
Sullivan	0	0	0	0	0	0	**1**	**1**	**1**	**1**
...					...					
noise1	0	1	0	0	1	1	0	0	0	0
noise2	0	0	0	1	0	0	0	0	0	1
noise3	0	1	0	0	0	1	0	1	1	0
...					...					

a matrix of food, automotive and clothes groups

field	RR_1	RR_2	RR_3
milk	.003	.018	**.577**
bread	.003	.018	**.577**
butter	.002	.018	**.578**
tire	.002	**.576**	.019
bulb	.003	**.576**	.019
oil	.003	**.579**	.015
shirt	**.500**	.001	.002
pants	**.500**	.003	.002
shoes	**.500**	.003	.002
hat	**.500**	.002	.002

b first three Ratio Rules

5.3 Ratio rules on binary data

We performed some experiments on binary (e.g., market basket) data. The goal was to see if Ratio Rules could distinguish between three different groups of items where the groups were food (*milk, bread, butter*), automotive (*tire, bulb, oil*), and clothes (*shirt, pants, shoes, hat*). Most of the matrix rows represented transactions involving items from one and only one group. In other words, given any pair of rows, all the items were either from exactly the same group or from two mutually disjoint groups. The rest of the rows were 'noise', which was generated by randomly selecting items across separate groups, and it was possible that representatives from several groups could be chosen. This matrix format is illustrated in Table 3a.

Table 3b shows the Ratio Rules for this type of matrix with 10000 rows and with the 10 attributes listed above. The rows comprise of 17.5% from the first group, 25% from the second group, 50% from the third group, and 7.5% noise; the dominant values of each rule vector are highlighted. Note that the component values of the Ratio Rules are roughly mutually disjoint, i.e., values outside the primary group are close to zero. From Table 3b we see that the three ratio rules are essentially: RR_1: *shirt* : *pants* : *shoes* : *hat* = 1 : 1 : 1 : 1, RR_2: *tire* : *bulb* : *oil* = 1 : 1 : 1, and RR_3: *milk* : *bread* : *butter* = 1 : 1 : 1. In this case, the Ratio Rules were able to identify almost perfectly the three groups despite the presence of noise: RR_1 represents the "clothes" group, RR_2 the "automotive" group, and RR_3 the "food" group.

We performed a sensitivity analysis to understand how the reconstruction accuracy is affected by noise. As in the example above, we used a matrix of 10000 rows with the same items and groups. Figure 10 displays the one-hole guessing error (GE_1) of Ratio Rules as a function of noise. The noise varied from 2–10% with the remaining (noise-free) rows chosen with equal probability from the three groups. Note that the guessing error grows slowly with increasing noise.

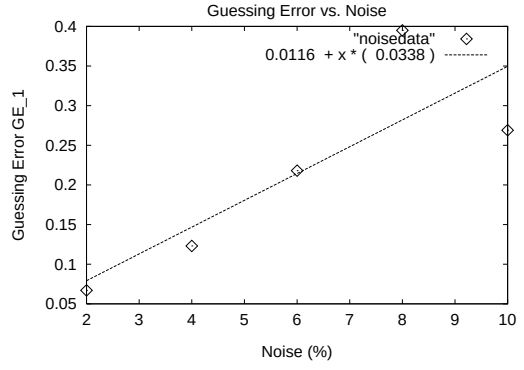


Fig. 10. Sensitivity analysis of 'noise' as a function of guessing error

5.4 Scale-up

Figure 11 demonstrates the scale-up of our algorithm. The vertical axis is the average actual computation time to determine the Ratio Rules (in seconds), as measured by the `time` utility of UNIX. The horizontal axis is the number of data matrix rows N. Since all of our data sets are relatively small ($N < 5000$) for this experiment, we used a 100000 $\times$ 100 data matrix created using the Quest Synthetic Data Generation Tool.[8] The methods were implemented in `C` and `Splus`. The experiments ran on a dedicated Sun SPARCstation 5 with 32 Mb of main memory, running SunOS 4.1.3. The disk drive was a Fujitsu M2266S-512 model 'Cranel-M2266SA' with minimum positioning time of 8.3 ms and maximum positioning time of 30 ms.

The plot is close to a straight line, as expected. The y-intercept of the line is the time to compute the eigensystem, which is always $O(M^3) = O(100^3)$, which apparently has a negligible effect on the curve.

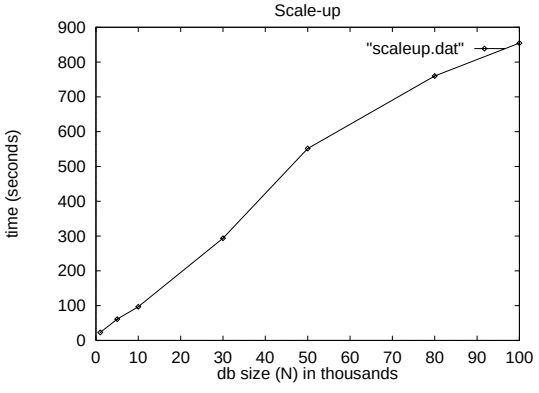


Fig. 11. Scale-up: time to compute RR versus db size N in records

6 Discussion

Here we show the visualization capabilities that Ratio Rules offer by presenting 2-D scatter-plots of the data sets used. Using the **'nba'** data set, we demonstrate how these Ratio Rules can be interpreted, with references to the plots. Finally, we present a qualitative comparison of the Ratio Rules versus general association rules [SA96].

6.1 Visualization

Recall that Ratio Rules identify the axes of greatest variation. Just like with PCA, by projecting the points onto the best two or three of these axes (i.e., the eigenvectors associated with the largest eigenvalues), the points can be plotted to give an idea of the density and structure of the data set. For example, Fig. 12 shows a scatter-plot of **'nba'**, statistics over the 1991–92 basketball season, of N=459 players for M=12 attributes and has been reduced to 2-dimensional RR-space (i.e., two Ratio Rules). In Fig. 12a, the x-axis corresponds to the first (and strongest) rule RR_1; the y-axis corresponds to RR_2. In Fig. 12b, the x-axis corresponds to RR_2 and the y-axis corresponds to RR_3. Most of the points are very close to the horizontal axis, implying that they all closely follow the first eigenvector and are considerably linear. The plot also shows that many of the attributes are correlated with one another, such as field goals and minutes played. There are two points that are clearly outliers: $(3000, 971)$ and $(2100, -1296)$, corresponding to Michael Jordan and Dennis Rodman, respectively. Figure 13 shows 2-D plots for part a **'baseball'** and part b **'abalone'**.

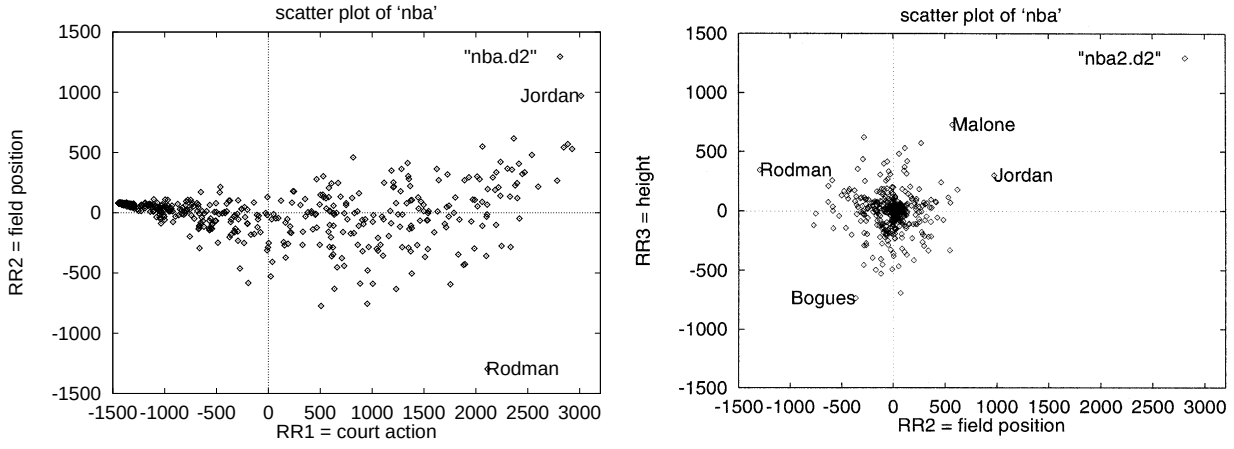
[8] Quest is available at `www.almaden.ibm.com/cs/quest/syndata.html`.

6.2 Interpretation of the Ratio Rules

In this section, we discuss an example using the **'nba'** data set of how a set of Ratio Rules can be interpreted as meaningful rules. The methodology is outlined in Fig. 14.

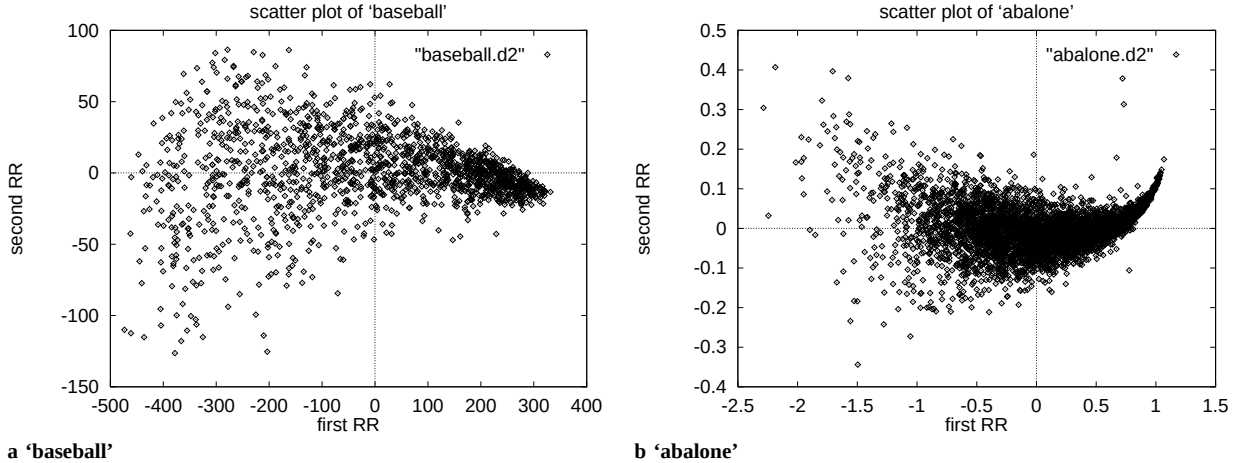
Table 4 presents the first three Ratio Rules (RR_1, RR_2, and RR_3) for the **'nba'** data set, which records statistics such as minutes played, points, total rebounds, assists, and steals. Based on a general knowledge of basketball and through examination of these rules, we conjecture that RR_1 represents the level of activity of a player, separating the starters from those who sit on the bench, and gives a 0.808:0.406 $\approx$ 2:1 ratio. This is a Ratio Rule with the obvious interpretation: the average player scores 1 point for every 2 mins. of play (equivalently, 1 basket for every 4 mins. played). According to RR_1, Michael Jordan was by far the most active player in almost every category (see Fig. 12a). RR_2 shows that the number of rebounds is negatively correlated with points in a 0.489:0.199 $\approx$ 2.45:1 ratio. This is because a goal attempt makes it difficult for a player to get in a good position for rebounding, and vice versa. For that reason, "minutes played" and "points" are also negatively correlated, meaning that a rebounder scores less as a percentage of time on the field than players who place emphasis on offense. Thus, RR_2 roughly represents the field position, separating the guards, who get the most opportunities to shoot, from the forwards, who are more likely to be rebounders. For example, we see, in Fig. 12a, the extremes among active players: star shooting guard Michael Jordan at one end with 2404 points and 91 rebounds, and power forward (and excellent rebounder) Dennis Rodman at the other with 800 points and 523 rebounds. RR_3 says that rebounds are negatively correlated with assists and steals. Typically, tall players make better rebounders because they can reach high and short players are better at assists and steals because they can move fast. Thus, RR_3 roughly represents the height of a player, with Mugsy Bogues (5'3") and Karl Malone (6'8") at opposite extremes (see Fig. 12b).

We looked at the Ratio Rules of the other data sets and found that they are also amenable to interpretation, and that they give intuitive rules. RR_1 for **'baseball'** indicates the effectiveness of a batter, giving a formula relating the total number of at-bats to the total number of hits, doubles and home runs; RR_2 distinguishes between those players who got on base by slugging and those who often drew a walk; RR_3 identifies the base stealers. The Ratio Rules from the **'abalone'** set were interpreted as follows: RR_1 gives a formula relating the amount of weight of different parts of a



a side view **b** front view

Fig. 12. A scatter plot of **'nba'**: two 2-D orthogonal views



a 'baseball' **b 'abalone'**

Fig. 13. Scatter plots of **a 'baseball'** and **b 'abalone'** in 2-D RR space

Table 4. Relative values of the RRs from **'nba'**

field	RR_1	RR_2	RR_3
minutes played	.808	−.4	
field goals			
goal attempts			
free throws			
throws attempted			
blocked shots			
fouls			
points	.406	.199	
offensive rebounds			
total rebounds		−.489	.602
assists			−.486
steals			−.07

mollusk to one other; RR_2 points out an negative relationship between the amount of shucked weight and the remaining weight; RR_3 distinguishes between long-thin and short-fat body types.

6.3 Ratio rules vs. association rules

Ratio Rules are quite different from association rules in many qualitative aspects. Here we compare and contrast the two paradigms. Of the association rules, we examine both

1. Solve the eigensystem;
2. Keep k strongest rules according to Eq. 4;
3. Display Ratio Rules graphically in a histogram;
4. Observe positive and negative correlations;
5. Interpret.

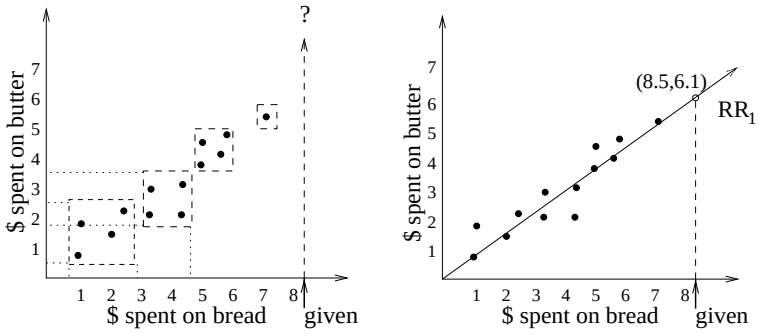
Fig. 14. Interpretation of Ratio Rules

Boolean and quantitative rules. Examples of each type of rule with which we are concerned follow:

- Boolean association rules [AIS93b]:
  $\{bread, milk\} \Rightarrow butter$
- Quantitative association rules [SA96]:
  $bread : [2-5] \Rightarrow butter : [1-2]$
- Ratio Rules:
  ratio of spendings *bread:butter = 2:3*

Boolean association rules have the advantages that they are easy to interpret and relatively easy to implement. The major drawback, however, is that a given data matrix **X** with, for example, amounts spent per customer per product is converted to a binary matrix by treating non-zero amounts as plain "1"s. This simplifies the data mining algorithms but tends to lose valuable information.

Quantitative association rule algorithms perform an important step to retain the above information. Figure 15a il-



a Quantitative **b** Ratio Rules

Fig. 15. Illustration of rules from a fictitious data set of sales on bread and butter: **a** quantitative association rules **b** Ratio Rules. The "given" entry asks for an estimation for butter, for the given amount spent on bread

lustrates how these rules might work for a fictitious data set with a few customers (points) and $M = 2$ products only, namely, "bread" and "butter". In this data set, the quantitative association rules will derive rules that correspond to the dashed rectangles of the figure. For example, the first two lower-left rectangles will yield the rules

$$\textit{bread} : [1 - 3] \quad \Rightarrow \quad \textit{butter} : [.5 - 2.5]$$

$$\textit{bread} : [3 - 5] \quad \Rightarrow \quad \textit{butter} : [2 - 3]$$

Ratio Rules, for the same setting of Fig. 15 and with $k = 1$ rule, will fit the best possible line through the data set; its unit vector is exactly the first rule of the given data matrix. Thus, the corresponding rule will be

$$\textit{bread} : \textit{butter} = .81 : .58$$

For the remaining discussion, we focus only on quantitative association rules since the focus is on real-valued data such as dollar amounts spent by customers on products. We compare the strengths of quantitative association rules with those of Ratio Rules.
The advantages of quantitative association rules include the following:

- They will be more suitable if the data points form clusters.
- They have been applied to categorical data.

The advantages of Ratio Rules include the following:

- They achieve more compact descriptions if the data points are linearly correlated, as in Fig. 15, or as in the real data sets that we saw earlier. In such cases, a single Ratio Rule captures the correlations, while several minimum bounding rectangles are needed by the quantitative association rules to convey the same information.
- They can perform extrapolations and predictions. For example, in Fig. 15, suppose that we are given that a customer bought \$8.50 of bread and we want to know how much butter s/he is expected to buy. Ratio Rules will predict \$6.10 on butter, as Fig. 15b illustrates. Quantitative association rules have no rule that can fire because the vertical line of "feasible solutions" intersects none of the bounding rectangles. Thus they are unable to make a prediction.
- Their derivation requires a single pass over the data set.
- They are easily implemented: thanks to highly fine-tuned eigensystem packages, and the remaining programming effort is minimal.

7 Conclusions

We have proposed a completely different type of rules as the target of data mining efforts, namely, *Ratio Rules*. These rules have significant advantages over Boolean and quantitative association rules:

- They lead to a natural measure, the "guessing error", which can quantify how good a given set of rules is.
- They can be used to estimate one or more unknown (equivalently, missing, hidden or corrupted) values when a new data record is given, based on the novel method proposed in Sect. 4.5; thus, they can also be used in forecasting, for "what-if" scenarios, and for detecting outliers.
- They are easy to implement. The most difficult part of our method is the solution of an eigensystem for which reliable packages and/or source code are widely available.
- They are fast and scalable, requiring a *single pass* over the data matrix, and growing linearly on the largest dimension of the matrix, presumably the number N of rows (customers).
- They give visualization for free, thanks to the dimensionality reduction properties of Ratio Rules.

We described how to interpret Ratio Rules and we discussed their qualitative differences from association rules. Finally, we presented experiments on several real data sets, which showed that the proposed Ratio Rules scale-up for large data sets, and can achieve up to 5 times smaller guessing error than its competitor. Our experiments on binary matrices showed that Ratio Rules can find large itemsets, even in the presence of noise. Future research could focus on applying Ratio Rules to data sets that contain categorical data.

Acknowledgements. We would like to thank Björn Thór Jónsson and Kostas Stathatos for their help in interpreting the Ratio Rules for the **'nba'** data set. We would also like to thank Rakesh Agrawal for offering us his synthetic data set generator, and Mike Franklin for providing an RS/6000 to install and run the generator.

References

[AAR96] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. "A Linear Method for Deviation Detection in Large Databases". In *Proc. of 2^{nd} International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, USA, August 1996.

[AIS93a] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. "Database Mining: A Performance Perspective". *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, December 1993.

[AIS93b] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. "Mining Association Rules between Sets of Items in Large Databases". In *Proc. of the 1993 ACM SIGMOD Conference*, pages 207–216, Washington D.C., USA, May 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules". In *Proc. of the* 20^{th} *VLDB Conference*, pages 487–499, Santiago, Chile, September 1994.

[AS96] Rakesh Agrawal and John C. Shafer. "Parallel Mining of Association Rules". *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, December 1996.

[BDO95] Michael Berry, Susan Dumais, and Gavin O'Brien. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, 37:573–595, 1995.

[BMS97a] Sergey Brin, Rajeev Motwani, and Craig Silverstein. "Beyond Market Baskets: Generalizing Association Rules to Correlations". In *Proc. of the 1997 ACM SIGMOD Conference*, pages 265–276, Tucson, Arizona, USA, May 1997.

[BMS97b] Oliver Buechter, Michael Mueller, and Josef Shneeberger. Improving forward selection with background knowledge: Finding interesting multiple regression models for sales prediction. In *Proc. PAKDD '97*, pages 344–357, Singapore, 1997.

[CHY96] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. "Data Mining: An Overview from a Database Perspective". *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, December 1996.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[Fal96] Christos Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Inc., 1996.

[FD92] Peter W. Foltz and Susan T. Dumais. "Personalized Information Delivery: an Analysis of Information Filtering Methods". *Comm. of ACM (CACM)*, 35(12):51–60, December 1992.

[FU96] Usama Fayyad and Ramasamy Uthurusamy. Data mining and knowledge discovery in databases. *Communications of the ACM: Data Mining and Knowledge Discovery (special issue)*, 39(11), November 1996.

[HF95] Jiawei Han and Yongjian Fu. "Discovery of Multiple-Level Association Rules from Large Databases". In *Proc. of the* 21^{st} *VLDB Conference*, pages 420–431, Zurich, Switzerland, September 1995.

[Hou96] Wen-Chi Hou. "Extraction and Applications of Statistical Relationships in Relational Databases.". *IEEE Transactions on Knowledge and Data Engineering*, 8(6):939–945, December 1996.

[Jol86] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. "An Effective Hash Based Algorithm for Mining Association Rules". In *Proc. of the 1995 ACM SIGMOD Conference*, pages 175–186, San Jose, California, USA, May 1995.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992. 2nd Edition.

[Qui93] John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Mateo, CA, 1993.

[SA95] Ramakrishnan Srikant and Rakesh Agrawal. "Mining Generalized Association Rules". In *Proc. of the* 21^{st} *VLDB Conference*, pages 407–419, Zurich, Switzerland, September 1995.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. "Mining Quantitative Association Rules in Large Relational Tables". In *Proc. of the 1996 ACM SIGMOD Conference*, pages 1–12, Montreal, Quebec, Canada, June 1996.

[SM83] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[SON95] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. "An Efficient Algorithm for Mining Association Rules in Large Databases". In *Proc. of the* 21^{st} *VLDB Conference*, pages 432–444, Zurich, Switzerland, September 1995.

[ST96] Abraham Silberschatz and Alexander Tuzhilin. "What Makes Patterns Interesting in Knowledge Discovery". *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, December 1996.