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; e-mail: flip@research.att.com
$^2$ University of Maryland, College Park, MD 20742; e-mail: {labrinid,kotidis}@cs.umd.edu
$^3$ Carnegie Mellon University, Pittsburgh, PA 15213; e-mail: christos@cs.cmu.edu

Edited by Jennifer Widom and Oded Shmueli. Received: March 1999 / Accepted: November 1999

Abstract  Association Rule Mining algorithms operate on a data matrix (e.g., customers × products) to derive association rules (Agrawal, Imielinski, & Swami, 1993b; Srikant & Agrawal, 1996). 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 (Fayyad & Uthurusamy, 1996), of which the quintessential problem in database research has been association rule mining (Agrawal et al., 1993b). Given a data matrix with, e.g., customers for rows and products for columns, association rule algorithms find rules that describe frequently co-occurring products, and are of the form

\[ \{\text{bread, milk}\} \Rightarrow \text{butter} \ (90\%)\]

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 (Agrawal et al., 1993b), as well as all the follow-up database work (Agrawal & Srikant, 1994), 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

\[ \text{Customers typically spend 1 : 2 : 5 dollars on bread : milk : butter.} \]
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:

- 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?’;
- 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, section 7 lists some conclusions and gives pointers to future work.

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.’

2 Related Work

(Agrawal, Imielinski, & Swami, 1993a) 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 (Chen, Han, & Yu, 1996) for an excellent survey of all three problems.

The seminal work of (Agrawal et al., 1993b) introduced the problem of discovering association rules and presented an efficient algorithm for mining them. Since then, new serial algorithms (Agrawal & Srikant, 1994; Park, Chen, & Yu, 1995; Savasere, Omiecinski, & Navathe, 1995) and parallel algorithms (Agrawal & Shafer, 1996) have been proposed. In addition, generalized association rules have been the subject of recent work (Srikant & Agrawal, 1995; Han & Fu, 1995).

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 (Srikant & Agrawal, 1996), 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

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

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 (Agrawal et al., 1993b); recent alternative criteria include the chi-square test (Brin, Motwani, & Silverstein, 1997) and probability-based measures (Silberschatz & Tuzhilin, 1996). Related issues include outlier detection and forecasting. See (Jolliffe, 1986) for a textbook treatment of both, and (Arning, Agrawal, & Raghavan, 1996; Hou, 1996; Buechter, Mueller, & Shneberger, 1997) for recent developments.

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 \( X \), where each row corresponds to a customer transaction (e.g., market basket purchase), and entry \( x_{ij} \) gives the
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.

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 Section 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, (Press, Teukolsky, Vetterling, & Flannery, 1992), Principal Component Analysis, PCA, (Jolliffe, 1986), Latent Semantic Indexing, LSI, (Foltz & Dumais, 1992), and the Karhunen-Loeve Transform, KLT, (Duda & Hart, 1973). Eigensystem analysis involves computing the eigenvectors and eigenvalues of the covariance matrix of the given data points (see subsection 4.1 for intuition and 4.2 for more formal treatment). In subsection 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. Subsection 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. Subsection 4.5 presents another major contribution: how to use the Ratio Rules to predict missing values.

4.1 Intuition Behind Ratio Rules

![Graphical representation of eigensystem analysis](image)

Fig. 1(a) 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. Fig. 1(b) 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 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.
Definition 1 For a square \( n \times n \) matrix \( S \), a unit vector \( \mathbf{u} \) and a scalar \( \lambda \) that satisfy
\[
S \times \mathbf{u} = \lambda \times \mathbf{u}
\]
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:

\[
X = U \times \Lambda \times V^T
\]

where \( U \) is a column-orthonormal \( N \times r \) matrix, \( r \) is the rank of the matrix \( X \), \( \Lambda \) is a diagonal \( r \times r \) matrix of the eigenvalues \( \lambda_i \) of \( X \), and \( V \) is a column-orthonormal \( n \times r \) matrix.

**Proof:** See (Press et al., 1992, p. 59).

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

Eq. 2 equivalently states that a matrix \( X \) can be brought in the following form, the so-called spectral decomposition (Jolliffe, 1986, p. 11):

\[
X = \lambda_1 \mathbf{u}_1 \times \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \times \mathbf{v}_2^T + \ldots + \lambda_r \mathbf{u}_r \times \mathbf{v}_r^T
\]

where \( \mathbf{u}_i \) and \( \mathbf{v}_i \) are column vectors of the \( U \) and \( V \) matrices respectively, and \( \lambda_i \) the diagonal elements of the matrix \( \Lambda \). Without loss of generality, we can assume that the eigenvalues \( \lambda_i \) are sorted in decreasing order. Fig. 2 illustrates the rotation of axes that SVD implies for an example where \( M=2 \) (i.e., 2-d points). The corresponding two directions \((x' \text{ and } y')\) that SVD suggests are shown, meaning that, if we are allowed only \( k=1 \) axis, the best direction to project 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.
It is confirmed by the SVD, which identifies them both: “blobs” of values, while the rest of the entries are zero. 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:

\[
\mathbf{X} = \mathbf{U} \times \mathbf{\Lambda} \times \mathbf{V}^T = \begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27 \\
\end{bmatrix} \times \begin{bmatrix}
9.64 & 0 & 5.29 \\
0 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71 \\
\end{bmatrix}
\]

or, in “spectral decomposition” form:

\[
\mathbf{X} = 9.64 \times \begin{bmatrix}
0.18 \\
0.36 \\
0.18 \\
0.90 \\
0 \\
0 \\
0 \\
\end{bmatrix} \times \begin{bmatrix}
0.58, 0.58, 0.58, 0, 0 \\
0, 0, 0, 0.71, 0.71 \\
\end{bmatrix} + 5.29 \times \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix} \times \begin{bmatrix}
0, 0, 0, 0, 0.71, 0.71 \\
0.53 \\
0.80 \\
0.27 \\
\end{bmatrix}
\]

Notice that the rank of the \( \mathbf{X} \) matrix is \( r=2 \); there are effectively 2 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” (that is, the group \{milk, bread, butter\}), and the “automotive concept” (that is, the group \{tire, oil\}). The intuitive meaning of \( \mathbf{U} \) and \( \mathbf{V} \) is as follows:

**Observation 1** \( \mathbf{U} \) can be thought of as the customer-to-concept similarity matrix.

**Observation 2** Symmetrically, \( \mathbf{V} \) is the product-to-concept similarity matrix.

For example, \( v_{1,3} = 0 \) means that the first product (milk) has zero similarity with the second concept (the “automotive concept”). \( \mathbf{V} \) contains Ratio Rules in its columns; in the above example, \( \mathbf{V} \) contains the following two rules:

- Customers typically spend \( 0.58 \times 0.58 \times 0 = 0 \) dollars on bread; milk; butter; tire; oil.
- Customers typically spend \( 0 \times 0 \times 0.71 \times 0.71 = 0 \) 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 \( \lambda_i \) elements of \( \mathbf{\Lambda} \) of the SVD of \( \mathbf{X} \). Moreover, the columns of \( \mathbf{V} \) are the eigenvectors of \( \mathbf{X}^T \times \mathbf{X} \).

\[
\mathbf{X}^T \times \mathbf{X} = \mathbf{\Lambda} \times \mathbf{\Lambda}^T \times \mathbf{V}^T
\]

**Proof:** See (Faloutsos, 1996). \( \square \)
The intuitive meaning of the $M \times M$ matrix $X^t \times X$ is that it gives the product-to-product similarities. In our example, we have the following product-to-product similarities:

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

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 $C = [c_{ij}]$ 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

$$C = X_c^t \times X_c$$

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

The following steps will compute the Ratio Rules in an I/O-efficient way: (a) zero-mean the input matrix to derive $X_c$, and simultaneously compute $C$ from Eq. 7 by updating partial sums; (b) compute the eigenvalues/vectors of $C$ and pick the first $k$. We assume that $C$ 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 (Agrawal et al., 1993b). Under this assumption, we can compute the column averages and the covariance matrix with a single pass over the $N$ (≈ millions) rows of the given $X$ matrix, using the algorithm of Fig. 3(a). Once we have $C$ in main memory, we can use any off-the-shelf eigensystem package to determine its eigenvalues and eigenvectors, as shown in Fig. 3(b).

3 If the number of columns are much greater than one thousand, as potentially might be the case in some market basket data analyses, then the methods from (Berry, Dumais, & O’Brien, 1995) could be applied to efficiently compute the eigensystem of the resulting sparse matrix.

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 $X$ from disk, during which partial sums are maintained and zero-mean centering is performed, and $O(NM^3)$ CPU operations to build the corresponding covariance matrix $C$. 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 (Agrawal & Shafer, 1996) 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 $N \gg M$.

### 4.4 Measuring the Goodness of a Rule-set: the “Guessing Error”

Let $R$ be a given set of rules. We would like to be able to assess how good $R$ 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 $R$ must allow for estimations of missing values in a given record/row.

Let’s consider a specific row (customer) $x_i$ 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 $R$ 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 $R$ on a data matrix $X$ is defined as the root-mean-square of the guessing errors of the individual cells, that is,

$$GE = \sqrt{\frac{1}{NM} \sum_i \sum_j (\hat{x}_{ij} - x_{ij})^2}$$

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.

**Definition 3** The “$h$-hole guessing error” for a set of rules $R$ on a data matrix $X$ is defined as the root-mean-square of the guessing errors of configurations of $h$ simultaneous cells, that is,
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 subsection 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 by the model 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. Fig. 4 gives
The set of unknowns are determined by the ratio rule, which is the following: the rules form a k-dimensional hyper-plane, on or close to which the data points lie. The h holes result in an h-dimensional hyper-plane b’ (= E_H × b^f) 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.

Fig. 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 (RR1). 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 V’ × x_concept = b’, from which the solution of 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 hyper-planes, which are illustrated in Fig. 5-7. Respectively, there are three possibilities regarding the equation from step 3 of the pseudocode,

\[ V' \times x_{\text{concept}} = b' \]

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

\[ x_{\text{concept}} = (V')^{-1} \times b' \]

Fig. 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 \(x_{\text{concept}}\).
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. In any
diag(1/μ_i) × R^T (13)
and, thus,

\[ x_{\text{concept}} = [V']^{-1} × b' \] (14)

Fig. 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 hyper-plane. 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. Fig. 7 illustrates an example in \( M = 3 \) dimensions, for \( h = 2 \) holes and cutoff \( k = 2.4 \).
event, note that AR are inherently unable to give extrapolations. Thus, we compared Ratio Rules with a straightforward technique for predicting values, named \textit{col-avgs: for a given hole, use the respective column average from the training set}. Note that \textit{col-avgs} is identical to the proposed method with $k=0$ eigenvalues.

Notice that linear regression, e.g., (Hou, 1996), 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 Sec. 6.3, association-based methods do not lead to prediction of missing values.

\textbf{Error Measure:} We use the $\text{GE}_h$ “guessing error” which was described in Sec. 4.4.

\textbf{Data sets:} We ran our experiments on a variety of real data sets (see Section 6.1 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;
- `abalone’ (4177 × 7) - physical measurements of an invertebrate animal, including length, diameter, and weights.\footnote{`baseball’ is available at www.usatoday.com/sports/baseball/sbstats.htm. \footnote{`abalone’ is available at www.ics.uci.edu/~mlearn/MLSummary.html.}}

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 (\textit{col-avgs}).

\begin{figure}[h]
\centering
\includegraphics[width=0.4\textwidth]{fig8.png}
\caption{Ratio of guessing error between RR and \textit{col-avgs}, for ‘nba’, ‘baseball’, and ‘abalone’.
\label{fig8}}
\end{figure}

\subsection{5.1 Reconstruction Accuracy}

Fig. 8 shows the $\text{GE}_h$ guessing error for the `nba’, `baseball’, and `abalone’ data sets, normalized by the guessing error attained by \textit{col-avgs}. As a frame of reference, we also present the normalized $\text{GE}_h$ of \textit{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 \textit{col-avgs}.

\subsection{5.2 Guessing Error for Simultaneous Holes}

In Fig. 9, we show $\text{GE}_h$ 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 $\text{GE}_h$ is constant with respect to $h$ for \textit{col-avgs} since the computation of $\text{GE}_h$ turns out to be the same for all $h$, for that method.
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.

<table>
<thead>
<tr>
<th></th>
<th>milk</th>
<th>bread</th>
<th>butter</th>
<th>tire</th>
<th>bulb</th>
<th>oil</th>
<th>shirt</th>
<th>pants</th>
<th>shoes</th>
<th>hat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Doe</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Johnson</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Taylor</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lee</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sullivan</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>noise1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>noise2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>noise3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) matrix of food, automotive and clothes groups

(b) first three Ratio Rules

<table>
<thead>
<tr>
<th>field</th>
<th>RR1</th>
<th>RR2</th>
<th>RR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>milk</td>
<td>.003</td>
<td>.018</td>
<td>.577</td>
</tr>
<tr>
<td>bread</td>
<td>.003</td>
<td>.018</td>
<td>.577</td>
</tr>
<tr>
<td>butter</td>
<td>.002</td>
<td>.018</td>
<td>.578</td>
</tr>
<tr>
<td>tire</td>
<td>.002</td>
<td>.576</td>
<td>.019</td>
</tr>
<tr>
<td>bulb</td>
<td>.003</td>
<td>.576</td>
<td>.019</td>
</tr>
<tr>
<td>oil</td>
<td>.003</td>
<td>.579</td>
<td>.015</td>
</tr>
<tr>
<td>shirt</td>
<td>.500</td>
<td>.001</td>
<td>.002</td>
</tr>
<tr>
<td>pants</td>
<td>.500</td>
<td>.003</td>
<td>.002</td>
</tr>
<tr>
<td>shoes</td>
<td>.500</td>
<td>.003</td>
<td>.002</td>
</tr>
<tr>
<td>hat</td>
<td>.500</td>
<td>.002</td>
<td>.002</td>
</tr>
</tbody>
</table>

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 3(a).

Table 3(b) shows the Ratio Rules for this type of matrix with 10,000 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 3(b) we see that the three ratio rules are essentially: RR1: shirt : pants : shoes : hat = 1 : 1 : 1 : 1, RR2: tire : bulb : oil = 1 : 1 : 1, and RR3: 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; RR1 represents the “clothes” group, RR2 the “automotive” group, and RR3 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 10,000 rows with the same items and groups. Fig. 10 displays the one-hole guessing error (GE1) 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.

![Guessing Error vs. Noise](image1.png)

**Fig. 10** Sensitivity analysis of ‘noise’ as a function of guessing error.

5.4 Scale-up

Fig. 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\textsuperscript{TM}. 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 100,000 \( \times \) 100 data matrix created using the Quest Synthetic Data Generation Tool.\textsuperscript{8} The methods were implemented in C and S-plus. The experiments ran on a dedicated Sun SPARCstation 5 with 32Mb of main memory, running SunOS 4.1.3. The disk drive was a Fujitsu M2266S-512 model ‘Cranel-M2266SA’ with minimum positioning time of 8.3ms and maximum positioning time of 30ms.

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.

![Scale-up](image2.png)

**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 (Srikant & Agrawal, 1996).

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 (a), the \( x \)-axis corresponds to the first (and strongest) rule \( RR_1 \); the \( y \)-axis corresponds to \( RR_2 \). In (b), 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. Fig. 13 shows 2-d plots for (a) ‘baseball’ and (b) ‘abalone’.

\textsuperscript{8} Quest is available at www.almaden.ibm.com/cs/quest/syndata.html.
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;

**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 (RR₁, RR₂, and RR₃) 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 RR₁ 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 ≈ 2:1 ratio. This is a Ratio Rule with the obvious interpretation: the average player scores 1 point for every 2 minutes of play (equivalently, 1 basket for every 4 minutes played). According to RR₁, Michael Jordan was by far the most active player in almost every category (see Fig. 12(a)). RR₂ shows that the number of rebounds is negatively correlated with points in a 0.489:0.199 ≈ 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, RR₂ 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 exam-
ple, we see, in Fig. 12(a), 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. 12(b)).

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Relative values of the RRs from ‘nba’.</th>
</tr>
</thead>
<tbody>
<tr>
<td>field</td>
<td>$RR_1$</td>
</tr>
<tr>
<td>minutes played</td>
<td>.808</td>
</tr>
<tr>
<td>field goals</td>
<td></td>
</tr>
<tr>
<td>goal attempts</td>
<td></td>
</tr>
<tr>
<td>free throws</td>
<td></td>
</tr>
<tr>
<td>throws attempted</td>
<td></td>
</tr>
<tr>
<td>blocked shots</td>
<td></td>
</tr>
<tr>
<td>fouls</td>
<td></td>
</tr>
<tr>
<td>points</td>
<td>.406</td>
</tr>
<tr>
<td>offensive rebounds</td>
<td></td>
</tr>
<tr>
<td>total rebounds</td>
<td></td>
</tr>
<tr>
<td>assists</td>
<td></td>
</tr>
<tr>
<td>steals</td>
<td></td>
</tr>
</tbody>
</table>

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 mollusk to one other; RR$_2$ points out a 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 Boolean and quantitative rules. Examples of each type of rule with which we are concerned follow:

- Boolean association rules (Agrawal et al., 1993b):
  \[
  \{bread, milk\} \Rightarrow butter
  \]
- Quantitative association rules (Srikant & Agrawal, 1996):
  \[
  bread : [2 - 5] \Rightarrow butter : [1 - 2]
  \]
- Ratio Rules:
  \[
  \text{ratio of spendings } bread:butter = \frac{2}{3}
  \]

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, e.g., 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. Fig. 15(a) illustrates 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

\[
\text{bread} : [3 - 5] \Rightarrow \text{butter} : [2 - 3]
\]

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

\[
\text{bread} : \text{butter} = .81 : .58
\]

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;
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 Section 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.

Acknowledgments

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


