Ratio Rules: A New Paradigm for Fast, Quantifiable Data Mining

Flip Korn, Alexandros Labrinidis, Yannis Kotidis
Department of Computer Science
University of Maryland
College Park, MD 20742
{flip,labrinid,kotidis}@cs.umd.edu

Christos Faloutsos*
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
christos@cs.cmu.edu

Abstract

Association Rule Mining algorithms operate on a data matrix (*e.g.*, customers $\times$ products) to derive *association rules* [2, 23]. 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 propose to use 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, say, butter. Thus, we can perform a variety of important tasks such as forecasting, answering "what-if" scenarios, detecting outliers, and visualizing the data. Moreover, we show how to compute Ratio Rules in a *single* pass over the dataset with small memory requirements (a few small matrices), in contrast to traditional association rule mining methods that require multiple passes and/or large memory. Experiments on several real datasets (*e.g.*, basketball and baseball statistics, biological data) demonstrate that the proposed method consistently achieves a "guessing error" of up to 5 times less than the straightforward competitor.

1 Introduction

Data mining has recently been receiving increasing interest [11], of which the quintessential problem is association rule mining [2]. Given a data matrix with, *e.g.*, customers for rows and products for columns, association rules find rules that describe frequently co-occurring products. Existing algorithms find rules of the form

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

meaning that customers who buy "bread" and "milk" also tend to buy "butter" with 90% confidence. What distinguishes database work from AI, Machine Learning and statistics work is its emphasis on large datasets. The initial association rule mining paper by Agrawal et al. [2], as well as all the follow-up database work [4], proposed algorithms to minimize the time to extract these rules through clever record-keeping to avoid additional passes over the dataset.

What is novel about the work in this paper is that it attempts to assess *how good* the derived rules are, an issue that has not been addressed at all in the database literature. We propose the "guessing error" as a measure of the "goodness" of a given set of rules for a given dataset. The idea is to pretend that a cell value (or values) of the matrix is "hidden" from us, and to try to guess the missing value(s) using the derived rules; the root-mean-square guessing error (averaged over all the cells of the given matrix) indicates how good a set of rules is.

*Work performed while at the University of Maryland. This research was partially funded by the Institute for Systems Research (ISR), and by the National Science Foundation under Grants No. EEC-94-02384, IRI-9205273 and IRI-9625428.

Proceedings of the 24th VLDB Conference
New York, USA, 1998

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

Customers typically spend $1:2:5$ *dollars on bread* : *milk* : *butter*.

Ratio Rules can be used for decision support by determining unknown (equivalently, hidden, missing or corrupted) values. We provide novel algorithms for estimating missing values, even if multiple values are simultaneously missing.

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

2 Related Work

Agrawal et al. distinguish between three data mining problems: identifying classifications, finding sequential patterns, and discovering association rules [1]. We review only material relevant to the latter, since it is the focus of this paper. See [9] for an excellent, recent survey of all three problems.

The seminal work of [2] introduced the problem of discovering association rules and presented an efficient algorithm for mining them. Since then, new serial algorithms [4, 16, 20] and parallel algorithms [3] have been proposed. In addition, generalized association rules have been the subject of recent work [22, 13].

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 and Agrawal [23], 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

$$bread : [3-5] \text{ and } milk : [1-2] \quad \Rightarrow \quad butter : [1.5-2]$$

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.

Traditional criteria for selecting association rules are based on the support-confidence framework [2]; recent alternative criteria include the chi-square test [7] and probability-based measures [21]. Related issues include outlier detection and forecasting. See [15] for a textbook treatment of both, and [5, 14, 8] for recent developments.

symbol	*definition*
N	number of records
M	number of attributes
k	cutoff (number of Ratio Rules retained)
h	number of holes
$\mathcal{H}$	set of cells which have holes
$\mathcal{R}$	set of rules
GE_1	guessing error over each hole
GE_h	guessing error over h holes
$\times$	matrix multiplication
$\mathbf{X}$	the $N \times M$ data matrix
$\mathbf{X}_c$	the centered version of $\mathbf{X}$
$\mathbf{X}^t$	the transpose of $\mathbf{X}$
$x_{i,j}$	value at row i, column j of the matrix $\mathbf{X}$
$\hat{x}_{i,j}$	reconstructed (approximate) value at row i and column j
$\bar{x}$	the mean cell value of $\mathbf{X}$
$\mathbf{C}$	the $M \times M$ covariance matrix ($\mathbf{X}_c^t \times \mathbf{X}_c$)
$\mathbf{V}$	the $M \times k$ RR matrix

Table 1: Symbols, definitions and notation.

3 Problem Definition

Ratio Rules can support the following applications, thanks to their ability to reconstruct missing values:

- 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 hyper-plane defined by the first 2 or 3 Ratio Rules, and plot the result, to reveal the structure of the dataset (*e.g.*, clusters, linear correlations, *etc.*).

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) [17], Principal Component Analysis (PCA) [15], Latent Semantic Indexing (LSI) [12], and the Karhunen-Loeve Transform

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

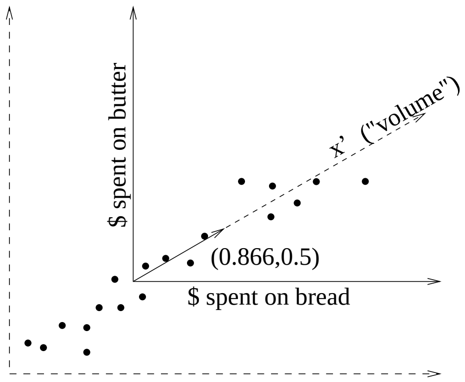


Figure 1: A data matrix in table form and its counterpart in graphical form, after centering (original axis drawn with dotted lines). As the graph illustrates, eigensystem analysis identifies the vector $(0.866, 0.5)$ as the "best" axis to project along.

(KLT) [10]. Eigensystem analysis involves computing the eigenvectors and eigenvalues of the covariance matrix of the given data points (see subsection 4.1 for the details). In subsection 4.2, 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 4.3 presents one of the two major contributions of this paper: the introduction of a measure for the "goodness" of a given set of rules. Subsection 4.4 presents the second major contribution: how to use the Ratio Rules to predict missing values.

4.1 Intuition Behind Ratio Rules

Figure 1 lists N customers and M products organized in an $N \times M$ matrix $\mathbf{X}$, where the entries are the dollar amount spent by customer i on product j. Table 1 gives a list of symbols used from here on and their definitions. To make our discussion more concrete, we will use rows and "customers" interchangeably, and columns and "products" interchangeably. 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 [19]), *etc.*

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 1 illustrates an example of an axis that this analysis finds. Suppose that we have M=2 dimensions; then our customers are 2-d points, as in Fig. 1. The corresponding direction x' this analysis suggests is shown, meaning that if we are allowed only k=1, the best direction to project on is the direction of x'. The direction x' is a *Ratio Rule* (RR for short) that governs the correlations between money spent on the products, based on customer purchasing activity. In this case, the projection of a data point on the x' axis gives the overall "volume" of the purchase. For the setting of Figure 1, the coordinates of the first RR = (0.866, 0.5) imply the rule "*bread* : *butter* $\Rightarrow$ \$0.866 : \$0.5"; that is, for the most of our customers (2-d points) 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.

Mathematically, the directions identified by eigensystem analysis are the eigenvectors of the *covariance matrix* $\mathbf{C}$ (see Eq. 2); each eigenvector has an associated *eigenvalue* whose magnitude indicates the variance of the points along that eigenvector.

The goal of this method is to reduce the dimensionality of a dataset 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 incrementally identifying the orthogonal direction with maximum variance (the second eigenvalue/vector, *etc.*). In the end, only the eigenvectors associated with the k largest eigenvalues, namely, the Ratio Rules, are kept. 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 cover 85% of the grand total [15, p. 94]. That is, choose the cutoff k such that

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

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

```
/* 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 (X[i][1],...,X[i][M]);
    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. 1;
    return the k largest eigenvectors;

complexity:
    O(M^3)
```

(b) Eigensystem computation

Figure 2: Pseudocode for efficiently computing Ratio Rules.

4.2 A Single-Pass Algorithm for Ratio Rules

The computation of Ratio Rules involves determining the eigenvectors of the covariance matrix $\mathbf{C}$ of the given $N \times M$ matrix $\mathbf{X}$. The covariance matrix $\mathbf{C} = [c_{ij}]$ intuitively is the "column-to-column" similarity matrix, which has high c_{ij} values if the columns i and j are correlated. Mathematically, it is defined as

$$\mathbf{C} \equiv \mathbf{X}_c^t \times \mathbf{X}_c \tag{2}$$

where $\mathbf{X}_c$ is derived by the given $\mathbf{X}$ matrix by subtracting the column average from every cell. That is, $\mathbf{X}_c$ is a zero-mean matrix, or "centered", 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 following steps will compute the Ratio Rules in an I/O-efficient way: (a) zero-mean the input matrix to derive $\mathbf{X}_c$; (b) compute $\mathbf{C}$ from Eq. 2; (c) 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 [2]. Under this assumption, we can compute the column averages and the covariance matrix with a single-pass over the N ($\approx$ millions) of rows of the given $\mathbf{X}$ matrix, using the algorithm of Figure 2(a). Once we have the covariance matrix $\mathbf{C}$ in memory, we can use any off-the-shelf eigensystem package to determine its eigenvalues and eigenvectors, as shown in Fig. 2(b).[1]

The proposed algorithm requires a *single pass* to compute the column averages and the covariance matrix. In more detail, it requires $O(N)$ disk operations to read the matrix $\mathbf{X}$ and $O(NM^2)$ main-memory operations to build the corresponding covariance matrix. Since the number of rows is typically in the hundreds of thousands (*e.g.*, sales, or customers), and the number of columns in the hundreds (*e.g.*, products, or patient symptoms), the algorithm of Fig. 2 is very efficient. Note that the algorithms of [3] require more than one pass over the dataset 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 $N \gg M$.

[1] 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 [6] could be applied to efficiently compute the eigensystem of the resulting sparse matrix.

4.3 Measuring the Goodness of a Rule-set: the "Guessing Error"

Let $\mathcal{R}$ be a given set of rules. We would like to be able to assess how good a given set of rules $\mathcal{R}$ 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 $\mathcal{R}$ 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 $\mathcal{R}$ and the rest of the values $x_{i,l}$ ($l \neq j$), 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 1 *The "single-hole guessing error", or simply the "guessing error", for a set of rules $\mathcal{R}$ on a data matrix $\mathbf{X}$ is defined as the root-mean-square of the guessing errors of the individual cells, that is,*

$$GE = \sqrt{\frac{1}{NM} \sum_{i}^{N} \sum_{j}^{M} (\hat{x}_{ij} - x_{ij})^2} \tag{3}$$

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_H^t;                           /* ``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_{H^c}]^t + d × [E_H]^t;
```

Figure 3: Pseudocode for filling holes.

Definition 2 *The "h-hole guessing error" for a set of rules $\mathcal{R}$ on a data matrix $\mathbf{X}$ is defined as the root-mean-square of the guessing errors of h cells* at a time, *that is,*

$$GE_h = \sqrt{\frac{1}{Nh|\mathcal{H}_h|}\sum_{i}^{N}\sum_{\mathcal{H}\in\mathcal{H}_h}\sum_{l\in\mathcal{H}}(\hat{x}_{i,l}-x_{i,l})^2} \qquad (4)$$

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

The way that $\mathcal{R}$ 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 dataset $\mathbf{X}$ to derive the rules $\mathcal{R}$ ("training set"), and some other portion $\mathbf{X}_{test}$ of the dataset $\mathbf{X}$ to compute the guessing error ("testing set"). The details of the choice of training and testing sets are orthogonal to our definition, and outside the scope of this paper, since they have been extensively examined in the machine learning and classification literature [18]. 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 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 $\mathcal{R}$ for a given testing dataset $\mathbf{Y}$ 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 dataset, a low "guessing error" implies that the derived rules have captured the essence of this dataset, 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.

4.4 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 its predicted value 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. Then we give the geometric intuition and show how the problem leads to a system of equations.

Definition 3 *An h-hole row vector* $\mathbf{b}_{\mathcal{H}}$ *is defined as a vector with holes (denoted with "?"s) at indices given in $\mathcal{H}$, where $\mathcal{H}$ is the set of "holes".*

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

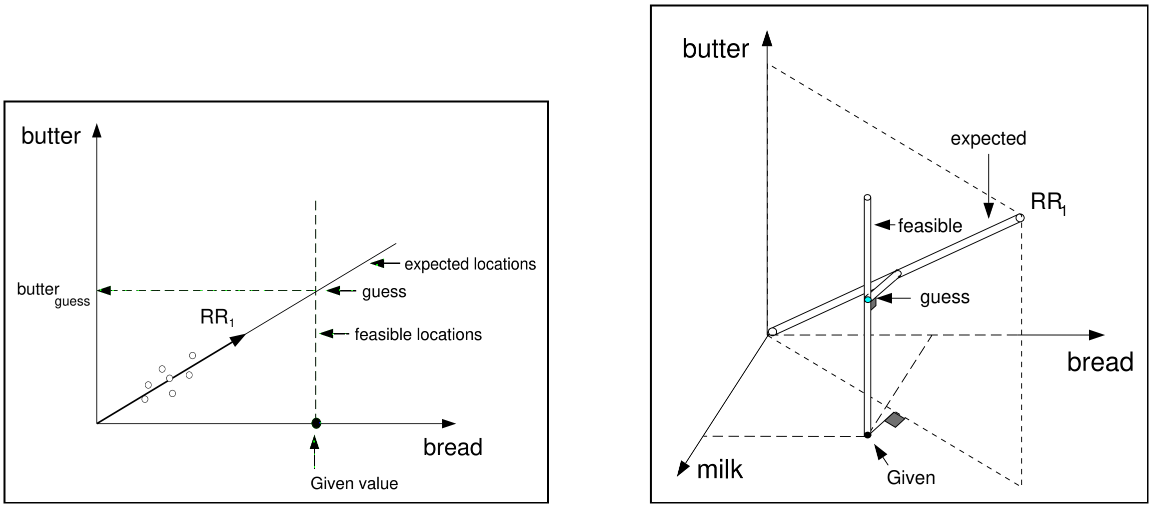
$$\mathbf{b}_{\{2,4\}} = [b_1, ?, b_3, ?, b_5]$$

Definition 4 *An* $(M-h)\times M$ elimination matrix $\mathbf{E}_{\mathcal{H}}$ *is defined as an $M\times M$ identity matrix with $h = |\mathcal{H}|$ rows removed, where the row indices are given in the set $\mathcal{H}$.*

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



(a) exactly-specified (b) over-specified

Figure 4: Two of the three possible cases: exactly defined, and over-specified

eliminate the "?"s from $\mathbf{b}_{\{2,4\}}$ as follows:

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

Once the user has specified partial knowledge from a transaction $\mathbf{b}_{\mathcal{H}}$ (*e.g.*, the dollar amounts spent by a new customer, for some products), the set of unknowns $\mathcal{H}$ are determined by the k Ratio Rules that have been kept, and are reported as $\hat{\mathbf{b}}$, that is, $\mathbf{b}_{\mathcal{H}}$ with the holes $\mathcal{H}$ filled in. The geometric intuition is the following: the rules form a k-dimensional hyper-plane $\mathbf{V}'$ ($= \mathbf{E}_{\mathcal{H}} \times \mathbf{V}$) 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 $\mathbf{b}'$ ($= \mathbf{E}_{\mathcal{H}} \times \mathbf{b}^t_{\mathcal{H}}$) 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 4(a) 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. Figure 3 gives the pseudo-code for the geometric description above.

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. 4-5. Respectively, there are three possibilities regarding the equation from step 3 of the pseudo-code,

$$\mathbf{V}' \times \mathbf{x}_{concept} = \mathbf{b}' \tag{5}$$

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

$$\mathbf{x}_{concept} = (\mathbf{V}')^{-1} \times \mathbf{b}' \tag{6}$$

Figure 4(a) 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 $\mathbf{V}'$ (see [17]). This uses the singular value decomposition of $\mathbf{V}'$:

$$\mathbf{V}' = \mathbf{R} \times diag(\mu_j) \times \mathbf{S}^t \tag{7}$$

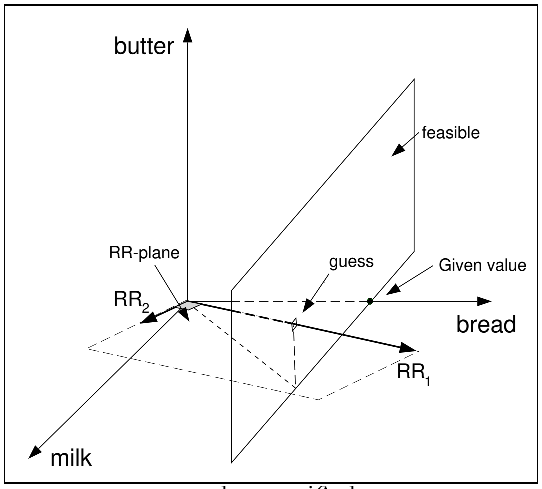
Since $\mathbf{V}'$ is singular, no inverse exists, but we can find a pseudo-inverse:

$$[\mathbf{V}']^{-1} = \mathbf{S} \times diag(1/\mu_j) \times \mathbf{R}^t \tag{8}$$

and, thus,

$$\mathbf{x}_{concept} = [\mathbf{V}']^{-1} \times \mathbf{b}' \tag{9}$$

Figure 4(b) illustrates an example in $M = 3$ dimensions, for $h = 1$ hole and cutoff $k = 1$.



under-specified

Figure 5: The last possible case: under-specified.

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. Figure 5 illustrates an example in $M = 3$ dimensions, for $h = 2$ holes and cutoff $k = 2$.

5 Experiments

We ran three sets of experiments. The first was to investigate the prediction accuracy achieved by the proposed method; the second was to examine the stability of Ratio Rules in estimating more than one simultaneous hole; the third was to see how our method scales up for large datasets.

Methods: 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. Multiple linear regression (*e.g.*, [14]) is remotely related to our proposed approach: it can predict missing values for a given, specified column of the data matrix, if everything else is known. Our method is more general because it can predict *arbitrary choices* of *arbitrary numbers* of missing columns, thanks to our technique in subsection 4.4. 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.

Error Measure: We use the GE_h "guessing error" described in Sec. 4.3.

Datasets: We ran our experiments on a variety of real datasets (see Section 6.1 for scatter-plots of them), described as follows:

- **'nba'** (459 $\times$ 12) - basketball statistics from the 1991-92 NBA season, including minutes played, field goals, rebounds, and fouls;
- **'baseball'** (1574 $\times$ 17) - batting statistics from Major League Baseball for four seasons; fields include batting average, at-bats, hits, home runs, and stolen bases;[2]
- **'abalone'** (4177 $\times$ 7) - physical measurements of an invertebrate animal, including length, diameter, and weights.[3]

Preliminary to running these experiments, for each dataset 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 Prediction Accuracy

Figure 7 shows the GE_1 guessing error for the **'nba'**, **'baseball'**, and **'abalone'** datasets, 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 datasets we tried and gave as low as one-fifth the guessing error of `col-avgs`.

5.2 Error Stability

In Fig. 6, we show GE_h for the **'nba'** and **'baseball'** datasets, for $1 \le h \le 5$ holes. The results for the **'abalone'** dataset were similar, and are omitted for

[2] The **'baseball'** dataset is available at http://www.usatoday.com/sports/baseball/sbstats.htm.

[3] The **'abalone'** dataset is available at http://www.ics.uci.edu/~mlearn/MLSummary.html.

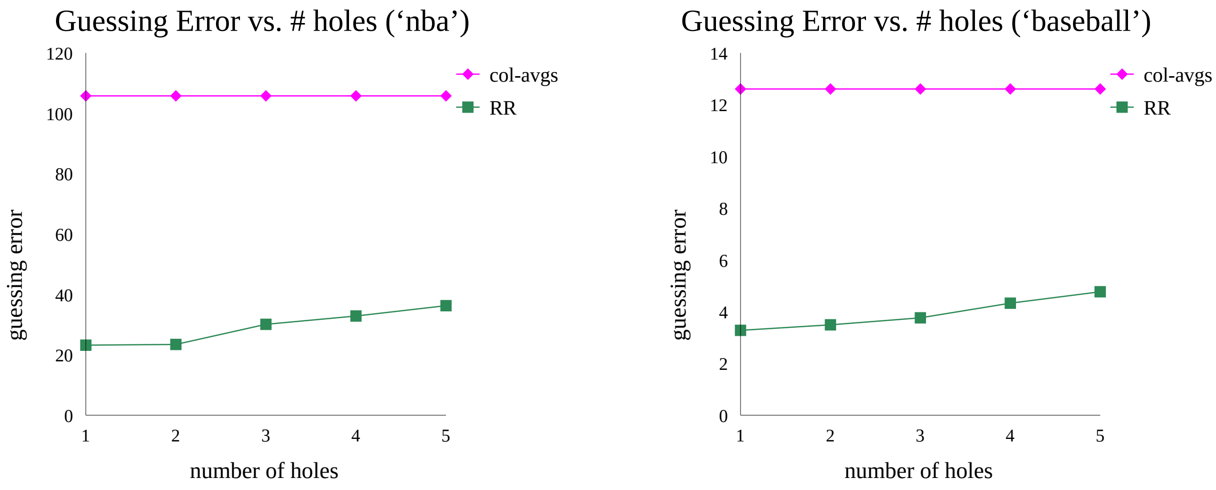


Figure 6: Guessing error vs. number of holes (1-5) for the **'nba'** and **'baseball'** datasets

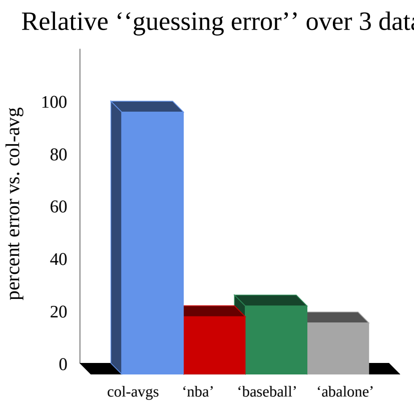


Figure 7: Ratio of guessing error between RR and `col-avgs` for **'nba'**, **'baseball'**, and **'abalone'**.

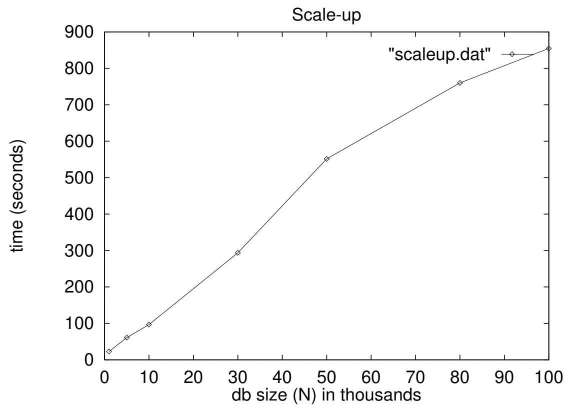


Figure 8: Scale-up: time to compute RR versus db size N in records.

brevity. Note that the guessing error is relatively stable for up to several simultaneous holes. Note that GE_h is constant with respect to h for colavgs since the computation of GE_h turns out to be the same for all h, for that method.

5.3 Scale-up

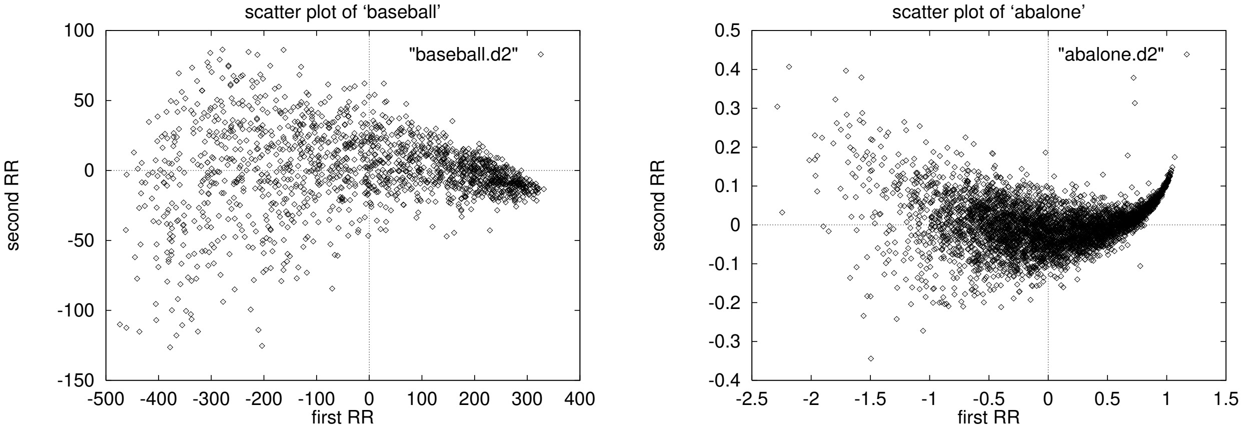
Figure 8 demonstrates the scale-up of our algorithm. The vertical axis is the average actual computation time to determine the Ratio Rules (in seconds), as measured by the `time` utility of UNIX^{TM} . The horizontal axis is the number of data matrix rows N. Since all of our datasets are relatively small ($N < 5000$) for this experiment, we used a 100,000 $\times$ 100 data matrix created using the Quest Synthetic Data Generation Tool.[4] The methods were implemented in `C` and `Splus`. 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.

6 Discussion

Here we show the visualization capabilities that Ratio Rules offer by presenting 2-d scatter-plots of the datasets used. Using the **'nba'** dataset, 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 association rules [23].

[4] Quest is available at
http://www.almaden.ibm.com/cs/quest/syndata.html.



(a) **'baseball'** (b) **'abalone'**

Figure 9: Scatter plots of (a) **'baseball'** and (b) **'abalone'** in 2-d RR space.

6.1 Visualization

Recall that Ratio Rules identify the axes of greatest variation. Similar to 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 dataset. For example, Figure 11 shows a scatter-plot of **'nba'**, which originally included the statistics 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. Figure 9 shows 2-d plots for (a) **'baseball'** and (b) **'abalone'**.

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

Figure 10: Interpretation of Ratio Rules.

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			$-.07$

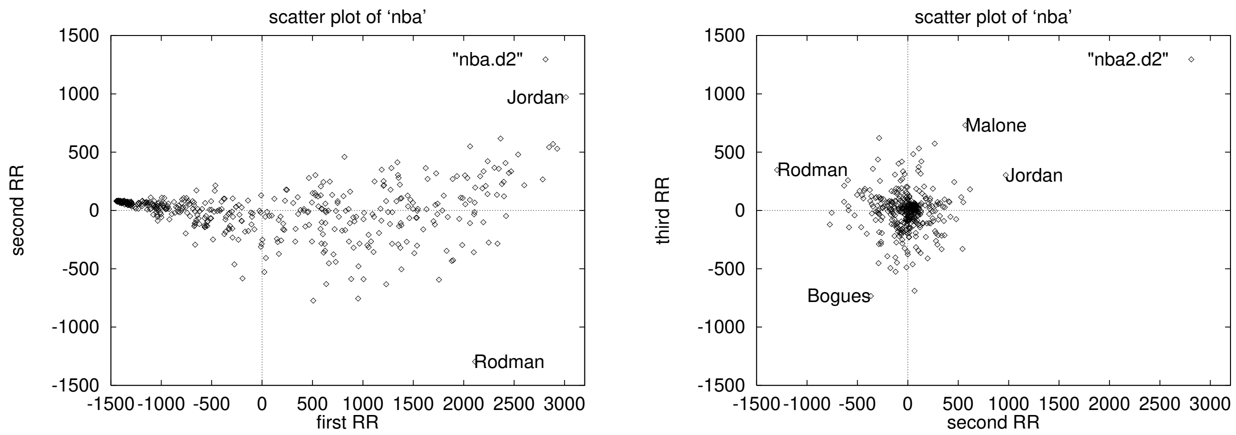
Table 2: Relative values of the RRs from **'nba'**.

6.2 Interpretation of the Ratio Rules

In this section, we illustrate by example how Ratio Rules can be interpreted as meaningful rules. The methodology is outlined in Figure 10.

Table 2 presents the first three Ratio Rules (RR_1, RR_2, and RR_3) for the **'nba'** dataset, whose fields include minutes played, fields goals, offensive rebounds, defensive rebounds, assists, and steals, among others.

By drawing on a basic knowledge of basketball and by examining these Ratio Rules, we conjecture the following: RR_1 represents "court action", 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 minutes of play (equivalently, 1 basket for every 4 minutes played). According to RR_1, Michael Jordan was by far the most active player in almost every category (see Fig. 11(a)). 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 "field position", separating the guards, who get



(a) side view

(b) front view

Figure 11: A scatter plot of **'nba'**: two 2-d orthogonal views.

the most opportunities to shoot, from the forwards, who are more likely to be rebounders. For example, in Fig. 11(a), we see 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 "height", with Mugsy Bogues (5'3") and Karl Malone (6'8") at opposite extremes (see Fig. 11(b)).

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 [2]:
  $\{bread, milk\} \Rightarrow butter$
- quantitative association rules [23]:
  $bread : [2-5] \quad \Rightarrow \quad butter : [1-2]$
- 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 $\mathbf{X}$ 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. Figure 12(a) illustrates how these rules might work for a fictitious dataset with a few customers (points) and $M = 2$ products only, namely, "bread" and "butter". In this dataset, 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

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

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

Ratio Rules, for the same setting of Figure 12 and with $k = 1$ rule, will fit the best possible line through the dataset; its unit vector is exactly the first rule of the given data matrix. Thus, the corresponding rule will be

$$bread : butter = .81 : .58$$

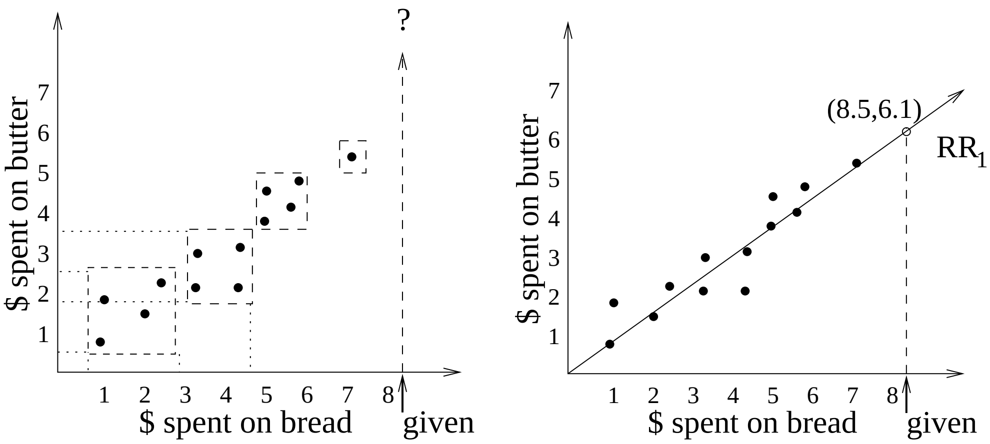
For the remaining discussion, we focus only on quantitative association rules since the focus is on real-valued data such as dollar amounts spent by customers on products. We compare the strengths of quantitative association rules with those of Ratio Rules.

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 Figure 12, or as in the real datasets 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;



(a) Quantitative (b) Ratio Rules

Figure 12: Illustration of rules from a fictitious dataset 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.

- They can perform extrapolations and predictions. For example, in Figure 12, 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 Figure 12(b) 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 dataset;
- They are easily implemented, thanks to highly fine-tuned eigensystem packages; 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 Section 4.4; 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 datasets, which showed that the proposed Ratio Rules scale-up for large datasets, and can achieve up to 5 times smaller guessing error than the competitor. Future research could focus on applying Ratio Rules to datasets 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'** dataset. We would also like to thank Rakesh Agrawal for offering us his synthetic dataset generator, and Mike Franklin for providing an RS/6000 to install and run the generator.

References

[1] 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.

[2] 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.

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

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

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

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

[20] 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.

[21] 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.

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

[23] 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.