Open access • Journal Article • DOI:10.1109/TKDE.2002.1019208

Unsupervised learning with mixed numeric and nominal data — Source link

C. Li, Gautam Biswas

Institutions: Middle Tennessee State University, Vanderbilt University

Published on: 01 Jul 2002 - IEEE Transactions on Knowledge and Data Engineering (IEEE Educational Activities Department)

Topics: Similarity measure, Similarity (network science), Cluster analysis, Hierarchical clustering and Conceptual clustering

Unsupervised Learning with Mixed Numeric and Nominal Data

Cen Li and Gautam Biswas, *Senior Member*, *IEEE*

Abstract—This paper presents a Similarity-Based Agglomerative Clustering (SBAC) algorithm that works well for data with mixed numeric and nominal features. A similarity measure, proposed by Goodall for biological taxonomy [15], that gives greater weight to uncommon feature value matches in similarity computations and makes no assumptions of the underlying distributions of the feature values, is adopted to define the similarity measure between pairs of objects. An agglomerative algorithm is employed to construct a dendrogram and a simple distinctness heuristic is used to extract a partition of the data. The performance of SBAC has been studied on real and artificially generated data sets. Results demonstrate the effectiveness of this algorithm in unsupervised discovery tasks. Comparisons with other clustering schemes illustrate the superior performance of this approach.

Index Terms—Agglomerative clustering, conceptual clustering, feature weighting, interpretation, knowledge discovery, mixed numeric and nominal data, similarity measures, χ^2 aggregation.

1 Introduction

The widespread use of computers and information technology has made extensive data collection in business, manufacturing and medical organizations a routine task. The primary challenge that drives the relatively new field of data mining or knowledge discovery from databases is the extraction of potentially useful information by careful processing and analysis of this data in a computationally efficient and sometimes interactive manner [21]. Frawley et al. [12] define knowledge discovery to be *"the non trivial extraction of implicit, previously unknown and potentially useful information in data."* This suggests a generic architecture for a discovery system. At the core of the system is the discovery engine, which computes and evaluates groupings, patterns, and relationships using a relevant set of features selected in the context of a problem solving task [2]. Depending on the discovery engine employed in the system, the results can be further analyzed to derive models as rules, analytic equations, and concept definitions under the chosen context (e.g., see [21]). Typically, the discovery engine is powered by one of two mechanisms:

- **Supervised learning schemes**, which assume that class labels of the data objects being analyzed are known. The process extracts rules/models that identify groups of objects with the same class label while differentiating among the objects that have different labels. The rules form the basis for classifying new data observations.
- **Unsupervised learning or clustering schemes**, which make no assumptions about the category structure. They use objective *criterion functions* to define similarity or dissimilarity among objects. The goal is to find structure in the form of "natural groups," i.e., to partition the data into groups where objects that are more similar tend to fall into the same group and objects that are relatively distinct tend to separate into different groups.

Both classification and clustering schemes require data objects to be defined in terms of a predefined set of features. Features represent properties of the object that are relevant to the problem solving task. For example, if we wish to classify automobiles by speed and power, body weight, body shape, and engine size are relevant features, but the color of the car body is not. Selection of appropriate features is an important task in clustering and classification applications, but we do not elaborate on that task in this paper.

Clustering methodologies, illustrated in Fig. 1, incorporate three main steps [2]: preprocessing, clustering, and interpretation. Data preprocessing involves the selection of relevant features for the analysis task and the characterization of individual features at the right level of granularity. The clustering step constructs a flat or hierarchical partitioning of the objects based on an objective criterion function and a control algorithm. The explanation step assigns meaning to the groups or structures discovered based on the feature value definitions associated with each group. The interpretation process is often governed by characteristics such as parsimony and completeness. Domain knowledge may be employed to assist the interpretation process.

Traditional clustering methodologies [7], [17] assume features are numeric valued, but as application areas have grown from the scientific and engineering domains to the medical, business, and social domains, one has to deal with features, such as gender, color, shape, and type of disease,

- *C. Li is with the Department of Computer Science, Middle Tennessee State University, Murfreesboro, TN 37132. E-mail: cli@mtsu.edu.*
- *G. Biswas is with the Department of Electrical Engineering and Computer Science Box 1679 Station B, Vanderbilt University, Nashville, TN 37325. E-mail: biswas@vuse.vanderbilt.edu.*

Manuscript received 22 July 1996; revised 02 Sept. 1998; accepted 27 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 104312.

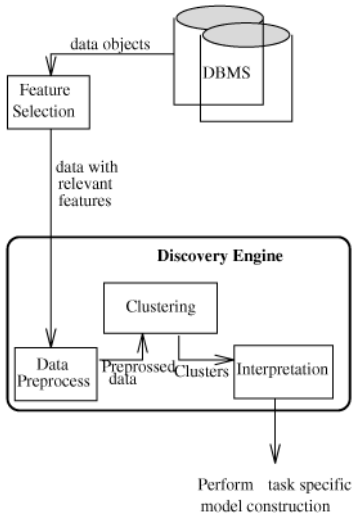

Fig. 1. An unsupervised classification discovery engine.

that are nominal valued. Schemes for processing nominal valued data, typically called conceptual clustering [10], combine clustering with the concept formation and interpretation tasks. In the real world, a majority of the useful data is described by a combination of numeric and nominal valued features. Attempts to develop criterion functions for mixed data have not been very successful, mainly because of the differences in the characteristics of these two kinds of data. To overcome this problem, pragmatic approaches have been adopted. Most solutions attempted fall into one of the following categories:

- Encode nominal attribute values as numeric integer values and apply distance measures used in numeric clustering for computing proximity between object pairs. In many cases, the translation from nominal data to numeric does not make sense and the proximity values are hard to interpret.
- Discretize numeric attributes and apply symbolic clustering algorithms (e.g., [2]). The discretization process often causes loss of important information especially the relative (or absolute) difference between values for the numeric features.
- Generalize criterion functions designed for one type of feature to handle numeric and nonnumeric feature values (e.g., [5], [22], [26]). The primary difficulty in doing this can be attributed to the nature of the criterion functions used. Criterion functions used in symbolic clustering are based on probability distribution functions. For nominal valued attributes, probability distributions can be approximated by simple counting schemes. Computing and bounding density estimates for numeric valued attributes are difficult. This method has been implemented in some systems, but experimental results discussed later in this paper show they produce better results for pure nominal valued data than they do for mixed or pure numeric valued data. Criterion functions used in numeric clustering are based on distance metrics that can not be extended to nominal attributes except in a superficial way.

The focus of this paper is on developing unsupervised learning techniques that exhibit good performance with mixed data. The core of the methodology is based on a similarity measure from biological taxonomy proposed by Goodall [15]. The measure processes numeric and nominal valued attributes within a common framework and can be conveniently coupled with an agglomerative control strategy that constructs a hierarchy or dendrogram, from which a distinct partition is extracted by a simple procedure that trades off distinctness for parsimony.

The rest of the paper is organized as follows: Section 2 reviews existing criterion functions and clustering systems. Section 3 presents our clustering system SBAC, with its two components: 1) the similarity measure and 2) the agglomerative control structure. Section 4 demonstrates the effectiveness of the SBAC system by comparative empirical studies of its performance. General discussion and the conclusions of this work follow in Section 5.

2 Background

Traditional approaches to cluster analysis (numerical taxonomy) represent data objects as points in a multidimensional metric space and adopt *distance metrics*, such as Euclidean and Mahalanobis measures, to define similarity between objects [7], [17]. On the other hand, conceptual clustering systems use conditional probability estimates as a means for defining the relation between groups or clusters. Systems like COBWEB [8] and its derivatives (e.g., [3], [22], [26]) use the *Category Utility (CU)* measure [14], which has its roots in information theory. The measure partitions a data set in a manner that maximizes the probability of correctly predicting a feature value given group C_k. Systems like WITT [16] use *correlation* measures to define groupings. These measures are tailored for nominal attributes, though variations, such as COBWEB/3 [22] and ECOBWEB [26] use modifications of the CU measure to handle numeric attributes. AUTOCLASS [5] uses a *finite mixture model* and derives groupings of objects that locally maximize the posterior probability of individual clusters given the feature distribution functions. The COBWEB/3, ECOBWEB, and AUTOCLASS systems are discussed in greater detail below.

2.1 COBWEB/3

The category utility function [14] defines a probability matching strategy to measure the usefulness of a class in correctly predicting feature values:

$$CU_k(nominal) = P(C_k) \sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2,$$

where $P(A_i = V_{ij})$ is the unconditional probability of attribute A_i taking on value V_{ij} and $P(A_i = V_{ij} \mid C_k)$ is the conditional probability of $A_i = V_{ij}$ given class C_k. The difference, $\sum_i \sum_j P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2$, represents an increase in the number of attribute values that can be predicted correctly for class k versus the expected number of correct predictions given no class information. The partition score, i.e., the utility of a partition structure

made up of K classes, is defined as the average CU over the K classes:

$$\frac{\sum_{k=1}^{K} CU_k}{K.}$$

COBWEB/3 combines the original COBWEB [8] algorithm with the methodology defined in CLASSIT [13] to handle numeric attributes in the CU measure. For numeric attributes, probabilities are expressed in terms of the probability density function(pdf) defined for the range of values that can be associated with the attribute. COBWEB/3 assumes that the numeric feature values are normally distributed and computes the $\sum_j P(A_i = V_{ij} \mid C_k)^2$ term in the following manner:

$$\int_V P^2(A_i = V \mid C_k)dv \equiv \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma_{ik}{}^2} e^{-\left(\frac{x-\mu_{ik}}{\sigma_{ik}}\right)^2} dv \equiv \frac{1}{2\sqrt{\pi}\sigma_{ik}},$$

where μ_{ik} and σ_{ik} are the mean and standard deviation of feature i in class k, respectively. Similarly, the unconditional probability, $\sum_j P(A_i = V_{ij})^2$, is computed as $\frac{1}{2\sqrt{\pi}\sigma_i}$, where σ_i is the standard deviation for feature A_i over the entire data set. With the direct interpolation of the criterion function, the counting scheme used for estimating probabilities for nominal features is switched to estimating the mean and the standard deviation for numeric features.

A problem arises with this approach as the range of possible values for a feature A_i narrows, i.e., σ_{ik} (or σ_i) becomes small. In the extreme case, when all objects in class C_k have a unique value, $\sigma_{ik} = 0$ and $\frac{1}{\sigma_{ik}} \to \infty$. Therefore, $CU_k \to \infty$. COBWEB/3 introduces a practical solution to this problem in the form of an additional parameter called *acuity*, $\frac{1}{\sigma}$. This value bounds the value of σ to be used in the CU_k computation. When σ_{ik} for a numeric attribute i and class k becomes less than 1($\sigma_{ik} < 1$), the acuity threshold is applied and $\frac{1}{\sigma_{ik}}$ is set to 1.

For the set of numeric features, CU is defined as:

$$CU_k(numeric) = \frac{P(C_k)}{2\sqrt{\pi}} \sum_i \frac{1}{\sigma_{ik}} - \frac{1}{\sigma_i}.$$

The overall CU is the sum of the CU contributions from nominal and numeric features:

$$CU_k = CU_k(nominal) + CU_k(numeric).$$

The nominal form of the CU function represents a well-understood information theoretic measure for evaluating the goodness of a partition structure. However, the numeric form of the CU measure has a number of limitations. First, it assumes that feature values are normally distributed. This may not hold for all data sets. Second, the mean and variance of the feature distribution are estimated using the sample mean and variance. When the sample populations are small, the accuracy of the estimate is suspect. Also, when feature values converge, σ_{ik} becomes small, but the computation is bounded by an ad hoc mechanism, the acuity measure. All this makes it hard to impose bounds the accuracy of the CU measure for numeric features.

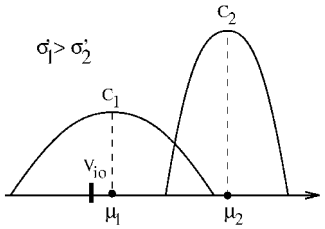

Fig. 2. A special case scenario for COBWEB/3.

Another limitation of the numeric CU measure is that it does not take into account the actual distance between object values in determining class structure. This can produce counterintuitive results as shown by the example in Fig. 2. Consider two intermediate clusters C_1 and C_2. Numeric feature A_i has mean μ_1 and variance σ_1 for C_1, and mean μ_2 and variance σ_2 for C_2. Also, $\mu_1 < \mu_2$ and $\sigma_1 > \sigma_2$. A new object to be clustered has value V_{i0} for A_i, where V_{i0} is closer to μ_1 than μ_2. When the new object is added to C_1, the cluster variance becomes σ_1' and when it is added to C_2, the variance is σ_2'. If $\sigma_1' > \sigma_2'$, then $CU_1 < CU_2$, so the object will be placed in C_2 despite the fact that $|V_{i0} - \mu_1| < |V_{i0} - \mu_2|$.

2.2 ECOBWEB

ECOBWEB is another extension of the COBWEB system. ECOBWEB attempts to remedy some of the disadvantages of the COBWEB/3 interpolation scheme:

- the normal distribution assumption for the pdf of numeric features and
- the acuity value for bounding the CU contribution from numeric features.

In the ECOBWEB approach, the probability distribution for numeric features is approximated by the probability distribution about the mean for that feature.

For numeric features, the ECOBWEB approach captures the essence of the CU function by introducing an approximation:

$$\sum_j P(A_i = V_{ij} \mid C_k)^2 \approx P(A_i = \bar{V}_i \mid C_k)^2,$$

i.e., the expected score is estimated around the mean, $\bar{V}_i$, of the feature value distribution. This is computed by

$$P(A_i = \bar{V}_i \mid C_k)^2 = \sum_i \left(\int_{-I_{ik}}^{I_{ik}} p_{ik}(v)dv \right)^2,$$

where $\bar{V}_i$ is the mean value for feature i in cluster k and I_{ik} is a designated interval range around $\bar{V}_i$. For a set of N numeric features, the CU measure is defined as:

$$CU_k = P(C_k) \sum_i^N \left(\int_{-I_{ik}}^{I_{ik}} p_{ik}(v)dv \right)^2 - \left(\int_{-I_i}^{I_i} p_i(v)dv \right)^2.$$

The choice of the interval size, I_{ik}, has a significant effect on the CU computation. The simplest definition of the interval size I_i is a fixed one, where

$$2I_i = \frac{\text{expected range of attribute values of } A_i}{\text{expected number of distinct intervals of attribute } A_i}.$$

TABLE 1
Three Methods Implemented in ECOBWEB for Defining the Interval Value

Method	Interval Value
static	$\frac{\text{expected range of the attribute values of } A_i}{\text{expected number of distinct intervals of attribute } A_i}$
dynamic	$\frac{2*\sigma_i}{\text{expected number of distinct intervals of attribute } A_i}$
adaptive	$\frac{\sqrt{2*\sigma_i * \text{expected range of attribute values of } A_i}}{\text{expected number of distinct intervals of attribute } A_i}$

This fixed interval size is used through the entire clustering process. As the clustering hierarchy grows deeper, the range of the attribute values within each class may become narrower, therefore, these distributions become sharper and more peaked. If the interval value, $2I_i$, is chosen to be too large, the $P(A_i = V_{ij} \mid C_k)$ will always evaluate to 1 and this methodology fails to differentiate contributions of features with narrow distributions lower down in the hierarchy. On the other hand, if the value chosen results in a very narrow interval, the probability about the means of two distributions can be equal even though the actual distributions may differ dramatically. To address these problems, ECOBWEB defines two other methods for computing the interval values: 1) a dynamic approach, where interval width is a function of the feature value variance and 2) an adaptive approach, which uses the geometric mean of the static and dynamic approaches in computing interval size.

Table 1 lists the three schemes that have been used to generate the $2I_i$ value. All of them rely on a user defined parameter, n, the "expected number of distinct intervals of property attribute A_i." Since the choice of the $2I$ value is likely to have a very significant effect on the performance of the system, it is important to see how the two user controlled parameters: 1) the *method* for calculating $2I_i$ and 2) n, affect the structure of the clustering hierarchy. A cross-comparative study is performed, where the system is run with different combinations of the *method* and the *n* values. The well-known IRIS[1] data set is used in this study. The top level partitions were extracted from the concept tree and their structural differences were compared using the *percentage of matched objects* measure. This measure computes the total number of common objects among corresponding clusters across two partitions created from two different runs. The computed number divided by the total number of objects in the data set, a value between 0 and 1, expresses the *percentage of matched objects* between the two partitions. The higher this value, the greater is the correspondence between the two partitions. A cross tabulation between pairs of partitions is listed in Table 2. In Table 2a, n was kept fixed at 8[2] and the three *methods* static, dynamic, and adaptive were applied. The pairwise similarity measures indicate a significant difference in the groupings for the three methods. In Table 2b, the *method* was kept fixed(static) and the value of n was varied. Results are shown for $n = 2, 8$, and 16.[3] The higher similarity scores indicate that the partition structures are less sensitive to changes in n, though there are still significant structural differences in the concept hierarchies generated. These results demonstrate that the clustering structures generated by ECOBWEB are very dependent on the two user defined parameters. This may not be an issue when the hierarchical structure is used for prediction purposes [25], but it is not a desirable characteristic in knowledge discovery tasks.

TABLE 2
Comparisons of Top Level Partitions Generated by ECOBWEB on IRIS Data with Different Combinations of n and *Method* Values

Method	static	dynamic	adaptive
static	1	0.187	0.333
dynamic		1	0.187
adaptive			1

(a)

n	2	8	16
2	1	0.707	0.613
8		1	0.667
16			1

(b)

2.3 AUTOCLASS

AUTOCLASS [5] imposes a classical finite mixture distribution model on the data and uses a Bayesian method to derive the most probable class distribution for the data given prior information(PI). The finite mixture model contains two parts: 1) the class probability, $P(X_i \in C_j \mid PI)$, the probability that object X_i belongs to class C_j independent of information about the object, but based on prior information about the allowed search space, the distribution assumptions, and the distribution parameters and 2) the intraclass pdf, $P(\vec{X}_i \mid X_i \in C_j, PI)$, the probability of observing the instance attribute values $\vec{X}_i$ given object $X_i \in C_j$. The intraclass mixture pdf is a product of individual or covariant attribute pdfs, such as the Bernoulli distributions for nominal attributes, Gaussian distributions for numeric attributes, and Poisson distributions for number counts.

Bayes rule can be applied to combine these two probabilities, i.e.,

$$P(\vec{X}_i, X_i \in C_j \mid PI) = P(X_i \in C_j \mid PI) \cdot P(\vec{X}_i \mid X_i \in C_j,\ PI),$$

from which the probability of observing an instance X_i with attribute value $\vec{X}_i$, regardless of class, is derived by summing over all classes. Then the probability of observing the entire data set is derived by taking the product over all instances, i.e.,

$$P(X \mid PI) = \prod_i \left(\sum_j P(X_i \in C_j \mid PI) \right) \cdot P(\vec{X}_i \mid X_i \in C_j, PI).$$

Assuming the model, T, represents the feature distribution in the data and $\vec{V}$ the parameter values of the model, the

1. The Iris data set, chosen for the study contains 150 objects. Each object is described by four numeric valued attributes: sepal length, sepal width, petal length, and petal width. The objects are equally distributed in the three classes: setosa, versicolor and virginica. From previous studies, it is known that the setosa class is distinct, but versicolor and virginica are mixed.

2. $n = 8$ is the default value in the system for the "expected number of intervals for a property." Experimental results in [25] indicated that *method*=static generated the best results for the ECOBWEB system.

3. In the rest of the paper, $n = 8$ and *method* = static are used in experiments with ECOBWEB.

Bayesian clustering method derives the clustering partition model with the highest posterior probability

$$P(\vec{V}, T \mid X) = \frac{P(\vec{V}, X \mid T)P(T)}{P(X)}.$$

For model selection purposes, this can be approximated by $P(\vec{V}, X \mid T)$, which is expressed by:

$$P(X, \vec{V} \mid T) = P(\vec{V} \mid T) \cdot P(X \mid \vec{V}, T).$$

The second term in the right-hand side of this equation corresponds to the $P(X \mid PI)$ term described above.

Assuming the number of classes for the data is known, the goal of AUTOCLASS is to maximize the posterior probability of the clustering partition model given the data. With no information on class membership, direct optimization approaches for finding the best parameter values from mixture pdfs are computationally intractable even for moderate size data sets. A variation of Dempster's EM algorithm [6] is used to approximate the solution. Weighted instance assignments and weighted statistics, calculated from normalized class probability and data instances, are used to represent known assignment statistics. This allows locally optimal estimation of the parameter values. Weighted instance assignments and weighted statistics can be updated using the newly estimated parameter values, which in turn allow for re-estimation of locally optimal parameter values. Cycling between these two steps moves the current parameter values and weight estimates toward a mutually predictive and locally maximal stationary point.

Once multiple locally maximum parameter value sets are estimated for a given classification pdf, the posterior probability of the current classification pdf can be approximated using the local statistics derived for those parameter value sets. The posterior probability of the current classification pdf is recorded. Then the attribute pdf is selectively changed in each class. This initiates another round of the estimation process, which involves finding the parameter value sets and the posterior probability of this new classification pdf. The classification pdf that has the highest posterior probability is retained at the end of this iterative process.

The above description assumes that the number of classes in the data is known in the repetitive process of searching for the best $P(\vec{V} \mid PI)$ and $P(T \mid PI)$, where T denotes the abstract mathematical form of the pdf. For completeness, another level of search for the optimal number of classes has to be added to this two-step process [4]. AUTOCLASS starts out with a number of classes J that is smaller than the predicted true number of classes for the data. If the resulting classes have significant probability, the number of classes are progressively increased. This process is terminated when the partition structure includes classes with negligible posterior probability. The classes with small posterior probability are dropped and the remaining classes represent an optimal classification of the data given the assumed class model function.

Despite the fact that a number of approximate optimization methods have been implemented in the AUTOCLASS system, the computational complexity required by the nested three level search process is extremely high. It often took us days and sometimes weeks, to get the system to generate meaningful results for moderate sized data sets on SPARC station architectures. The computational complexity is directly attributable to the exhaustive search approach.

AUTOCLASS also suffers from the over fitting problem associated with the maximum likelihood optimization methods for probabilistic models. It has been suggested that "Occam's Factor" comes into play when considering different partition models (i.e., partitions with larger number of clusters have smaller prior probabilities), which limits the degree of over fitting. However, our experimental results show that this phenomenon is not strong enough to prevent AUTOCLASS from producing many more classes than some of the other unsupervised classification methods, especially when numeric features dominate the class structure.

3 The SBAC System

From our earlier discussion, it is clear that the clustering process is primarily governed by two factors:

1. The *criterion function* for evaluating the goodness of a partition structure. Examples are the mean square error used in numeric partitioning schemes, the category utility measure used in COBWEB and related systems and similarity and correlational measures used in agglomerative algorithms. Section 2 discussed two CU-based approaches and the Bayesian AUTOCLASS scheme that can be applied to mixed data.
2. The *control algorithm* for generating the partition structure. Control algorithms can be partitional or hierarchical. Partitional algorithms derive a flat structure, whereas, hierarchical schemes form cluster in a bottom up (agglomerative) or top down (divisive) manner. AUTOCLASS uses a partitional control structure and the COBWEB-based algorithms employ divisive control schemes. All three algorithms described in Section 2 employ divisive control algorithms. The COBWEB-based algorithms are incremental, so the clusters formed are a function of the data ordering [9].

The SBAC methodology uses a similarity measure defined by Goodall [15] and adopts a hierarchical agglomerative approach to build partition structures. Given a pairwise similarity (or dissimilarity) matrix of the objects to be clustered, SBAC can use the single, complete link, and group-average methods [17] to perform the aggregation process. In this paper, we use the group average method, because empirical studies showed that it produced the best results.

3.1 The Similarity Measure

A number of similarity measures have been used as measures of proximity [17]. Examples are

- matching coefficients for nominal valued data,
- correlation measures for numeric data, and
- s- and p-proximity measures from fuzzy logic theory for mixed nominal and numeric data [31].

The p-measure uses an exponential function based on differences between feature values, to compute the proximity between two objects. The s-function employs prespecified fuzzy membership functions. Zemankova and Kandel [31] discuss the notion of weighted union of fuzzy sets but do not apply it to compute similarity values for multifeature objects or object descriptions with mixed nominal and numeric data. The CU and correlation measures are best suited for binary-valued features and result in loss of information and arbitrary weighting of features when applied to multivalued nominal features. This is because individual features are weighted in proportion to the number of values associated with them. There is even greater loss of information when they are applied to numeric features because the measures do not consider the magnitude of the difference between two feature values.

The Goodall similarity measure defines a unified framework for handling nominal and numeric features. It was first proposed for biological and genetic taxonomy problems, where unusual characteristics shared by biological entities is often attributed to closely related genetic information resulting in these entities being classified into the same species [15]. In this paper, this similarity measure has been extended to clustering problems in more general domains. A pair of objects (i, j) is considered more similar than a second pair of objects (l, m), if and only if the objects i and j exhibit a greater match in feature values that are less common in the population. In other words, similarity among objects is decided by the uncommonality of their *feature value* matches. Similarity computed using this heuristic helps to define more cohesive, tight clusters where objects grouped into the same cluster are likely to share special and characteristic feature values. One should note that common feature values also play a role in the similarity computation and in the clustering process. The similarity computation is realized by weighing feature value matches between a pair of objects by the frequency of occurrence of the feature value in the data set.

Consider two pairs of objects (i, j) and (l, m). For a feature k, value $(V_i)_k = (V_j)_k$ and value $(V_l)_k = (V_m)_k$, but $(V_i)_k \neq (V_l)_k$. $(V_i)_k$ occurs equally or more frequently in the population than $(V_l)_k$. This is expressed as

$$((p_i)_k = (p_j)_k) \geq ((p_l)_k = (p_m)_k),$$

where $(p_i)_k$, $(p_j)_k$, $(p_l)_k$, and $(p_m)_k$ define the probabilities of occurrence of the respective feature values in the population. Feature k makes a greater contribution to the similarity value for object pair (l, m) than it does for object pair (i, j). This is summarized as follows:

$$\left.\begin{array}{l}((V_i)_k = (V_j)_k) \wedge ((V_l)_k = (V_m)_k) \\ ((p_i)_k = (p_j)_k) \geq ((p_l)_k = (p_m)_k)\end{array}\right\} \Longrightarrow (S_{ij})_k \leq (S_{lm})_k. \quad (1)$$

For numeric feature values, the similarity measure takes both the *magnitude* of the feature value difference and the *uniqueness* of the feature value pair into account. The smaller the magnitude of the difference between the values $((V_i)_k, (V_j)_k)$, the less likely it is that a pair of values picked at random will fall in the segment defined by the endpoints $(V_i)_k$ and $(V_j)_k$, therefore, the more similar this pair of objects. Consider two pairs of objects (i, j) and (l, m). For numeric feature, k

$$(|(V_i)_k - (V_j)_k| > |(V_l)_k - (V_m)_k|) \Longrightarrow (S_{ij})_k < (S_{lm})_k. \quad (2)$$

When $(|(V_i)_k - (V_j)_k| = |(V_l)_k - (V_m)_k|)$ the similarity value is influenced by the *uniqueness* of the segment defined by the values. The uniqueness of a segment is computed by summing up the frequency of occurrence of all other values in the population that are between the pair of values that define the segment. For the object pair (i, j) this value is computed as $\sum_{t=(V_i)_k}^{(V_j)_k} p_t$ and for object pair (l, m) it is $\sum_{t=(V_l)_k}^{(V_m)_k} p_t$. For two segments of equal length but different endpoints, the segment which includes the lower cumulative frequency of occurrence of other values within its endpoints is defined to be more unique and, therefore, this segment makes a greater contribution to the similarity measure.

$$\left.\begin{array}{l}|(V_i)_k - (V_j)_k| = |(V_l)_k - (V_m)_k| \\ \sum_{t=(V_i)_k}^{(V_j)_k} p_t \geq \sum_{t=(V_l)_k}^{(V_m)_k} p_t\end{array}\right\} \Longrightarrow (S_{ij})_k \leq (S_{lm})_k. \quad (3)$$

3.1.1 Computing Similarity for Nominal Features

The similarity index for a nominal valued attribute k is computed as follows:

$$\begin{array}{lll}(V_i)_k \neq (V_j)_k & \Longrightarrow & (S_{ij})_k = 0, \\ (V_i)_k = (V_j)_k & \Longrightarrow & 0 < (S_{ij})_k < 1.\end{array}$$

As discussed above, when $(V_i)_k = (V_j)_k$, the similarity value is a function of the uncommonality of the feature value within the population. For two objects i and j, with an identical value for feature k (say $(V_i)_k$), we first define its *More Similar Feature Value Set*, $MSFVS((V_i)_k)$. This is the set of all pairs of values for feature k that are equally or more similar to the pair $((V_i)_k, (V_i)_k)$. The value pairs are selected according to relation defined in (1). Note that a value pair is more similar if it has lower frequency of occurrence. The probability of picking a pair $((V_l)_k, (V_l)_k) \in MSFVS((V_i)_k)$ at random is

$$(p_l)^2{}_k = \frac{(f_l)_k \cdot ((f_l)_k - 1)}{n \cdot (n-1)},$$

where $(f_l)_k$ is the frequency of occurrence of value $(V_l)_k$ in the population and n is the total number of objects in the population. Summation of the probabilities of all such pairs gives the dissimilarity score of the pair, $(D_{ii})_k$. Thus, the similarity of the pair $((V_i)_k, (V_i)_k)$ is computed as:

$$(S_{ii})_k = 1 - (D_{ii})_k = 1 - \sum_{l \in MSFVS((V_i)_k)} (p_l)_k^2. \quad (4)$$

A simple example illustrates this computation. Table 3 shows a population of 10 objects, each described by a nominal and a numeric feature. The similarity value for non identical nominal features is 0, i.e., $S_{(a,b)} = S_{(a,c)} = S_{(b,c)} = 0$. For identical value pairs, i.e., $S_{(a,a)}$, $S_{(b,b)}$, and $S_{(c,c)}$, we first compute the MSFVS for each pair. In the given population, $f(a) = 3$, $f(b) = 3$, and $f(c) = 4$, i.e., values a and b are less

TABLE 3
Sample Data Set of 10 Objects, Each Described by a Nominal and a Numeric Feature

Object Number	Feature 1 (nominal)	Feature 2 (numeric)
1	a	6
2	b	5.5
3	b	7.5
4	a	6
5	c	10.5
6	c	9
7	c	7.5
8	c	9
9	a	7.5
10	b	7.5

frequent than value c. The $MSFVS$ for feature value pair (c, c),

$$MSFVS(c,c) = \{(a,a),(b,b),(c,c)\}.$$

Given the MSFVS, the similarity value for the pair (c, c) is calculated by (4).

$$\begin{aligned} D_{(c,c)} &= {p_a}^2+{p_b}^2+{p_c}^2 = \tfrac{3(3-1)}{10(10-1)}+\tfrac{3(3-1)}{10(10-1)}+\tfrac{4(4-1)}{10(10-1)} = 0.267, \\ S_{(c,c)} &= 1-D_{(c,c)} = 0.733. \end{aligned}$$

Feature 1 contributes a value of 0.733 to the similarity between objects 5 and 6.

$$\begin{aligned} D_{(a,a)} &= {p_a}^2+{p_b}^2 = \tfrac{3(3-1)}{10(10-1)}+\tfrac{3(3-1)}{10(10-1)} = 0.133, \\ S_{(a,a)} &= 1-D_{(a,a)} = 0.867. \end{aligned}$$

Feature 1 contributes a value of 0.867 to the similarity between objects 1 and 4. This is larger than the contribution of feature 1 to the similarity between objects 5 and 6, (0.733).

The computational complexity for the similarity calculation for a feature with m_d observed values is: $O(n + m_d log m_d)$. If $m_d \ll n$, the total number of objects, this reduces to $O(n)$.

3.1.2 Computing Similarity for Numeric Attributes

This computation is similar to the similarity calculation for nominal features. The calculation of the similarity value for a pair of objects whose kth feature value is $((V_i)_k, (V_j)_k)$ starts with the determination of the *More Similar Feature Segment Set*, $MSFSS((V_i)_k, (V_j)_k)$. This includes all pairs of values $((V_l)_k, (V_m)_k)$ satisfying (2) or (3). The probability of picking two objects from the population having values $(V_l)_k$ and $(V_m)_k$ for feature k, where

$$((V_l)_k, (V_m)_k) \in \mathrm{MSFSS}((V_i)_k, (V_j)_k),$$

is

$$\begin{cases} 2(p_l)_k(p_m)_k = \frac{2(f_l)_k(f_m)_k}{n(n-1)}, & (p_l)_k \neq (p_m)_k, \\ (p_l)_k(p_m)_k = \frac{(f_l)_k((f_l)_k-1)}{n(n-1)}, & (p_l)_k = (p_m)_k, \end{cases}$$

where f_l and f_m are the frequency of occurrence of values $(V_l)_k$ and $(V_m)_k$, respectively and n is the total number of objects in the population. Summing up the probabilities of all value pairs in MSFSS$((V_i)_k, (V_j)_k)$ produces the dissimilarity

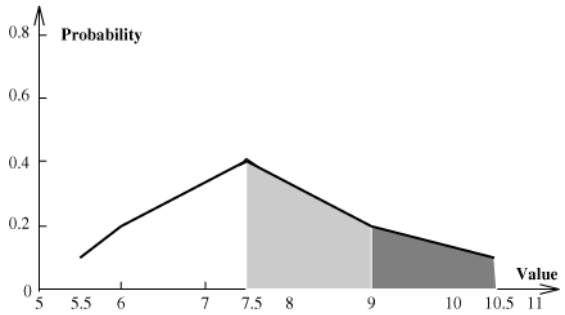

Fig. 3. Numeric feature distribution for an example population.

contribution of feature k, $(D_{ij})_k$. From this the similarity of the pair $((V_i)_k, (V_i)_k)$ is computed as:

$$(S_{ij})_k = 1 - (D_{ij})_k = 1 - \sum_{l,m \in MSFSS((V_i)_k,(V_j)_k)} 2(p_l)_k(p_m)_k.$$

The simple example from the previous section (Table 3) helps demonstrate the calculation of the similarity measure for numeric features. The probability distribution of the numeric feature of the sample population is plotted in Fig. 3. Consider two pairs of feature values (7.5, 9) and (9, 10.5). Both are 1.5 units apart, but the population included in the two intervals are different. The population included in the interval (7.5, 9), shaded light, is larger than the population included in the interval (9, 10.5), which is shaded dark. Therefore, the similarity contribution from feature 2 for the pair (9, 10.5) should be greater than that for the pair (7.5, 9). To calculate the similarity value for each pair, we first identify the MSFSS for each pair:

$$\begin{aligned} MSFSS(7.5,9) &= \{(5.5,5.5),(6,6),(7.5,7.5),(9,9),(10.5,10.5),(5.5,6), \\ &\quad (6,7.5),(7.5,9),(9,10.5)\} \\ MSFSS(9,10.5) &= \{(5.5,5.5),(6,6),(7.5,7.5),(9,9),(10.5,10.5),(5.5,6),(9,10.5)\}. \end{aligned}$$

The computed dissimilarity score for each pair of values is:

$$\begin{aligned} D_{(7.5,9)} &= p_{5.5}p_{5.5}+p_6p_6+p_{7.5}p_{7.5}+p_9p_9+p_{10.5}p_{10.5}+2p_{5.5}p_6+2p_6p_{7.5}+ \\ &\quad 2p_{7.5}p_9+2p_9p_{10.5}=0.51, \\ D_{(9,10.5)} &= p_{5.5}p_{5.5}+p_6p_6+p_{7.5}p_{7.5}+p_9p_9+p_{10.5}p_{10.5}+2p_{5.5}p_6+2p_9p_{10.5}=0.27. \end{aligned}$$

Finally, the similarity measure for the pair (9, 10.5) is $S_{(9,10.5)} = 1 - D_{(9,10.5)} = 0.73$, which is greater than that of the pair (7.5, 9), $S_{(7.5,9)} = 1 - D_{(7.5,9)} = 0.49$.

Note that the similarity contribution from numeric features is a function of both distance and density. As long as a change of scale or a transformation for a numeric feature preserves the order of relative distances between its feature values, the similarity measure for the feature value pairs is invariant. Also, the monotonic nature of the (dis)similarity measures is preserved. For example, $D(7.5, 9) < D(9, 11)$ because $|7.5 - 9| < |9 - 11|$ even though the range from 9 to 11 is very sparsely populated in the distribution in Fig. 3. On the other hand, if two distances are equal, as in the example shown above, the pair that includes a sparser population in its defined interval is considered more similar. Therefore, the distance (and similarity) measure for numeric values preserve metric properties.

The direct implementation of the above calculation takes $O({m_c}^4)$ time complexity for a numeric feature with m_c unique values. The computation can be sped up by first finding the difference of all pairs of values and sorting the differences in ascending order and, then, using an accumulator to collect the similarity score by counting pairs of values in the sorted difference list. With this implementation, the time complexity for the computation is cut down to $O(m_c^2 log m_c^2)$. In the worst case, $m_c = n$, where n is the total number of objects and the above complexity is bounded by $O(n^2 log n)$.

3.1.3 Aggregating Similarity Contributions from Multiple Features

We now extend the similarity score computations for individual features to an aggregate similarity score for a pair of objects described by multiple, mixed type features. Assuming that the individual results are expressed as the square of a standard normal deviate, Fisher's χ^2 transformation [11]: $\chi^2 = -2ln(P_i)$ works well with data from continuous populations and for discrete populations where a feature has a large number of distinct observations associated with it.

However, for nominal features with a small number of possible observations, e.g., a binomial distribution with low index, empirical results show that the transformation causes the mean and standard deviation of the transformed population to deviate significantly from the theoretical mean and standard deviation, which diminishes the power of the tests [20]. To remedy this problem, Lancaster suggested a modified transformation called the *mean value* χ^2 transformation [20], χ_m^2. Instead of using the probability of the event actually observed, P, in the transformation, an intermediate value between the probability of the event actually observed, P, and the next smaller probability in the discrete set that includes it, P', is used. Through the probability integral transformation, the *mean value* χ^2 transformation becomes:

$$\chi_m^2 = \int_{P'}^{P} (-2lnP) d(\frac{P}{P - P'}) = 2 - \frac{2(PlnP - P'lnP')}{P - P'}.$$

Thus, we combine the similarity test scores from numeric features using Fisher's χ^2 transformation:

$$(\chi_c)_{ij}^2 = -2 \sum_{k=1}^{t_c} ln((D_{ij})_k), \tag{5}$$

where t_c is the number of numeric features in the data. χ_c also follows χ^2 distribution with t_c degrees of freedom. The similarity test scores for the nominal features are combined using Lancaster's *mean value* χ^2 transformation:

$$(\chi_d)_{ij}^2 = 2 \sum_{k=1}^{t_d} (1 - \frac{(D_{ij})_k ln(D_{ij})_k - (D_{ij})'_k ln(D_{ij})'_k}{(D_{ij})_k - (D_{ij})'_k}, \tag{6}$$

where t_d is the number of nominal attributes in the data, $(D_{ij})_k$ is the dissimilarity score for nominal attribute value pair $((V_i)_k, (V_j)_k)$, and $(D_{ij})'_k$ is the next smaller dissimilarity score in the nominal set. χ_d^2 is χ^2 distributed with t_d degrees of freedom.

TABLE 4
The Step by Step Calculation of Similarity Measure Between Pairs of Objects

Calculation Steps	Object Pairs	
	(Obj6, Obj9)	(Obj5, Obj8)
χ_c^2	1.35	2.62
χ_d^2	1.037	3.19
χ^2	2.387	5.81
D	0.66	0.21
S	0.34	0.79

The addition of two χ^2 distributions is still χ^2 with degree of freedom equal to the sum of the two degrees of freedom, i.e., the probability distribution for combining the two types of features is χ^2 distributed with $(t_c + t_d)$ degrees of freedom. The significance value of this χ^2 distribution can be looked up in standard tables or approximated from the expression:

$$D_{ij} = e^{-\frac{\chi_{ij}^2}{2}} \sum_{k=0}^{(t_d+tc-1)} \frac{(\frac{1}{2}\chi_{ij}^2)^k}{k!},$$

where $\chi_{ij}^2 = (\chi_c)_{ij}^2 + (\chi_d)_{ij}^2$. The overall similarity score representing the set of $(t_c + t_d)$ independent similarity measures is $S_{ij} = 1 - D_{ij}$.

We revisit the sample population of objects given in Table 3 to illustrate the computation for the similarity measure for pairs of objects. For the nominal feature, the dissimilarity measure for object pairs (obj6, obj9) and (obj5, obj8) were computed to be $D_{(a,c)} = 1$ and $D_{(c,c)} = 0.267$, respectively. For the numeric feature, the similarity measure for the two pairs were computed as $D_{(7.5,9)} = 0.51$ and $D_{(9,10.5)} = 0.27$, respectively. Since there is only one nominal feature and one numeric feature in this example, the summation of t_c and t_d terms in (5) and (6) for the aggregate similarity computation are reduced to single term operations. Results of the step by step calculations for χ_d, χ_c, the dissimilarity measure, and the similarity score for the object pairs (obj6, obj9) and (obj5, obj8) appear in Table 4. The overall similarity between object 5 and object 8 is higher than the similarity score for object 6 and object 9. This is consistent with the fact that for each individual feature, values between object 5 and object 8 are more similar to each other than those between object 6 and object 9.

3.2 The Control Structure

SBAC's agglomerative, hierarchical clustering algorithm based on the Unweighted Pair Group Method with Arithmetic Average (UPGMA) [17], starts with a pairwise dissimilarity matrix D of the set of objects to be clustered. Dissimilarity between a pair of objects is the complement of their similarity score, $D_{ij} = 1 - S_{ij}$. At any step, the clusters that have the minimum pairwise dissimilarity value are merged into a single cluster. Dissimilarity between the new cluster and the other clusters is defined as the average dissimilarity between an old cluster and the component clusters of the new cluster. The end result is a dendrogram (or classification tree) whose leaf nodes are individual

TABLE 5
The Control Structure of SBAC System

1. Begin with n clusters, i.e., each object i, $1 \leq i \leq n$ defines a cluster. Set sequence number $m = 0$ for the current partition, and corresponding level $L(m) = L(0) = 0$.
2. Find the least dissimilar pair of clusters in the current partition, say the pair $((r), (s))$, where $D[(r), (s)] = \text{Min}_{i,j}\{D[(i), (j)]\}$ for all pairs $((i), (j))$ in the current dissimilarity matrix.
3. Increment the sequence number: $m = m + 1$. Merge clusters $(r), (s)$ into a single cluster (p) to form the next partition m. Set the corresponding level of this partition to $L(m) = D[(r), (s)]$.
4. Update the dissimilarity matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s), and adding a row and column corresponding to the newly formed cluster. The dissimilarity between the new cluster, (p) and an existing cluster (k) is defined as follows:

$$D[(p), (k)] = \frac{n_r}{n_r + n_s} D[(r), (k)] + \frac{n_s}{n_r + n_s} D[(s), (k)],$$

   where n_r and n_s are the number of objects in clusters (r) and (s), respectively.
5. Repeat Steps 2-4 until all objects are in a single cluster.
6. Extract the final partition from the dendrogram by selecting the "significant" cluster down each path of the dendrogram in a manner described in Section 3.3.

objects and whose root defines a cluster or group that includes all objects. Table 5 describes the steps of the algorithm.

In the classification tree, each node has an associated dissimilarity score, which indicates the dissimilarity level at which its child nodes merge together. Naturally, the child nodes form a more cohesive group than their parent, so from the root down the dissimilarity score along each path decreases monotonically down to the leaf nodes, which have dissimilarity values of 0. Since our goal is to form clusters that are cohesive but not too fragmented, we have to deal with the classic trade-off between cohesiveness and fragmentation. To achieve the proper balance between these two, we traverse the tree top-down in a depth first fashion and cut off traversal at points where the difference in dissimilarity values between parent and child is less than a predefined percentage threshold, t. The set of nodes on the frontier of the traversal along the different paths define the clusters of interest. In our experiment, we set the threshold value $t = 0.3 * D(root)$, where $D(root)$ is the dissimilarity score for the root node.

To illustrate the scheme, an example of the depth-first traversal is shown for the classification tree in Fig. 4. Given $D(root) = 0.876$, we compute the threshold value as $t = 0.263$. Following the path from the root to node 2 and then to node 4, the dissimilarity scores drop by 0.083 and 0.081, respectively, which are lower than the threshold value. A significant drop in dissimilarity score, 0.585, is observed from node 4 to node 8, so the search is terminated below node 8. Paths from node 4 to its other children, node 9 and 10, show dissimilarity score drops below the threshold, so the search is terminated below nodes 9 and 10. The search then continues from other children of node 2, in this case node 5, where a significant dissimilarity score drop of 0.289 is observed. This makes node 5 part of the extraction frontier. From the root to node 3, the drop of dissimilarity score is again higher than 0.263. So node 3 becomes part of the extraction frontier and the search terminates. The end result is that nodes 8, 9, 10, 5, and 3 define the partition structure.

The computational complexity of the SBAC algorithm is derived in two parts: 1) the complexity for generating the dissimilarity matrix for all pairs of objects and 2) the complexity for constructing the dendrogram or concept tree. It is well established in literature [17] that the upper bound

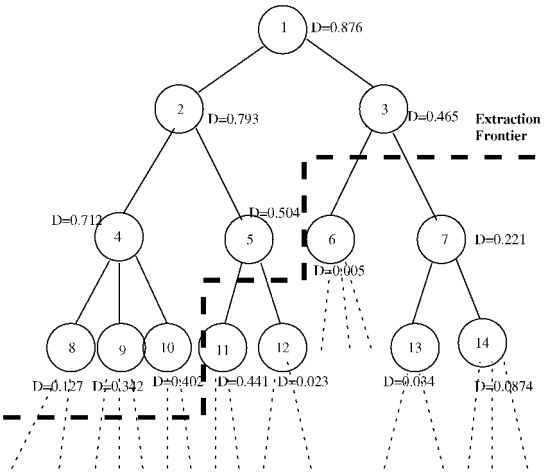

Fig. 4. The partition extraction algorithm in SBAC.

complexity for part 2) is $O(n^2)$, where n is the number of objects in the data set. Also, constructing the similarity matrix for pairs of objects, given the similarity index for individual attributes, is $O(n^2)$. Using the time complexities to compute the individual similarity indices reported earlier, the overall time complexity for generating partitions is $O(n^2 + m_c^2 log m_c^2)$, where m_c is the highest possible number of distinct values observed for a numeric feature. For large data sets, this reduces to $O(n^2)$, where $n \gg m_c$, and for small data sets, it is $O(m_c^2 log m_c)$, where $n \ll m_c$.

4 Experimental Studies

Our goal for performing empirical studies with SBAC were twofold: 1) to gain a better understanding of the characteristics of the Goodall similarity measure and (2) to compare the performance of SBAC with existing unsupervised discovery systems, such as AUTOCLASS, COBWEB/3, and ECOBWEB. To achieve this, we used two kinds of data 1) artificially generated data, with mixed nominal and numeric features and 2) three real data sets, the hand written 8OX character data, the mushroom data, and the heart disease data.

4.1 Artificial Data

We first describe the method used for generating the artificial data and then present a comparative analysis of the results.

4.1.1 Data Description

The artificial data set had 180 data points, equally distributed into three classes: G_1, G_2, and G_3. Each data point was described using four features: two nominal and two numeric. The nominal feature values were predefined and assigned to each class in equal proportion. Nominal feature 1 has a unique symbolic value for each class and nominal feature 2 had two distinct symbolic values assigned to each class. The numeric feature values were generated by sampling normal distributions with different means and standard deviations for each class, where the means for the three classes are three standard-deviations apart from each other. For the first class, the two numeric features are distributed as $\mathrm{N}(\mu = 4, \sigma = 1)$ and $\mathrm{N}(\mu = 20, \sigma = 2)$; for the second class, the distributions were $\mathrm{N}(\mu = 10, \sigma = 1)$ and $\mathrm{N}(\mu = 32, \sigma = 2)$; and for the third class, $\mathrm{N}(\mu = 16, \sigma = 1)$ and $\mathrm{N}(\mu = 44, \sigma = 2)$.

For the base data set, $d0\%c0\%$, none of the feature values were corrupted. The remaining data sets were created by randomly corrupting the base data set using 1) non-Gaussian noise on both the nominal features and the numeric features and 2) Gaussian noise, on the numeric features. To avoid the influence of order effects for the COBWEB/3 and ECOBWEB algorithms, the order of objects for each run were randomized using the same seed for random number generation. This order was used for all four algorithms.

In our experiments, we created data sets that had corrupted numeric features, or corrupted nominal features, but not both. Our primary goal here was to study how each one of the systems was biased toward numeric and nominal features.

Data with Non-Gaussian Noise. Six data sets were created as variations of the $d0\%c0\%$ data set by successively mixing up 20 percent, 40 percent, and 60 percent of the feature values for one class with the other two classes. For example, the data set $d20\%c0\%$ represents the data set where 20 percent of the nominal feature values picked randomly in each class were changed to values from other classes, but none of the numeric feature values were corrupted. On the other hand, $d0\%c40\%$ implies that no nominal feature values were corrupted, but 40 percent of the numeric feature values in each class were corrupted by picking their values from the normal distributions defined for the other two classes. The corrupted data sets used in this study can be described as: $d20\%c0\%$, $d40\%c0\%$, $d60\%c0\%$, $d0\%c20\%$, $d0\%c40\%$, and $d0\%c60\%$.

Data with Gaussian Noise. Four data sets were created by adding successively higher degree of Gaussian noise to the numeric features. The noise was characterized by $N(0, \frac{x}{2}\sigma)$, where x indicated the degree of noise introduced. For example, the data set $d0c(\frac{\sigma}{2})$ introduced into the base data set Gaussian noise with mean = 0 and standard deviation = half the value of the standard deviations of the original feature values. The standard deviations of the two numeric features in the base data set were 1.0 and 2.0, respectively. So, $d0c\frac{\sigma}{2}$ represents a data set with well-separated nominal features and two numeric features corrupted with noise distributions $N(0, \frac{1}{2})$ and $N(0, 1)$, respectively. The four corrupted data sets created for this experiment were $d0c(\frac{\sigma}{2})$, $d0c(\sigma)$, $d0c(\frac{3\sigma}{2})$, and $d0c(2\sigma)$.

4.1.2 Results

The partitional structures generated by the four clustering systems were evaluated using a *misclassification count* measure under the assumption that the structure derived from the $d0\%c0\%$ data set was the true structure. The *misclassification count* computes the number of object misclassifications in the C_i class assuming G_i is the true class. C_i and G_i correspond when a majority of the C_i objects have the G_i label and no other G_k ($k \neq i$) group has a larger number of objects in C_i. If more than three groups are formed, the additional smaller C groups are labeled as fragmented groups. The misclassification count is computed as follows:

1. if an object falls into a fragmented C group, where its type (G label) is a majority, it is assigned a value of 1,
2. if the object is a minority in any group, it is assigned a misclassification value of 2,
3. otherwise the misclassification value for an object is 0.

The misclassification values summed over all objects is the misclassification count for the partition. This is used to measure the goodness of the derived partition. The smaller this value, the closer this partition structure is to the assumed true structure.

Results for Data with Non-Gaussian Noise The partitional structures generated by the four methods are listed in Table 6. The *misclassification counts* for the partitional structures are plotted in Fig. 5. The solid lines represent the effects of increasing noise added to the numeric attributes and the dotted lines represent the effects of adding noise to the nominal attributes. It was observed that

TABLE 6
Partitional Structures Generated by Clustering Systems on Artificial Data Sets with Non-Gaussian Noise

Data Sets		SBAC	COBWEB3	AUTOCLASS	ECOBWEB
base data	d0%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_3:15) $C_2(G_2$:60) $C_3(G_3$:29) $C_4(G_3$:16)
noisy numeric features	d0%c20%	$C_1(G_1$:59) $C_2(G_2$:60, G_3:1) $C_3(G_3$:59, G_1:1)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:37) $C_2(G_2$:38) $C_3(G_3$:38) $C_4(G_3$:6) $C_5(G_1$:11, G_2:6)) $C_6(G_1$:1, G_2:6, G_3:5) $C_7(G_1$:5, G_3:6) $C_8(G_2$:5, G_3:6) $C_9(G_1$:6, G_2:5)	$C_1(G_1$:48, G_2 :2, G_3:5) $C_2(G_1$:12, G_2:11, G_3:55) $C_3(G_2$:47)
	d0%c40%	$C_1(G_1$:60, G_3:1) $C_2(G_2$:60) $C_3(G_3$:59)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:36, G_3:1) $C_2(G_2$:11, G_3:16) $C_3(G_2$:25, G_3:1) $C_4(G_1$:22) $C_5(G_3$:20) $C_6(G_1$:1, G_3:9) $C_7(G_2$:2, G_3:7) $C_8(G_2$:7, G_3:2) $C_9(G_2$:5, G_3:4) $C_{10}(G_1$:1, G_2:5) $C_{11}(G_2$:5)	$C_1(G_1$:36, G_2:12, G_3:9) $C_2(G_1$:11, G_2:36, G_3:12) $C_3(G_1$:13, G_2:12, G_3:39)
	d0%c60%	$C_1(G_1$:59, G_2:1, G_3:2) $C_3(G_2$:58, G_1:1) $C_3(G_3$:58, G_2:1)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:36) $C_2(G_1$:14) $C_3(G_2$:17, G_3:14) $C_4(G_2$:12, G_3:17) $C_5(G_2$:13, G_3:11) $C_6(G_2$:12, G_3:11) $C_7(G_1$:10, G_2:6, G_3:7)	$C_1(G_1$:60, G_2:2, G_3:57) $C_2(G_2$:58, G_3:3)
noisy nominal features	d20%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_2:7, G_3:1) $C_2(G_2$:53) $C_3(G_3$:59)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_3:12) $C_2(G_2$:60) $C_3(G_3$:27) $C_4(G_3$:21)
	d40%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:12, G_2:14, G_3:36) $C_2(G_1$:11, G_2:35, G_3:12) $C_3(G_1$:37, G_2:11, G_3:12)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60, G_2:18, G_3:10) $C_2(G_2$:42) $C_3(G_3$:28) $C_4(G_3$:22)
	d60%c0%	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:17, G_2:15, G_3:26) $C_2(G_1$:16, G_2:25, G_3:18) $C_3(G_1$:27, G_2:20, G_3:16)	$C_1(G_1$:60) $C_2(G_2$:60) $C_3(G_3$:60)	$C_1(G_1$:60) $C_5(G_3$:18) $C_2(G_2$:21) $C_6(G_3$:21) $C_3(G_2$:23) $C_7(G_3$:21) $C_4(G_2$:16)

as the noise levels in the numeric features increased, the AUTOCLASS partitions deteriorated quite rapidly, but there was little deterioration when the nominal features were corrupted. COBWEB/3 showed the opposite effect. It was more sensitive to deterioration in the nominal features. For ECOBWEB, the deterioration in partition structure was almost equal for degraded numeric and nominal features. Overall, ECOBWEB was more sensitive to noisy numeric features whereas COBWEB/3 was more sensitive to noisy nominal features. Also, unlike the other algorithms, ECOBWEB did not perform very well on the base data set. Of the four algorithms, SBAC was the most robust and showed very little deterioration in performance as the degradation levels were increased. Also, unlike the other three algorithms, SBAC showed no noticeable bias toward either the nominal or numeric features.

We analyze the SBAC algorithm in more detail. For similarity score computations among objects belonging to the same class, when a particular feature value for an object was corrupted (i.e., it took on a value from another class), this feature's contribution to the similarity score decreased significantly, but the remaining feature contributions remained high. As a result, the effect of the corrupted features on the similarity score was greatly mitigated. This is illustrated in Table 7. The average contribution from the nominal features and the numeric features to the overall similarity measure are compared for pairs of objects from the same class and pairs of objects from different classes. For the data set with well-separated numeric features and corrupted nominal features, the difference in average similarity for the within class and between class object pairs is quite small. This indicates that the predictability of noisy nominal features is low. But there is a big difference in the within class and between class average similarity values for the numeric features. Moreover, the contribution to similarity from the numeric features is about two times the contribution from the nominal features within a class and about $\frac{1}{2}$ between classes. This big difference implies the robustness of the similarity measure to added noise and this is primarily due to the difference in contributions between the uncorrupted versus corrupted features. The overall results for the data set with well-separated nominal features and noisy numeric features were just the same. Here noise reduced the difference between the within and between class similarity values for the numeric features, while maintaining a large difference for the nominal features. However, the differences in contribution were not as

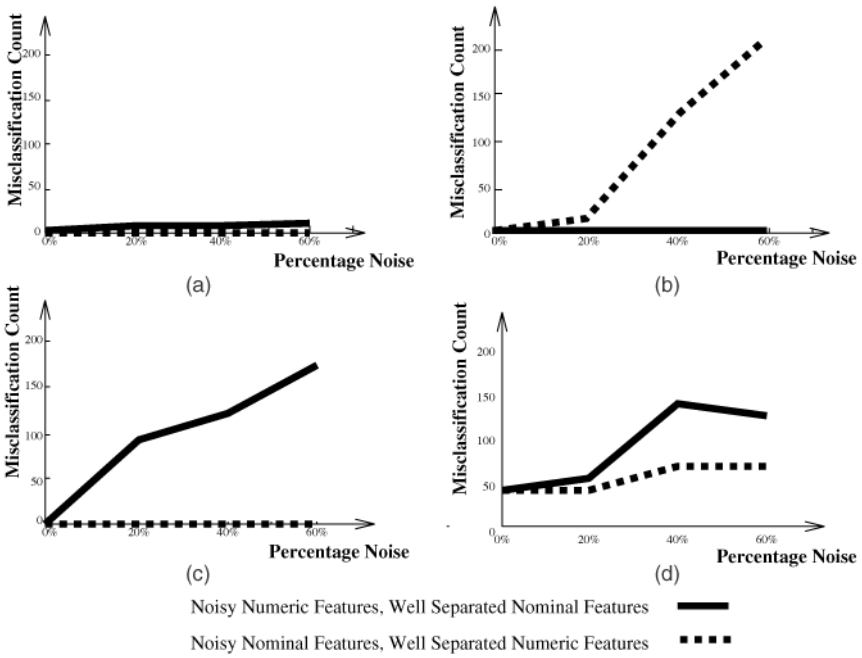

Fig. 5. Comparison of partitional structures generated by clustering systems on artificial data sets with non-Gaussian noise. (a) SBAC, (b) COBWEB/3, (c) AUTOCLASS, and (d) ECOBWEB.

distinct, which explains the slightly larger misclassification percentages observed in Fig. 5a.

The AUTOCLASS results may be explained by comparing the data generation scheme to the assumptions that the algorithm makes about the data. The numeric data was generated from Gaussian distributions. This provided a closer fit to the AUTOCLASS assumption for the distribution of the prior probability. However, AUTOCLASS assumes a Bernoulli distribution with uniform Dirichlet conjugate prior for the nominal attributes and this did not match the way we generated the nominal data for this experiment. Nevertheless, it is still a little surprising that AUTOCLASS seemed to disregard the distinctive nominal attributes completely in $d0\%c*\%$ cases. Another possible reason for the deterioration of AUTOCLASS's performance to the gradual increase of noise in nonnumeric features is that the model assumed by AUTOCLASS is Gaussian. Separate experiments conducted by adding Gaussian noise to the numeric features demonstrated better performance. This is discussed later. Also, when comparing the partition structures generated by the other algorithms to the AUTOCLASS partitions, AUTOCLASS tended to construct more fragmented partition structures. This may be attributed to the fact that with the addition of noise, AUTOCLASS found better fit when using a larger number of component distributions. Cheesman and Stutz discuss this issue (*Occam's Razor* [5]) and implement schemes to help prevent AUTOCLASS from over fitting. However, the schemes do not seem to be activated early enough to prevent fragmentation in the partition structures created by AUTOCLASS.

The partition structure generated by ECOBWEB was not dominated by either well-separated nominal attributes or well-separated numeric attributes. Despite the fact that ECOBWEB and COBWEB/3 share the same criterion function for nominal attributes, the partition structure formed by COBWEB/3 is better than the one generated by ECOBWEB for the $d0\%c\star\%$ data sets. The difference in the structures can be explained by the differences in the implementations of the CU function for numeric features. For COBWEB/3, the $CU_{numeric}$ score depends solely on the standard deviation of the feature distribution for a class. When noise is introduced, the standard-deviations of the "model classes" increase, causing $CU_{numeric}$ to decrease. The higher the level of noise, the less the contribution of $CU_{numeric}$ to the overall CU score. Thus when numeric features were corrupted, the clustering process is mainly based on the $CU_{nominal}$ score which remains high for all $d0\%c\star\%$ cases. For ECOBWEB, its $CU_{numeric}$ score is

TABLE 7
Similarity Contribution from Nominal Features and Numeric Features Averaged Over All Pairs of Objects from the Same Class and all Pairs of Objects from Different Classes

Similarity Contribution	**Object Pairs for Data d40%c0%**		**Object Pairs for Data d0%c40%**	
	Same Class	Between Classes	Same Class	Between Classes
Average χ_c^2	8.44	1.82	4.66	3.68
Average χ_d^2	4.54	3.72	7.65	2.20
Average D	0.17	0.69	0.20	0.67
Average S	0.83	0.31	0.80	0.33

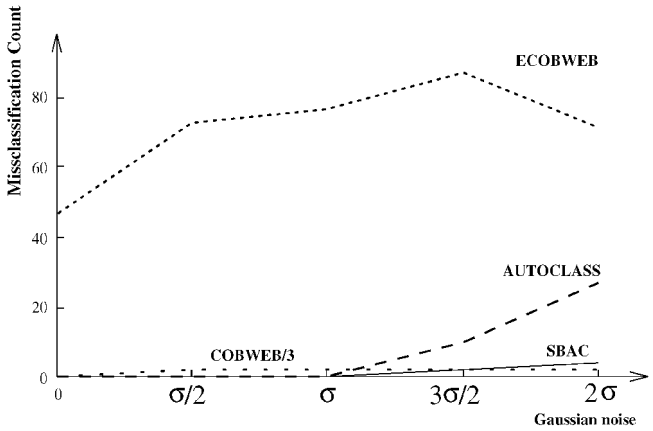

Fig. 6. Comparison of partitional structures generated by clustering systems on artificial data sets with Gaussian noise, σ is the standard deviation of the numeric feature values of the base data set.

calculated as the probability of a small interval around the mean value of the feature distribution in a class. When noise is introduced, the mean values of the features are shifted away from those of the "model classes." Depending on the size of the interval defined, the probability calculated around the mean and, thus, the $CU_{numeric}$ contribution, can be large enough to make a difference in the overall CU score. This perturbed the class structures that would have been generated using the $CU_{nominal}$ scores. It is interesting to observe that the performance of ECOBWEB improved when high levels of noise were introduced to the numeric features (60 percent and larger). We conjecture that an excessive amount of noise makes the distribution of numeric features close to uniform over the entire range. No matter what the estimated mean value for each of the three classes is, the computed probability of the small interval around the mean is going to be about the same. Therefore, the $CU_{numeric}$ scores have little effect on the overall CU measure and the $CU_{nominal}$ scores govern the cluster formation process.

On the other hand, ECOBWEB performs noticeably better than COBWEB/3 with noisy nominal but clean numeric data (the $d\star\%c0\%$ data). This is because nominal features make a greater contribution than the numeric features to the CU measure for COBWEB/3. This is especially true for the $CU_{numeric}$ scores when the standard deviations of feature distributions in a "true class' exceeds 1. ECOBWEB's $CU_{numeric}$ measure is based on the approximation of the true probability of feature distributions in a class. When the approximation is close enough to the real probability, it captures distinct differences in numeric feature values and it mitigates the effect of the $CU_{nominal}$ scores derived from the corrupted nominal features.

Results from Data with Gaussian Noise The *misclassification count* was again used as the performance measure to evaluate the four clustering systems. The results are plotted in Fig. 6. The misclassification count for the SBAC remains low till noise levels become very large(2σ). The performance is in conformance with the results for non-Gaussian noise (Fig. 5a). A significant performance improvement is observed for AUTOCLASS. ECOBWEB's performance with data having Gaussian and non-Gaussian noise follows a similar pattern. In both cases, performance degrades as noise levels are increased and they improve slightly when noise levels become high. Like SBAC and AUTOCLASS, there are only minor changes observed for the clustering results generated by COBWEB/3, when Gaussian noise is present.

In order to explain these results, we revisited the criterion functions and the assumptions employed by each system. For SBAC, noise mitigation is achieved the same way as with non-Gaussian noise. Since the nominal features are well-separated for each class, a big difference in the χ_d^2 score is expected for pairs of objects from the same class versus pairs of objects from different classes. For numeric features with low levels of added Gaussian noise ($\sigma_{noise} = \frac{1}{2}\sigma_{feature}$ and $\sigma_{noise} = \sigma_{feature}$), the relative distance of feature values is not altered by much, so feature values within a class remain closer than the feature values between classes. As a result, the χ_c^2 score for objects within a class are higher than for objects across classes and the partition structure is preserved. When noise levels are increased beyond $\sigma_{feature}$, feature values overlap at the boundaries between classes, thus the differences between within class and between class similarities (i.e., χ_c^2) decreases for the numeric features. Therefore, objects at the boundaries between classes may be misclassified.

AUTOCLASS showed much better performance for data with Gaussian noise. Low noise levels did not affect the separability of feature values, therefore, AUTOCLASS correctly estimated the parameters of the normal distribution for each cluster and this was sufficient to generate the right partition structure. When noise levels were higher, AUTOCLASS had trouble estimating the parameters of the overlapping distributions. The result was a partition structure with fragmented groups. This explains the larger misclassification values.

4.2 Real Data

Our experiments with real data sets focus on the interpretability of the partition structures generated by SBAC. We exploit the characteristic of the Goodall measure of weighing features according to their commonality when computing similarity values. The importance of individual features in defining class structure can be directly linked to their assigned weights.

4.2.1 Data Description

Three representative data sets, one with numeric valued features, one with nominal valued features, and a third with a combination of numeric and nominal features were used in these experiments.

Hand Written Character (8OX) Data. Character-8OX (CH8OX) data is extracted from the well-known Munson hand printed Fortran character set. A hand written character is interpreted as a binary image placed on a 24x24 grid. The distance to the first cell of the character from the perimeter measured in eight directions, defines the eight numeric valued features for this data set. The 45 object data set used in this study has characters "8," "O," and "X," written three times each by five different authors. The overall goal is to partition the data into the three groups and

extract features from the individual groups for automated handwriting recognition.

Mushroom Data. This data was extracted from the mushroom database in the UCI data repository. The mushroom database consists of mushroom descriptions represented by 22 nominal features. Each data object belongs to one of two classes: *edible*(e) and *poisonous*(p). There are 23 species of mushrooms in this data set and we randomly picked a subset of 200 data objects making sure 100 of them were poisonous and 100 were edible. Further, the database source notes that there are no simple rules for determining the edibility of a mushroom. The goal of the clustering study is to separate the mushroom objects into the poisonous and nonpoisonous categories.

Heart Disease Data The heart disease data, generated at the Cleveland Clinic,[4] contains a mixture of nominal and numeric features. Heart disease refers to the build-up of plaque on the coronary artery walls that restricts blood flow to the heart muscle, a condition that is termed "ischemia." The end result is a reduction or deprivation of the necessary oxygen supply to the heart muscle. The data set consists of 303 patient instances defined by 13 features. Five of these

1. age,
2. resting blood pressure,
3. cholesterol,
4. maximum heart rate achieved during stress test, and
5. ECG ST depression,

are numeric-valued features, and eight:

6. gender,
7. chest pain type,
8. fasting blood sugar,
9. resting electrocardiograph,
10. chest pain during stress test,
11. ECG ST segment slope on exertion,
12. flouroscopy indication of calcification of major cardiac arteries, and
13. thalium scan

are nominal-valued features. The data comes from two classes: people with no heart disease and people with different degrees (severity) of heart disease.

The class labels available in the UCI repository were used for postevaluation in the interpretation study but they were not used as a feature in the clustering process.

4.2.2 Results

The clustering partitions generated by the SBAC system on the three data sets are summarized in Table 8.

Hand Written Character Data. The feature values in this data set are approximately normally distributed. The nature of the characters and the individual writing styles made the character pairs ("8," "X") and ("8," "O") hard to distinguish because six out of eight measurements for these two pairs of characters are quite similar. The data feature values indicate that character "X" should be quite distinct from the character "O."

4. The Cleveland Clinic heart disease database is maintained at the UCI repository for machine learning databases. Acknowledgment goes to Robert Detrano, MD, PhD as the principal investigator for the data collection study.

TABLE 8
SBAC Generated Partition Structures for the Three Data Sets

Data Sets	Clustering Partition
Character 8OX	C_1('X':13) C_2('O':15, '8':13, 'X':2) C_3('8':2)
Mushroom	C_1(e:71) C_2(e:29) C_3(p:100)
Heart Disease	C_1(f:126, s:37) C_2(f:38, s:102)

From the dendrogram, the partition extraction process generated a three cluster structure, where cluster one, C_1, contains 13 "X" characters, cluster two, C_2, contains a mixture of the three characters, and cluster three, C_3, contains two "8" characters. Looking further down into the dendrogram subtree rooted at C_2, reveals a subdivision into three clusters with C_{21} containing 15 "O" objects and one "8" object, C_{22} containing 10 "8" objects, and C_{23} containing two "8" objects and two "X" objects. Characters "8" and "O" separate into their own groups, but the similarity values between some of the hand written "8" and "X" objects contribute to the mixed cluster C_{23}.

To explain why SBAC groups the data objects into this partition structure, it is useful to study its feature value distributions from the clusters generated. Table 9 gives the mean values and the standard deviations of the eight features for clusters C_1, C_2, C_{21}, and C_{22}. It is clear that the significant differences between cluster pairs C_1 and C_2 and C_1 and C_{21}/C_{22} can be attributed to the four measurements along the top, bottom, left and right directions. The differences in measurements along the other four directions are small.

The similarity measure assigns a larger weight to numeric features with a larger spread in their values. For the four distinctive features the mean difference of their feature values between clusters C_1 and C_2 is 3.67 and this is much higher than the mean difference of the feature values between clusters C_{21} and C_{22}, which is 2.29. In other words, character "X" has feature values that distinguish it from characters "8" and "O," but the latter two have more similar feature values. Therefore, "X" easily separates from the

TABLE 9
Feature Distributions for Clusters Generated with the ch8OX Data: Mean (Standard Deviation)

Measure Direction	C_1 'X'	C_2 'O' and '8'	C_{21} 'O'	C_{22} '8'
top right	7.38(2.9)	6.85(1.6)	6.75(1.8)	7.00(1.0)
bottom left	7.07(2.3)	7.42(1.3)	7.25(1.0)	7.70(1.6)
top left	7.23(2.2)	6.19(0.9)	6.62(0.7)	5.50(0.8)
bottom right	5.84(1.6)	5.88(1.6)	5.87(0.6)	5.90(2.5)
left	9.00(0.8)	6.03(1.7)	5.06(1.3)	7.60(1.1)
right	9.69(0.9)	6.26(1.8)	5.37(1.5)	7.70(1.2)
top	6.84(2.1)	3.00(1.4)	3.81(1.1)	1.70(0.6)
bottom	7.69(2.8)	3.23(1.5)	4.06(1.1)	1.90(1.1)

TABLE 10
Feature Distribution for Clusters Generated with the Mushroom Data

Feature Name	edible		poisonous
	C_1	C_2	C_3
cap shape	**t, b**, x, f	**s**, x, f	x, f
cap surface	s, y	**f**, s	s, y
cap color	y, w, n	**g**, y, w, n	w, n
bruises	t	**f**, t	t
odor	a, l	**n**, a, l	**p**
gill spacing	c	**w**, c	c
gill size	b	b, n	n
gill shape	k, n, g, w, p	k, n, g, w, p	k, n, w, p
stalk shape	e	**t**, e	e
stalk root	**c, r**	**b**, e	e
stalk surface above ring	s	**f**, s	s
stalk surface below ring	**y**, s	**f**, s	s
ring type	p	**e**, p	p
spore print color	n, k	**u**, n, k	n, k
population	**n**, s, y	**a**, s, y	s, v
habitat	**m, p**, g	**d**, g, u	g, u

other two. For the remaining objects, the measurement differences for the "top" and "bottom" direction features become more significant and this helps separate the two characters into separate groups.

Mushroom Data SBAC generated a three cluster partition for the mushroom data: 100 poisonous mushrooms formed one group and the edible mushrooms separated into two clusters, one with 29 objects and the other with 71 objects. Table 10 lists the feature profiles of the poisonous mushroom group and the two edible mushroom groups. A quick glance reveals that the feature value distributions for the two types of mushrooms are quite similar.

The similarity measure of SBAC gives more weight to the rarer and more distinctive nominal valued features in the similarity calculation. Features such as cap shape, cap color, gill spacing, stalk shape, and ring type, values are shared between the two types of mushrooms, and do not play a significant role in the similarity computation. However, the values for the feature "odor" are disjoint for the two mushroom types. The similarity measure assigned a larger weight to this feature, which then played an important role in defining the partition structure derived.

As discussed earlier, the edible mushroom objects were randomly drawn from different edible mushroom categories from the larger data set. From Table 10, the most noticeable differences between the two clusters of edible mushrooms is the feature "stalk root," Cluster C_2 contains the values, c, r, and cluster C_3 includes the values b, e. Features like cap surface, gill size, and habitat also provide contrasts between the two groups of edible mushrooms.

Heart Disease Data SBAC produced a two cluster partition for the heart data. The feature value distributions for the two groups of patients appear in Table 11. For the numeric features, the mean value and standard deviations are listed in the table. Features (4) maximum heart rate achieved, (5) ECG ST depression, (7) type of chest pain experienced, (10) whether or not patient had chest pain during the stress test, and (13) number of cardiac arteries that have calcification had the most influence on the formation of the two cluster structure. On the other hand, features (3) cholesterol level, (8) fasting blood sugar, and (9) resting ECG provided no significant discriminating information in the partition structure.

A closer study of the profiles of these two groups of patients reveals that patients in group one have low risk and likely absence of heart disease. All the patients in this group showed no symptoms of ischemia during the stress test, which is significant in diagnosing the absence of heart disease. Also, indicators like no artery calcification seen on flouroscopy, ability to attain higher heart rates without distress during exercise, no chest pain exhibited on exertion, and normal thalium scan results suggest this group has low risk for heart disease. The presence of atypical anginal/nonanginal pain could be related to a variety of health factors (not just heart disease).

It is also significant that the second group had a higher average age and were mostly male. Statistically, this group is more susceptible to heart problems than younger individuals and females. Other supporting features include: calcification of cardiac arteries, which increases the risk of cardiac disease for this population since it contributes to the narrowing of the arteries, signs of cardiac ischemia on exertion during their stress test, inability to tolerate a high heart rate, and symptomatic chest pain.

Table 12 gives the clustering partitions generated by COBWEB/3, AUTOCLASS, and ECOBWEB for the three data sets. Comparing these to the SBAC results in Table 8, we observe that the AUTOCLASS partitions are fragmented, especially for the "Character 8OX" and "Heart Disease" data sets. For the "Heart Disease" data set, even small fragmented clusters have very mixed class labels. A similar phenomena is observed in the ECOBWEB results for the "Character 8OX" data. The partitions generated by COBWEB/3 are comparable to those generated by SBAC.

5 Discussion and Conclusions

Limitations of earlier methodologies and criterion functions in dealing with data described by mixed nominal and numeric features prompted us to look for a criterion function that would give better performance in clustering and discovery tasks. We have demonstrated that the similarity measure, initially proposed by Goodall [15], works well with data with mixed type features. The measure assigns greater weight to feature value matches that are uncommon in the population. For nominal valued features, an uncommon feature value match is assigned a greater similarity value than a more common feature value match. For numeric valued features, the uncommonality of feature value pairs is a function of the distances between pairs and the density of the population encompassed between the two values. A common framework is defined for incorporating individual feature similarity values for both numeric and nominal features. Illustrative examples are presented to demonstrate the properties of the similarity measure. An efficient

TABLE 11
Patient Profiles for the Heart Disease Data

	Feature Name	C_1	C_2
Numeric Feature	(1) age	52.1years(9.3)	57.0years(7.8)
	(2) resting blood pressure	130.0(16.3)	136.0(18.6)
	(3) cholesterol	246.6(56)	246.8(45)
	(4) maximum heart rate achieved	158.4bpm(17.5)	133.3bpm(23.9)
	(5) ECG ST depression	0.57mm(0.8)	1.58mm(1.2)
Nominal Feature	(6) gender	female/male	male
	(7) chest pain type	notang/abnang	anginal/non-anginal asymptomatic
	(8) fasting blood sugar	normal	normal
	(9) resting ECG	normal	normal
	(10) chest pain during stress test	false	true
	(11) ECG ST segment slope on exertion	upsloping	flat/down sloping
	(12) calcification of cardiac arteries	0 arteries	1, 2 or 3 arteries
	(13) thalium scan	normal	fixed or reversible defects

algorithm for computing the similarity values, especially for numeric features, has been developed.

The similarity measure is incorporated into an agglomerative clustering algorithm, SBAC and its performance is studied on real and artificially generated data. Comparative studies on the artificial data reveal the limitations of AUTOCLASS, COBWEB/3, and ECOBWEB in working with mixed data. On the other hand, SBAC works well with mixed data and is equally robust to noisy numeric and nominal features.

In previous work [2], we discussed systematic discretization methods to generate uniform data descriptions and avoid the limitations of the category utility measure in dealing with numeric features. This similarity measure goes a step further in that it retains the characteristics of probabilistic matching schemes and includes useful information like the actual separation in values of the numeric features. By studying contributions of different features in the similarity contributions, one can perform better interpretation studies with SBAC and the Goodall measure.

A challenging, yet promising, next step is to study variations of the form of the heuristic in Goodall's measure. For example, instead of assigning similarities based on uncommonality of the feature value matches, other expert-supplied heuristics may be employed to refine the proximity measure. This allows the incorporation of weighting

TABLE 12
Results from COBWEB/3, AUTOCLASS, and ECOBWEB on the Real Data Sets

Data Sets	COBWEB/3	AUTOCLASS	ECOBWEB
Character 8OX	C_1(8:5, O:2, X:15) C_2(8:10, O:3) C_3(O:10)	C_1(8:7, O:1, X:13) C_2(8:2, O:2) C_3(O:11) C_4(O:1, X:2)	C_1(8:2), C_2(X:2) C_3(8:2) C_4(8:2) C_5(8:3, O:8) C_6(8:1, O:3) C_7(O:2), C_8(X:2) C_9(X:2), C_{10}(X:2) 14 singletons
Mushroom	C_1(e:71) C_2(p:100, e:29)	C_1(p:100) C_2(e:54), C_3(e:17) C_4(e:11), C_5(e:18)	C_1(p:100, e:12) C_2(e:88)
Heart Disease	C_1(f:33, s:114) C_2(f:131, s:25)	C_1(f:2, s:37), C_2(f:2, s:7) C_3(f:25, s:8), C_4(f:8, s:1) C_5(f:17, s:2), C_6(f:6, s:3) C_7(f:11, s:4), C_8(f:1, s:8) C_9(f:6, s:10), C_{10}(f:6, s:2) C_{11}(f:10, s:4), C_{12}(f:4, s:3) C_{13}(f:4, s:10), C_{14}(f:1, s:3) C_{15}(f:13, s:1), C_{16}(f:1, s:4) C_{17}(f:4, s:10), C_{18}(f:4, s:1) C_{19}(f:7, s:6), C_{20}(f:3, s:1) C_{21}(f:9, s:3), C_{22}(f:2, s:1) C_{23}(f:11), C_{24}(s:9) C_{25}(f:1, s:1), C_{26}(f:6)	C_1(f:59, s:119) C_2(f:105, s:20)

schemes that incorporate the relevance of features in the context of the problem solving tasks. This notion has been discussed in Stepp and Michalski's Goal Dependency Network(GDN) [27], where task-oriented classification goals are employed in selecting relevant features, constructing new features and capturing the interrelationships between concepts. Similar ideas can be extracted from Kolodner's work on automatic indexing in organization and retrieval from long-term memory [18]. Kolodner discusses ways for characterizing indices by their predictive power and their uniqueness in describing desired features. From an interpretation viewpoint, these extensions may make the clustering scheme a much more effective tool for customized data mining tasks.

Recent approaches have used symbolic rules to bias the cluster formation process. Thompson and Langley [29] performed case studies on the use of background knowledge in incremental concept formation. More recently, Talavera and Bejar [28] have employed rules expressed as first-order logic expressions to bias an agglomerative clustering algorithm. Merging of two clusters is disallowed if their definitions partially match two different rules specified in the background knowledge base. Once a cluster completely satisfies a rule, it is free to merge with other clusters. This work generalizes Explanation Based Learning (EBL) [24] approaches where the deductive scheme plays the primary role in the cluster or concept formation process and data objects have to be assigned class labels. The Talavera and Bejar algorithm uses an unsupervised learning approach where the inductive component is the primary driver of cluster (concept) formation process. Prior knowledge just biases the data-driven search process. A similar method by Wagstaff and Cardie [30] uses prespecified "must-link" and "cannot-link" constraints among the data objects to bias the cluster formation process. The SBAC algorithm differs from these methods in that the bias is built into the criterion function. The assumption is prior knowledge in the form of rules is not available in the domain of study, therefore, weaker and more domain-independent heuristics are employed to bias the search for partition structure. This technique is more applicable in novel situations where very little knowledge is available on task and domain structure, or the goal of the study is to discover new ways of structuring information in a problem-solving domain.

Another interesting direction followed by Beck et al. [1], applies conceptual clustering [8] methods to database schema generation. They combine the use of EBL [23] and case-based reasoning [19] to match new objects to existing categories (class definitions in the database) and to create new category definitions when exceptions are detected (i.e., a good match cannot be established). In future work, it will be interesting to see how the weighting features of the similarity measure employed in SBAC may be used for category matching and for defining alternate models when a poor fit between an object and the existing categories is detected. Identifying the features that contribute most to the mismatch, as we did in our interpretation studies may provide a first step in generating new category definitions.

ACKNOWLEDGMENTS

The authors would like to thank Jerry Weinberg for helping interpret the results of the heart disease data set. They would also like to thank those responsible for contributing to and maintaining the Univerity of California at Irvine repository of machine learning databases. Anonymous reviewers provided a number of helpful suggestions for improving the manuscript.

REFERENCES

[1] H.W. Beck, T. Anwar, and S.B. Navathe, "A Conceptual Clustering Algorithm for Database Schema Design," *IEEE Trans. Knowledge and Data Eng.,* vol. 3, pp. 396-411, 1994.

[2] G. Biswas, J. Weinberg, and C. Li, "ITERATE: A Conceptual Clustering Method for Knowledge Discovery in Databases," *Artificial Intelligence in the Petroleum Industry,* B. Braunschweig and R. Day eds., pp. 111-139, 1995.

[3] G. Biswas, J. Weinberg, and D.H. Fisher, "ITERATE: A Conceptual Clustering Algorithm for Data Mining," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 28C, pp. 219-230, May 1998.

[4] P. Cheesman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, "Autoclass: A Bayesian Classification System," *Proc. Fifth Int'l Conf. Machine Learning,* June 1988.

[5] P. Cheesman and J. Stutz, "Bayesian Classification (AUTOCLASS): Theory and Results," *Advances in Knowledge Discovery and Data Mining,* 1995.

[6] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. Series B,* vol 39, no. 1, pp. 1-38, 1977.

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

[8] D.H. Fisher, "Knowledge Acquisition via Incremental Conceptual Clustering," *Machine Learning,* vol 2, pp. 139-172, 1987.

[9] D.H. Fisher, "Iterative Optimization and Simplification of Hierarchical Clusterings," *J. Artificial Intelligence Research,* vol. 4, pp. 147-180, 1996, http://www.cs.washington.edu/research/jair/.

[10] D.H. Fisher and P. Langley, "Methods of Conceptual Clustering and Their Relation to Numeric Taxonomy," *Proc. Artificial Intelligence and Statistics,* W. Gale, ed., 1986.

[11] R.A. Fisher, *Statistical Methods for Research Workers.* 13th ed. Edinburgh and London: Oliver and Boyd, 1963.

[12] W.J. Frawley, G.P. Shapiro, and C.J. Matheus, "Knowledge Discovery in Databases: An Overview," *Knowledge Discovery in Databases,* G. Piatetsky-Shapiro and W. J. Frawley eds., pp. 1-27 Menlo Park, Calif.: AAAI/MIT Press, 1991.

[13] J.H. Gennari, P. Langley, and D.H. Fisher, "Models of Incremental Concept Formation," *Artificial Intelligence,* vol. 40, pp. 11-61, 1989.

[14] M. Gluck and J. Corter, "Information, Uncertainty, and the Utility of Categories," *Proc. Seventh Ann. Conf. Cognitive Soc.,* pp. 283-287, 1985.

[15] D.W. Goodall, "A New Similarity Index Based On Probability," *Biometrics,* vol. 22, pp. 882-907, 1966.

[16] S.J. Hanson and M. Bauer, "Conceptual Clustering, Categorization, and Polymorphy," *Machine Learning,* vol. 3, pp. 343-372, 1989.

[17] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs: Prentice Hall, 1988.

[18] J.L. Kolodner, "Maintaining Organization in a Dynamic Long-Term Memory," *Cognitive Science,* vol. 7, pp. 243-280, 1983.

[19] *Workshop on Case-Based Reasoning.* J.L. Kolodner, ed., San Mateo, Calif.: Morgan Kaufmann, 1988.

[20] H.O. Lancaster, "The Combining of Probabilities Arising from Data in Discrete Distributions," *Biometrika,* vol. 36, pp. 370-382, 1949.

[21] C. Li and G. Biswas, "Knowledge-based Scientific Discovery from Geological Databases," *Proc. First Int'l Conf. Knowledge Discovery and Data Mining,* pp. 204-209, Aug. 1995.

[22] K. McKusick and K. Thompson, "COBWEB/3: A Portable Implementation," Technical Report FIA-90-6-18-2, NASA Ames Research Center, 1990.

[23] S. Minton, J. Carbonell, C. Knoblock, D. Kuokka, O. Etzioni, and Y. Gil, "Explanation Based Learning: A Problem Solving Perspective," *Artificial Intelligence,* vol. 40, pp. 63-118, 1989.

[24] D. Ourston and R.J. Mooney, "Theory Refinement Combining Analytic and Empirical Methods," *Artificial Intelligence,* vol. 66, pp. 311-344, 1994.
[25] Y. Reich, "Building and Improving Design Systems: A Machine Learning Approach," PhD thesis, Dept. Civil Eng., Carnegie Mellon Univ., 1991.
[26] Y. Reich and S.J. Fenves, "The Formation and Use of Abstract Concepts in Design," *Concept Formation: Knowledge and Experience in Unsupervised Learning,* D.H. Fisher, M.J. Pazzani and P. Langley, eds., pp. 323-352, Los Altos, Calif.: Morgan Kaufmann, 1991.
[27] R.E. Stepp and R.S. Michalski, "Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects," *Machine Learning: An Artificial Intelligence Approach,* R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds., vol. 2, pp. 471-498, Los Altos, Calif.: Morgan Kaufmann, 1986.
[28] L. Talavera and J. Bejar, "Integrating Declarative Knowledge in Hierarchical Clustering Tasks," *Proc. Int'l Symp. Intelligent Data Analysis,* pp. 211-222, 1999.
[29] K. Thompson and P. Langley, "Case Studies in the Use of Background Knowledge: Incremental Concept Formation," *Proc. AAAI-92 Workshop Constraining Learning with Prior Knowledge,* pp. 60-68, 1992.
[30] K. Wagstaff and C. Cardie, "Clustering with Instance-Level Constraints," *Proc. 17th Int'l Conf. Machine Learning,* pp. 1103-1110, June 2000.
[31] M. Zemankova and A. Kandel, "Implementing Imprecision in Information Systems," *Information Sciences,* vol. 37, pp. 107-141, 1985.

Cen Li received the BS degree in computer science from Middle Tennessee State University in 1993. She received the MS and the PhD degrees in computer science from Vanderbilt University in 1995 and 2000, respectively. Her research interests include machine learning, data mining, and statistical data analysis. Currently, she is an assistant professor in the Computer Science department at Middle Tennessee State University.

Gautam Biswas received the PhD degree in computer science from Michigan State University in Lansing, Michigan. He is an associate professor of computer science and engineering and management of technology at Vanderbilt University. He conducts research in intelligent systems, with primary interests in hybrid modeling and analysis of complex embedded systems and their applications to diagnosis and fault-adaptive control. He is also involved in developing simulation-based environments for learning and instruction. More recently, he has been working on developing computer-based teachable agents that motivate deep thinking and reflection in learning by teaching and collaborative problem solving. Other research areas include Hidden Markov Model techniques for clustering of temporal data sequences and decision-theoretic planning and scheduling for intelligent manufacturing systems. His research is currently supported by funding from The Defense Advanced Research Projects Agency (DARPA), US National Aeronautics and Space Agency (NASA), US National Science Foundation (NSF), and the US Office of Naval Research. Dr. Biswas has served on the program committee of a number of conferences. He was chair of the 1997 International Joint Conference on Artifical Intelligence Workshop on Engineering Problems for Qualitative Reasoning, cochair of the 1996 Principles of Diagnosis Workshop, the 1999 American Association for Artifical Intelligence, Spring Symposium on Hybrid Systems and Artifical Intelligence, the 2001 Workshop on Qualitative Reasoning, senior program committee for AAAI-97 and AAAI-98, and technical committee cochair for the 2000 IEEE SMC conference. He is currently an associate editor for the *IEEE Transactions on Systems, Man, and Cybernetics*, *IEEE Transactions on Knowledge and Data Engineering*, and the *International Journal of Applied Intelligence*. He is a senior member of the IEEE and a member of the IEEE Computer Society, ACM, AAAI, and the Sigma Xi Research Society.