ITERATE: A Conceptual Clustering Algorithm for Data Mining

Gautam Biswas, Senior Member, Jerry B. Weinberg, Member, and Douglas H. Fisher, Member.

Abstract— **The data exploration task can be divided into three interrelated subtasks: (i) feature selection, (ii) discovery, and (iii) interpretation. This paper describes an unsupervised discovery method with biases geared toward partitioning objects into clusters that improve interpretability. The algorithm, ITERATE, employs: (i) a data ordering scheme and (ii) an iterative redistribution operator to produce maximally cohesive and distinct clusters. Cohesion or intra-class similarity is measured in terms of the match between individual objects and their assigned cluster prototype. Distinctness or inter-class dissimilarity is measured by an average of the variance of the distribution match between clusters. We demonstrate that interpretability, from a problem solving viewpoint, is addressed by the intra- and inter-class measures. Empirical results demonstrate the properties of the discovery algorithm, and its applications to problem solving.**

Keywords— **knowledge discovery, data mining, conceptual clustering, concept formation, criterion function, order bias, iterative redistribution.**

I. Introduction

In recent years, technology has advanced to the point where electronic data collection and storage have become tasks that can be accomplished easily. However, the abundance of available data forges new problems, such as how to effectively and efficiently analyze this data using automated mechanisms to better understand, characterize, and validate known phenomena and trends, as well as discover new and interesting phenomena. Frawley, Piatetsky-Shapiro, and Matheus [21] cite examples of a number of forward-looking companies that are developing tools and techniques, mainly based on ID3-like classifier systems [35] to analyze their databases for interesting and useful *patterns*. For example, American Airlines uses knowledge discovery techniques to periodically search its frequent flyer database to find profiles of its better customers and target them for specific promotions. General Motors uses classification methods to study its automotive troubleshooting databases and derive diagnostic expert systems for its different car models.

Knowledge discovery is defined as *"the non trivial extraction of implicit, previously unknown and potentially useful information in data"* [42]. An architecture for a knowledge discovery system, which begins with raw data and ends with useful knowledge, includes a number of processing steps, such as data selection, feature-value selection and transformation, incorporation of prior knowledge, and interpretation [12]. At the center of the system is a discovery or "data mining" method that extracts and evaluates data groupings, patterns, and relationships in the context of a problem solving task. Discovery methods for extracting patterns from data are typically based on heuristic search methods, which have roots in statistical analysis, numerical taxonomy, and conceptual clustering methods. While the first two have been successfully applied to numeric data[1], conceptual clustering has been applied to a mixture of numeric, ordinal, and cardinal (symbolic) data.

Typically, an exhaustive search for structure is an exponential problem, therefore, clustering algorithms incorporate *biases* that guide the search process in specific areas of the total hypothesis space of possible groupings or relationships among a given set of data [8], [34]. A lot of the work in conceptual clustering has developed biases from a cognitive modeling viewpoint. Actual problem solving applications of conceptual clustering algorithms have focused on the task of classification and flexible prediction, i.e., predicting missing feature values in specific contexts with high accuracy [14].

In this paper we describe a conceptual clustering algorithm, ITERATE, whose biases are specifically adapted to the process of discovering interesting patterns from data. When interpreted in the context of a domain or problem solving situation this results in potentially useful and possibly new knowledge. The essential task here is *interpretation* of the generated patterns, and this is best addressed by creating groups of data that demonstrate cohesiveness within but clear distinctions between the groups.

There are a number of processing steps in the overall knowledge discovery task, but, in our framework, we consolidate these tasks into three key subtasks: (i) feature selection, (ii) discovery, and (iii) interpretation. Feature selection deals with issues for characterizing the data to be studied in the context of the current problem solving situation. This includes making decisions about which features are relevant descriptors, and the level of specification that values of these features will assume. A knowledge discovery framework should facilitate iteration among fea-

Manuscript received xx x, xxxx; revised yy yy, yyyy.

G. Biswas is with the Department of Computer Science, Box 1679 Sta B, Vanderbilt University, Nashville, TN 37235. E-mail: biswas@vuse.vanderbilt.edu.

J.B. Weinberg is with the Dept. of Computer Science , Southern Illinois University at Edwardsville, Edwardsville, IL 62026. E-mail: jweinbe@siue.edu.

Doug Fisher is with the Department of Computer Science, Box 1679 Sta B, Vanderbilt University, Nashville, TN 37235. E-mail: dfisher@vuse.vanderbilt.edu.

Biswas and Weinberg were partially supported by grants from Amoco and Arco Research Labs.

[1]An exception to this is a measure of similarity developed by Goodall [24] that attempts to add ordinal and distance information to nominal values using a probability analysis and prior domain knowledge. The SBAC system has adapted this measure for knowledge discovery [30]

ture/value selection and pattern discovery subtasks. This allows knowledgeable users and domain experts to conduct empirical studies in the knowledge discovery process by creating different characterizations of the data. This method suggests that the feature selection subtask is an integral part of the discovery process, but we do not discuss this subtask in this paper. The discovery subtask involves the derivation of structure or groupings among a set of data objects in a particular context or problem solving situation. Groupings or data objects can be based on metric-based measures of similarity and dissimilarity [26], or probabilistic measures of similarity based on counts of identically-valued features [9], [14]. Some methods evaluate groupings based on measures of parsimony, interestingness, novelty, and understandability. Such criteria can be subjective or objective [36]. The interpretation subtask assigns meaning to the groupings discovered [17]. While some knowledge discovery systems attempt to validate knowledge in terms of known domain theories [28], in general, this subtask is usually left to domain experts who relate derived class structure of data to domain theory, and, more specifically, to the current problem solving situation. This process can also lead to discovery of new concepts that improve problem solving and extend domain theories [12], [13], [27].

The interpretation process is directly linked to the overall goal of improving problem solving performance in the domain of interest. In the knowledge discovery framework, *interpretability* relates to the qualities of the partition structure of the data generated. In our framework, we focus on two qualities that impact the utility of concepts generated:

1. *concept distinctness* or *inter-class dissimilarity*. Distinctness of two concepts is defined in terms of the differences in their prototypical class descriptions. More distinct concepts produce better problem space decompositions, resulting in greater problem solving efficiency [37].
2. *cohesion* or *intra-class similarity*. This is defined in terms of how individual objects match the prototypical description of the class they are assigned to by the algorithm. The ability to classify instances and make inductive inferences increases with the similarity of the instance to the class prototype (this is also called *central tendency* [1]). Classes that exhibit high class cohesion improve the specific classification of previously unseen instances.

We believe that partitioning schemes that maximize intra-class similarity and inter-class dissimilarity measures generate more *interpretable* partitions. Two control operators, *anchored dissimilarity ordering* and *iterative redistribution*, are developed within the ITERATE framework to improve cluster cohesion and distinction.

To evaluate ITERATE's performance, we define two *a posteriori* measures: (i) *cohesion*, and (ii) *distinctness* that directly link to intra-class similarity and inter-class dissimilarity, respectively. Cohesion measures how well a cluster prototype predicts feature values of its member objects. Distinctness compares the average predictability of all feature values across two different class descriptions. A measure called the *variance of the distribution match* is defined to capture distinctness. For two perfectly distinct clusters, the distinctness equals the sum of their cohesion values. ITERATE's ability to converge on the more interesting parts of the partition search space (high intra-class similarity and inter-class dissimilarity) is studied.

In addition to performing experiments that demonstrate the hypothesized biases of ITERATE, we conduct two additional studies that focus on interpretation and problem solving. A dataset of mineral samples is clustered, and the resulting partition structure is interpreted using expert-supplied domain knowledge. ITERATE is also evaluated in the context of a decision task. The partitioning structure created by ITERATE is applied to the problem solving task, and a comparison is made of system performance with and without the use of the partition structure.

Section 2 discusses the biases of ITERATE's criterion function and suggests a control structure that exploits this bias to generate cohesive and distinct clusters. Section 3 describes the ITERATE algorithm. Section 4 presents the *a posteriori* evaluation measures and discusses the results of the experiments conducted. Section 5 presents the summary and conclusions.

II. Clustering Systems and Criterion Functions

In clustering schemes, data objects are represented as vectors of *feature-value* pairs. Features represent properties of an object that are relevant to the problem-solving task. For example, if we wish to classify automobiles in terms of the speeds they can achieve, body weight, body shape, and engine size are relevant features, but color of the car body is not. Feature vectors may be a combination of numeric and non-numeric descriptors. If one looks at geological data, features such as age, porosity, and permeability are numeric-valued, whereas descriptors, such as rock type and facies structure are non numeric and nominal-valued. Therefore, it becomes important to deal with algorithms that can work with a combination of numeric- and nominal-valued data. The best way to combine nominal, ordinal, and numeric valued features is still an open question.

Numerical taxonomy methods use pairwise relations between numerical feature-valued objects stored in a proximity matrix as the basis for defining groups or clusters. If the objects are defined as points in a multi-dimensional metric space, measures such as the Euclidean and Mahalanobis metrics are used to define dissimilarity between objects. Cluster analysis methods can be parametric or non-parametric, and hierarchical or partitional [26].

Conceptual clustering methods have primarily focused on data objects described as nominal-valued features or mixed nominal/numerical-valued features, and typically rely on non-parametric probabilistic measures to define groupings. CLUSTER/2 [32], bases its criterion function on measures of *common attribute values within a cluster*, *non intersecting attribute values between clusters*, and *simplicity of the conjunctive expression for describing a cluster*. UNIMEM [29] builds a classification tree based on

a *Hamming* distance[2] measure which mainly focuses on intra-cluster similarity. WITT [25], defines intra cluster similarity in terms of the strength of pairwise attribute relationships (*co-occurrences*) that exist within a cluster or group. The strength of these relationships is defined in terms of correlational measures that are represented as contingency tables. AUTOCLASS [9] and COBWEB [14] define classes as a probability distribution over the attributes of the objects. AUTOCLASS, a parametric scheme, adopts the Bayesian classification approach. Fisher's COBWEB uses the *category utility* measure developed by Gluck and Corter [23] to predict the preferred level of categorization in human hierarchical organizations.

The creation of a taxonomy is the end result of cluster analysis. The next step is to relate nodes of the classification hierarchy to salient concepts in the domain of interest. This is usually accomplished by characterizing each group in terms of a general description, which depends on the nature of the *bias* used in the clustering algorithm [8], [38]. The bias has been defined as "the set of all factors that collectively influence hypothesis definition"(i.e., the nature of the clusters or groupings formed). These factors include the definition of the hypothesis space and the algorithm that searches this space for concept descriptions [8]. These, in turn, can be expressed in terms of a *criterion function* chosen to influence grouping and concept formation and the control structures that guide the search for derivation of structure in the data.

Our focus is on non-parametric conceptual clustering schemes. We start off with the *category utility* criterion function because it tends to address the tradeoff between *intra-cluster similarity* and *inter-cluster dissimilarity* in evaluating partitional structures. A study of the biases of the category utility function led us to develop a scheme for pre-ordering the data objects before generating a partition structure. This approach biases the result toward more cohesive groupings.

Category Utility: Analysis of its Bias

In conducting research on cognitively preferred levels of categorization (basic level phenomenon), Gluck and Corter [23] adopted a *probability matching strategy* to establish the usefulness or utility of a category. They defined the Category Utility (CU) of a class C_k as:

$$CU_k = P(C_k)\left\{ \sum_i \sum_j [P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2] \right\}, \quad (1)$$

where $P(A_i = V_{ij})$ is the probability of feature A_i taking on value V_{ij}, and $P(A_i = V_{ij} \mid C_k)$ is the conditional probability of $A_i = V_{ij}$ in class C_k. This represents an increase in the number of feature values that can be correctly guessed for class C_k $(P(A_i = V_{ij}|C_k)^2)$, over the expected number of correct guesses given that no class information is available $(P(A_i = V_{ij})^2)$. 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:

$$\frac{\sum_{k=1}^{K} CU_k}{K} \quad (2)$$

Gluck and Corter demonstrated the efficacy of category utility in predicting the preferred level of categorization given a pre-existing classification hierarchy. Fisher adapted this probability matching measure to develop a conceptual clustering algorithm called COBWEB [14], which, given a set of objects expressed as feature-value vectors builds a classification tree. COBWEB uses a greedy incremental approach to build a hierarchy by incorporating data objects one-at-a-time into an existing hierarchy structure. A data object is placed into a level of the hierarchy using one of two operations: (i) classify data object into an existing class or (ii) create a new class. The operation that produces a partition with the higher partition score is the one applied to update the partition. The data object is recursively classified until the object is placed in a singleton class (a class consisting of a single instance). Empirically, Fisher demonstrates that the method does well in modeling basic level phenomena and the resulting classification trees perform well in flexible prediction tasks [15], [16].

A well-studied characteristic of greedy, incremental algorithms is their order dependency. Their control structures generate different classification trees for different data orders. For example, the CU function represents a tradeoff between size, $P(C_k)$, and cohesiveness (Co) or predictive accuracy of feature-values, $[(\sum_i \sum_j (P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2)]$ of a class or category. The term $P(C_k)$ causes a bias toward larger categories, and data orderings with consecutive presentation of a group or groups of highly similar data objects may skew the partition structure. A number of studies [4], [22], [31] have examined the effects of order-dependency on classification tree structure and concept formation. Gennari, et al. [22] and McKusick and Langley [31] have established that the COBWEB control structure and evaluation function are oriented toward "maximizing predictive accuracy, (and) the hierarchies it constructs may not reflect the underlying class structure of the domain" [31]. What this implies is that it is likely to produce spurious intermediate nodes in the classification trees [31], which can cause unnecessary *fragmentation* in the final partitions. Fragmentation makes it difficult to interpret and extract useful information from a partition structure.

An analysis of the partition score in an extreme case of skewed data presentation reveals the nature of the bias. Consider the situation where $m + n$ instances have already been classified in an emerging partition, creating $n + 1$ classes, where n classes are singletons[3] (cohesiveness Co_{single}) and one class has m objects (cohesiveness Co_m). This situation illustrates an extreme case of uneven class

[2]The Hamming distance is the same as the Manhattan metric when all feature values are considered to be binary, i.e., present or absent.

[3]A singleton concept maximizes the CO value since $P(A_i = V_{ij} \mid C_k) = 1$ for all attributes. Also, the CO value for all singletons in a partition are equal.

growth[4]. In such a situation, it can be shown that:

$$(m + n + 2)Co_{ms} = Co_{single} \quad (3)$$

As expected, the size $(m + 1)$ of the larger class biases the addition of the data instance to the class. However, the larger class is actually favored by the much larger factor $(m + 1) + (n + 1)$, i.e., the size of the class plus the size of the partition. Intuitively, the measure has a bias toward limiting the number of classes, even when the new data instance has only a small number of feature values in common with the larger class. A more complete analysis of this phenomenon is discussed elsewhere [39].

A number of attempts have been made to mitigate the effects of order dependency in the emerging partition structures. Most of these come under the auspices of reclassification methods [18]. The COBWEB system described in [14] introduced *split* and *merge* operators to mitigate order effects. The split operator divides a category in a partition into its sub-categories thereby promoting them in the hierarchy. Conversely, the merge operator combines a number of categories into a more general super-category thereby demoting the combined categories in the hierarchy. Both operators are applied locally within each partition, with the merge operator being applied to all pairwise combinations of categories. The operation that produces the largest increase in the partition score is applied to generate a modified hierarchy. McKusick and Langley [31] and Fisher [14] have experimented with *promote* operators which extend the split and merge operators. This operator promotes a class or grouping that is more similar to an ancestor than its direct parent, and allows subsequent redistribution to place these objects in proper categories lower down in the hierarchy.

Fisher, Xu, and Zard [18] introduced a non-incremental control structure in a system called AGGLOM. This mirrors traditional agglomerative approaches like single- and complete-link clustering, where two classes whose combination produces the best change in CU are merged into a single class. Initially, each data object forms a singleton class, and the agglomerative procedure results in a binary tree. The second step is to traverse the tree top down, applying the split operator in an attempt to improve the partition score of each level. By extending the search, AGGLOM removes the order dependency, but the size bias of the CU function could still affect cluster formation. An analysis of this approach reveals that larger classes are favored by an order of m, where m is the size of the class. Therefore, given a data set that represents a class structure that contains both very cohesive classes and weakly cohesive classes, AGGLOM will similarly derive skewed class structure [39].

We adopt a different viewpoint in developing ITERATE. Rather than attempting to mitigate the order dependency effects and maximize the partition score (average category utility), we use the phenomenon as an opportunity to manage both skewed order presentations and skewed data sets. This is addressed by manipulating the data order so that the cohesiveness factor plays a more important role in the early partition formation process. Previous work shows that interleaved[5] orders produce better classification trees and better final groupings in terms of the rediscovery task and interpretation task [4], [5].

To exploit the size bias of the CU function, data objects are ordered using the ADO (Anchored Dissimilarity Ordering) algorithm[6]. The object chosen to be next in the order is the one that maximizes the sum of the Manhattan distance between it and the previous n objects in the order. The Manhattan distance between two objects defined by nominal-valued attributes is simply the number of differences in the attribute-value pairs. The window size, n, is user defined, but empirically corresponds to the actual number of classes expected in the data. The first data instance in the order (i.e., the anchor) is chosen as the instance most dissimilar from a prototypical instance of the entire data set of the node.

III. The ITERATE Algorithm

It has been demonstrated in the partitional numeric clustering schemes that different initial partitions can lead to different final clusterings based on the square error converging to local minima, and this is especially true if the groupings are not well separated [26]. Therefore, choice of a "good" initial partition is of primary importance in obtaining the best grouping possible. It has also been suggested that the results of a hierarchical clustering scheme can be used to select the initial partition, especially if the nature of the data set is not well known [26]. This idea can be exploited by extracting initial partitions[7] from a classification tree and then introducing a partitive control structure [26] in the form of an *iterative redistribution* operator to further refine the partition structure. The ITERATE algorithm adopts this approach by combining hierarchical and partitional control schemes to generate *cohesive* and *maximally distinct* clusters.

The algorithm has three primary steps:

1. derive a classification tree using category utility as a criterion function for grouping instances.
2. extract a "good" initial partition of data from the classification tree as a starting point to focus the search for desirable groupings or clusters, and
3. iteratively redistribute data objects among the groupings to achieve maximally separable clusters.

Given our focus on data mining, the design of ITERATE is governed by four important assumptions about the

[4]This is not unusual in real data sets, e.g., the mushroom data set we used for evaluating ITERATE's performance has a large cohesive class of poisonous mushrooms, and a number of less cohesive classes of edible mushrooms. A similar observation can be made of the Iris data set.

[5]Interleaved corresponds to an order where objects from different classes are presented in sequence in an attempt to obtain a maximally dissimilar ordering among the objects.

[6]This extends an ordering algorithm developed by Xu [41], [19].

[7]The concept of creating initial partitions can be linked to pruning methods [16] and extraction of basic level categories [15].

nature of the data: (i) a large percent of the attributes describing data objects are nominal-valued[8], (ii) no pre-existing classification is available for the data being analyzed, (iii) the data objects to be classified are available at the start of the clustering process, and (iv) the size of the database is large enough to make computational efficiency of the clustering algorithm a major concern.

A. Derivation of Classification Tree

Numeric partitional clustering algorithms form an initial partition by randomly specifying k seed points. A better way is to use a hierarchical clustering scheme to direct initial partition formation [11]. We use this approach and create a classification tree as the first step in defining a "good" initial partition.

The algorithm for generation of the classification tree is summarized below:

Initialize: **Set** $L = N_1$,
  O_1 = set of data objects to be clustered.
 Loop till L empty
  Get first element from L, say N_k
  If O_k contains more than one object
   Sort objects according to anchored
    dissimilarity ordering (ADO) scheme
   Loop for every object in O_k
    If first object in O_k
     Create new node N_{k+c} as child of N_k
     Place object in node N_{k+c}
    Else:
     (i) **Place** object in all child nodes one by one,
      and compute partition scores for each one
     (ii) **Place** object as new child node under N_k
      and compute partition score.
    Assign object to node for which the
     partition score is highest, and
     update $A_i = V_{ij}$ count for node.
   End Loop
   Place new children in L.
  End If
 End Loop

N_i represents a node in the classification tree, and O_i defines the set of data objects in node N_i. N_1 is the root of the tree and O_1 is the set of data objects to be clustered. The creation of the classification tree uses a simple partitional control scheme to divide each group of data objects into sub classes, starting with the entire group as the root of the tree. The algorithm uses the partition score (equation 2) to determine if an object is to be placed in one of the existing groups or becomes a new group on its own. The tree is constructed in a breadth-first manner, where each level of the tree is completed prior to creating the next lower level.

[8]The method for handling numeric valued features in the ITERATE algorithm is discussed in [6].

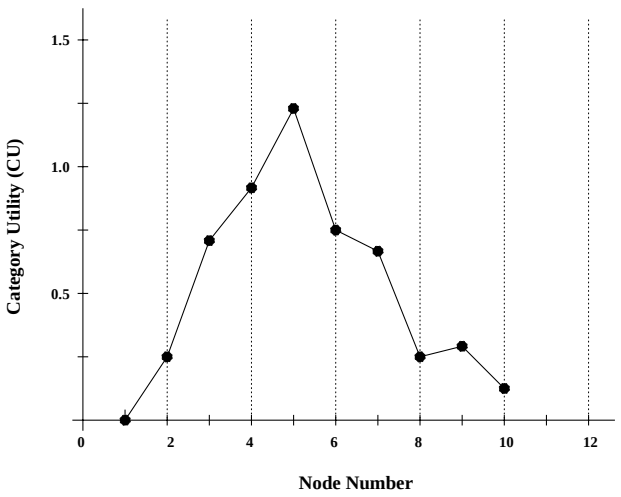


Fig. 1. CU values along a path of a Classification tree

Prior to dividing a group of data objects into a sub-class structure, the group of objects is ordered using Anchored Dissimilarity Ordering (ADO) to exploit the biases of the category utility measure as discussed in the last section.

B. Extraction of Initial Partition

The initial partition structure is extracted by comparing the CU value (equation 1) of classes or nodes along a path in the classification tree. For any path from root to leaf of a classification tree this value initially increases, and then drops (see Fig. 1). Classes from the level below the point where the CU value falls, can be considered to overfit the data, and, therefore, not useful for the clustering task.

Using this as the basis, the algorithm for generating the initial partition is outlined below[9]:

Initialize: set **node** N = root
 set **list** = $\{N\}$
 Loop till **list** empty
  get **children**(N)
  if
   $CU_N > CU_c \; \forall \; c \in$ **children**(N)
    N defines a group in the initial partition (i)
  else
   $\forall c$ such that $c \in$ **children**(N) and $CU_N \leq CU_c$
    add c to **list** (ii)
   $\forall c$ such that $c \in$ **children**(N) and $CU_N > CU_c$
    make c a group in initial partition (iii)
  end if
  remove N from **list**
  set N = first element of **list**
 End Loop

Step (i) of the algorithm ensures that a node at which CU_k peaks will be included in the initial partition. Note that this algorithm is conservative. It prefers more specific

[9]Conceptually the tree generation step and partition extraction step are described separately, for efficiency though, the current ITERATE implementation combines these steps.

to more general concepts (clusters) in forming the initial partition. Along a particular path from root to leaf, if a child node has a greater CU value than the node itself, this node is not picked to be a component of the initial partition (Step (ii)). If the CU value decreases for other children of this node, those children are picked for the initial partition (Step (iii)). Once a node is picked along a path, no other nodes below this node can be included in the initial partition. Thus the algorithm ensures that no concept subsumes another.

The implication of the conservative approach is that seeds picked from lower levels of the classification tree represent more specific concepts, and, therefore, tend to be more cohesive in their attribute value specifications. If these concepts are significant, they remain in the final partition, otherwise they may be merged into other groups during iterative redistribution. Empirical studies in Section 4 illustrate this fact.

C. Iterative Redistribution: Improving Partition Structure

The iterative redistribution operator is applied to maximize the cohesion measure for individual classes in the partition. The redistribution operator assigns object d to class k for which the category match measure CM_{dk} (equation 4) is maximum. When ties occur, and the data object's current class is a contender, the object is retained in the same class. Otherwise ties are broken arbitrarily. A redistribution iteration consists of determining each object's assignment and updating the partition based on that assignment. The redistribution operator is applied iteratively till quiescence.

Partitive schemes like ISODATA [2] adopt a *square-error* criterion function, with the Euclidean metric providing the measure for computing distance between data objects and cluster centers. Data objects are reassigned to optimize a chosen criterion function, usually the *mean-square error*. With nominal-valued data, the match between an object d and a class k is defined as a probabilistic similarity measure, called the *category match measure* [17]:

$$CM_{dk} = P(C_k) \sum_{i,j \in \{A_i\}_d} (P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2), \tag{4}$$

Note that a class C_k is defined in terms of the conditional probability distribution of all feature values for the class, i.e., $P(A_i = V_{ij} \mid C_k)$, $\forall\ i, j$ in class C_k. Also, the category match measure assumes that a data object has only one value per attribute (represented by $j \in \{A_i\}_d$ in the above equation). Category match measures the increase in expected predictability of class C_k for the attribute values present in data object d.

IV. Experimental Results

A primary goal of knowledge discovery is the interpretation of discovered concepts in the context of domain knowledge. We work on the premise that concept distinctness and cohesion are the key to creating effective and useful partition structures. The experiments reported here study the effects of ITERATE's biases on these two characteristics. Two additional studies examine the interpretation task in the context of specific problem solving domains. The first of these studies maps the partition structure generated by ITERATE on mineral samples into an expert-supplied classification structure. The second study focuses on the more complex problem solving task of characterizing hydrocarbon plays for the purpose of estimating the potential of oil reserves. This task was implemented as a knowledge-based system, PLAYMAKER [3], which uses over 500 expert-supplied rules to query users about geological characteristics of a region of interest to classify the play structure. This study looks at how the expert-supplied rules can be partitioned into a classification structure to better focus the query selection process. Focusing the query selection process results in the system seeking only relevant data, and the play structure and its hydrocarbon potential can be established quickly. Changes in system performance, when the derived structure was used to guide the problem solving process are discussed.

A. Evaluating Final Partitions

Corresponding to the two factors: (i) generate maximally cohesive clusters (intra-class similarity) and (ii) achieve maximum separability (inter-class dissimilarity) among the clusters in a partition, we define two post-hoc probabilistic measures of partition quality. Unlike mean-square error algorithms and schemes like CLUSTER/2 which define categorical evaluation measures, our evaluation measures are based on probability matching schemes, because ITERATE's cluster definitions are probabilistic.

Cohesion is measured as the increased predictability of each feature value of the objects in the dataset given the assigned class structure. The increase in predictability for an object d assigned to class k, M_{dk}, is defined as:

$$\sum_{i,j \in \{A_i\}_d} [P(A_i = V_{ij} \mid C_k)^2 \ - \ P(A_i = V_{ij})^2]. \tag{5}$$

The cohesion of the partition structure is measured as the sum of the M_{dk} values for all objects in the data set. This can be interpreted as the increase in match between a data object and its assigned cluster prototype over the match between the data object and the dataset prototype. Note that this is the second term of the CM measure (equation 4). The $P(C_k)$ term trades off fragmentation with cohesion during the class formation process. On the other hand, the evaluation measure concentrates solely on cohesion for defining the quality of the partition.

Distinctness of clusters in the final partition is measured as inter-class dissimilarity using a probability match measure termed the *variance of the distribution match*. Variance of the distribution match between classes k and l in a given partition is measured as:

$$\frac{1}{n} \sum_{i}^{n} \sum_{j} [P(A_i = V_{ij} \mid C_k) - P(A_i = V_{ij} \mid C_l)]^2. \tag{6}$$

The greater this value, the more dissimilar are the two classes being compared, and, therefore, the concepts they

Class	Type	Minerals
D1	Halide	Halite, Fluorite
D2	Oxide	Periclase, Corundum, Rutile, Cassiterite, Spinel
D3	Hydroxide	Diaspore, Brucite, Gibbsite
D4	Carbonate	Calcite, Dolomite, Magnesite, Siderite
D5	Sulfate	Barite, Celestite, Anhydrite, Gypsum
D6	Phosphate	Monazite, Apatite, Dahhlite
D7	Silica	Quartz
D8	Feldspar	Orthoclase, Sanidine, Microclase, Albite, Oligoclase, Andesine, Labradorite, Bytownite, Anorthite
D9	Feldspathoid	Leucite, Nepheline, Cancrinite, Sodalite
D10	Inosilicate	Enstatite, Hypersthene, Diopside, Augite

TABLE I
Mineral Data: Grouping by Chemical Composition

Facies Structures	
Aeolian	AEO
Alluvial Fan	AFN
Basin	BAS
Barrier Beach	BBE
Delta	DEL
Estuarine	EST
Fluvial	FLU
Glacial	GLA
Lacustrian	LAC
Shelf	SHF
Slope	SLP
Submarine Fan	SFN
Tidal Flat	TFL

Geological Attributes	
Depositional setting	DSE
Primary Bedding Type	PBT
Primary Bedding Shape	PBS
Vertical Sediment Variation	VSV
Downdip Sediment Association	DSA
Updip Sediment Association	USA
Interbedded Sediment Association	ISA
Sediment Type	STY
Lithology	LTY
Paleomarker	PMA
Bedding Thickness	BTH
Fauna	FNA
Aerial Geometry	AGM
Sediment Texture	STX
Paleoenvironment Indicator	PEI
Sediment Structure	SST

TABLE II
List of Facies Structures & Geological attributes

represent. When comparing two partitions, the one that produces the greater *average* variance between classes should be the preferred partition since the classes in this partition represent the more distinct concepts.

B. Dataset Descriptions

The Soybean and Iris data sets were obtained from the UCI repository of machine learning databases and domain theories. The database of soybean diseases [33] used is a subset of the original data, consisting of 47 instances from 4 classes, each represented by 35 nominal attributes. Classes C1, C2, and C3 have 10 data objects each, and class C4 has 17 data objects. Classes C1 and C2 are distinct, but classes C3 and C4 are more similar.

The Iris dataset [20] contains 150 object descriptions distributed equally in 3 classes: *setosa*, *virginica*, and *versicolor*. The original data object descriptions are four numeric-valued features: sepal length and width, and petal length and width. The features were converted nominal-valued form using a discretization algorithm discussed in [6]. The characteristic of the data set is that setosa separates well from the other two classes, but versicolor and virginica are mixed.

The mineral data set was created by an expert geologist. It contains 39 different minerals from ten different groups based on chemical composition. Each mineral is described in terms of optical properties, such as color, form, cleavage, relief, birefringence, and interference figure. Given that the features correspond to optical properties, it was not clear whether clustering could recreate the chemical groupings shown in Table I. Therefore, analysis of this dataset illustrates the process of characterizing and interpreting the groupings formed, i.e., Step 3 in the exploratory data analysis task.

The last data set was extracted from PLAYMAKER, a rule-based system for characterizing hydrocarbon plays [3]. A set of 144 rules for classifying one of 13 *facies* structures in geological formations (see Table II) was extracted from the larger PLAYMAKER rule base. Examples of some expert supplied PLAYMAKER rules appear below.

If PMA `subaerial`
STY `sandstone`
LTY `homogeneous`
PBT `dipping-parallel`
DSE `shelf`
then facies `Aeolian (7)`

If PMA `freshwater`
PBS `flat-top-lens`
DSE `shelf`
then facies `Fluvial (4)`
facies `Glacial (1)`
facies `Alluvial-Fan (1)`

If PMA `fresh-water`
DSE `shelf`
then facies `Lacustrian (3)`
facies `Glacial (1)`
facies `Fluvial (3)`

Note that rules have overlapping conditions. This can be explained by the fact that some are more general and others more specific. Rules may have multiple conclusions and these conclusions are weighted by a belief factor shown in parentheses.

Conversion of PLAYMAKER rules to ITERATE data objects is discussed in [7]. All feature values are nominal. An interesting feature of the data set is that no rule uses more than 5 of the 16 features, in fact, on the average, a rule uses only 3.04 attributes, therefore, $75 - 80\%$ of the attribute values are missing. The modification to the ITERATE algorithm to handle large numbers of systematic missing values is described in [40]. The belief values were not included in the definition of ITERATE data objects.

C. Quality of the Final Partitions

In exploratory data analysis, class structure of a data set is not known beforehand. Therefore, objective measures have to be defined to evaluate the partition structure

generated by a clustering algorithm. Previous studies [14], [41] use maximizing the partition score (i.e., average predictability) as the norm for evaluating the "goodness" of a partition. That criterion may not result in both the individual intra-class similarities and pairwise inter-class dissimilarities being maximized. For example, the highest partition score obtained on the Soybean data, 1.468, corresponds to a three class structure – D1, D2, and D3 & D4 merged into one class. The partition score corresponding to the class structure with four classes is 1.389. This is explained by the fact that the partition score maximizes the *average* predictability over a partition as opposed to maximizing predictability for an individual class. On the other hand, category match (equation 4) with the iterative redistribution control structure focuses on global movements of individual objects to enable them to seek their best-match clusters. This results in better intra-class similarity and inter-class dissimilarity than the partition score measure. This was observed in the experiments conducted, where the average cohesiveness values were higher for all three data sets, and that the final partitions were consistently generated in more desirable parts of the partition search space.

Final partition structures generated using the ADO ordering scheme versus using random initial ordering of data are compared to demonstrate the effectiveness of ADO in generating better partitions. Partition quality is measured in terms of the average M_{dk} (cohesion) and average VDM values (distinctness) described in Section 4.

The mean and standard deviations of the average M_{dk} value for the Soybean, Iris, and Mineral data sets over 51 runs each are shown in Table III. Note that we see an increase in the average match value for all three data sets implying more cohesion in the clusters of a partition. Moreover, the smaller standard deviation values indicate consistency in the convergence to better structures.

The t-test was employed to check for statistical significance in the difference in means. The t-statistic is given by:

$$t = \frac{\bar{X} - \mu_0}{\hat{s}_X/\sqrt{N}}.$$

$\bar{X}$ is the sample mean, μ_0 is the hypothesized population mean, and s_X is the unbiased standard deviation of the sample. The computed values appear in Table IV. All values greater than the 2.36 are significant with 99% or greater confidence, and the Soybean and Iris data satisfy this threshold. The mineral data mean may be considered significant at only the 80% confidence level.

The statistical significance of the difference in the standard deviations (actually the variances), was checked using the directional t-test with null hypothesis – $H_0 : {\sigma^2}_{ran} < {\sigma^2}_{ADO}$. Given N observations, the t-statistic with $N-2$ degrees of freedom is:

$$t = \frac{(\hat{s}^2_{ran} - \hat{s}^2_{ADO})\sqrt{N-2}}{2\hat{s}_{ran}\hat{s}_{ADO}\sqrt{1 - {r^2}_{exp\cdot ADO}}},$$

where the $\hat{s}'s$ represent the estimated standard deviations

Method	Soybean		Iris		Mineral	
	mean	s-dev	mean	s-dev	mean	s-dev
ADO	7.575	0.327	1.794	0.012	2.808	0.278
Random	6.460	0.789	1.368	0.218	2.759	0.536

TABLE III
Differences in average match as a measure of Cohesion

Means			
Comparison	Soybean	Iris	Mineral
Random-ADO	24.35	13.96	1.26
Standard Deviations			
Comparison	Soybean	Iris	Mineral
Random-ADO	24.35	64.02	5.192

TABLE IV
t-statistic: Means and Standard Deviations for M_{dk}

for the two sets of runs, and r^2 represents the square of the correlation between the two samples. The t-statistic computed is presented in Table IV. All values above 2.34 reject the null hypothesis with a greater than 99% confidence.

For the soybean data (Table V), the variance of the combined class 3/4 with the classes 1 and 2 (last column of the three class partition) versus the variance of the individual classes 3 and 4 (columns 3 and 4 of the four class partition), one notes that the variance values for the latter are higher than the former. Moreover, the variance between class 3 and 4 distributions is 0.25. Though smaller than the other classes it is non negligible ($> \frac{1}{2}$ of the average variance 0.45). The overall average variance for the four class partition is a little higher than the average variance for the three class partition. This establishes the four class partition as the favored structure.

For the Iris data, we compared the average variance of the distribution match of the given class structure (using the known object labels) with the variance of the distribution match of the partition structure generated by ITERATE (Table VI). The average variance of the ITERATE-generated classes was higher, indicating that they were more distinct than the original class descriptions.

Table VII reports the average variance of the distribution match for the three data sets on the experimental runs. The t-statistic for establishing the statistical significance in the VDM means is shown in Table VIII. All values

Four class partition					**Three class partition**			
	C1	C2	C3	C4		C1	C2	C3/4
C1	0.0	0.462	0.390	0.407	C1	0.0	0.462	0.344
C2		0.0	0.631	0.537	C2		0.0	0.514
C3			0.0	0.245	C3/4			0.0
C4								
Average = 0.45					**Average = 0.44**			

TABLE V
Variance of Distribution Match: Soybean Data

Given partition				ITERATE partition		
	C1	C2	C3	C1	C2	C3
C1	0.0	1.166	1.235	0.0	1.279	1.286
C2		0.0	0.641		0.0	0.578
C3			0.0			0.0
Average = 1.02				**Average = 1.05**		

TABLE VI
Variance of Distribution Match: Iris Data

Method	Soybean	Iris	Mineral
ADO	0.4454	1.11	0.999
Random	0.4412	1.05	0.973

TABLE VII
Average Variance of the Distribution Match

greater than 2.343 are considered significant with 99% or greater confidence. Both the soybean and mineral data sets showed significant increases as expected. The iris data set showed a significant decrease in the mean VDM. This is because random orderings tended to converge to a 2 class structure with VDM = 1.138, as compared to the average VDM of 1.05 for the 3 class structure. However, M_{dk}, which measures the cohesion of individual classes is significantly higher for the 3 class structure, indicating that, in this case, the ADO ordering achieves greater within-class cohesion while trading off inter-class separability. In general, our empirical results demonstrate that ITERATE's pre-ordering scheme, ADO, and redistribution operator do well in improving cohesion within classes and maximizing inter-class dissimilarity.

D. Interpreting Class Structures: The Mineral data

The groupings obtained for the mineral data set are illustrated in Table IX. The numbers in parentheses correspond to the number of objects in each of the groups. Note that the optical features used to characterize the data objects do not produce the chemical groupings (Table I). Instead these features produce a partition that corresponds to the crystal structure of the minerals shown in Table IX. It may seem unusual that gibbsite, which is *monoclinic* forms its own singleton class (Class 3). However, it is known that hydroxides often absorb water and exhibit variations in their structural properties. Though classes 3,4, and 5 have similar (i.e., monoclinic) crystal structures, their significant features have different values (see Table X). A feature-value is considered significant if it has high predictiveness ($P(C_k \mid A_i = V_{ij})$) for the class (k), and the predictability ($P(A_i = V_{ij} \mid C_k)$) of the feature value ($A_i = V_{ij}$) is also high. For example, the feldspar group and the sulfates (Class 4) are mostly colorless though they may exhibit some clouding, whereas the pyroxenes exhibit various shades of color in their thin sections. Gibbsite on the other hand is colorless with a brownish tinge. Other significant features that exhibit differences are cleavage, relief, and birefringence.

Method	Soybean	Iris	Mineral
Random-ADO	33.884	9.96	12.309

TABLE VIII
t-statistic: Significance of differences in VDM

crystal structure	ITERATE
isometric/isotropic (6)	C1(6)
hexagonal (11)	C6(11)
feldspar monoclinic, triclinic, and orthorhombic (14)	C4(14)
pyroxene monoclinic, triclinic, and orthorhombic (5)	C5(5)
tetragonal (2)	C2(2)
singletons (gibbsite)	C3(1)

TABLE IX
Mineral Data: ITERATE Partition Structure

E. PLAYMAKER

Geology, the problem solving domain for the PLAYMAKER system, is characterized by fuzzy domain concepts because geological data often imply multiple characteristics, and therefore, precise domain models for problem solving are hard to come by [3]. As a result, problem solving rules are often acquired on a case-by-case basis and problem solving traces do not show much common structure. This makes it hard to apply induction schemes like chunking (SOAR [27]) and macro-operators [13] on problem solving traces to derive aggregated problem solving steps that can be used to focus the reasoning process. Therefore, we focus on an alternative approach to improve problem solving efficiency by directly exploiting domain structure present in the expert-supplied problem-solving knowledge encoded as rules in the knowledge base. Domain structure refers to inherent regularities and differences that can be observed in the expert-supplied knowledge. Our hypothesis is that this structure should produce better performance in problem-solving tasks by providing a reorganization of the problem solving knowledge in a way that provides access to the most relevant knowledge in specific problem-solving situations. The net result should be a more efficient dialog with the user.

In our experiment, ITERATE was used to create a hierarchical partition structure (see Fig. 2) for the 144 PLAYMAKER rules for classifying 13 different facies structures. The derived class structures for groups of rules are termed *rule models* [10]. A rule model is a generic description for the set of rules that constitute that rule model. These models were used to focus hypothesis refinement during a problem solving session.

The rule model hierarchy produced by running ITERATE recursively on the facies rules is shown in Figure 2. Each node (number in parentheses) defines a rule model, which is defined in terms of its associated attribute values. A more complete description of the rule hierarchy and its

Class	C4			C5			C3		
Feature	Value	$P(A.V \mid C)$	$P(C \mid A.V)$	Value	$P(A.V \mid C)$	$P(C \mid A.V)$	Value	$P(A.V \mid C)$	$P(C \mid A.V)$
color	cl	0.86	0.6	cln,etc.	0.2	1.0	clpbr	1.0	1.0
form	ahsh	0.36	0.71	pr	0.4	1.0	eh	1.0	1.0
cleav	p001	0.43	1.0	pl110	0.8	0.8	pl001	1.0	1.0
relief	lo	0.5	1.0	hi	1.0	0.71	mod	1.0	1.0
biref	wk	0.86	0.63	st	0.4	0.67	mod	1.0	1.0
extinc	plcl	0.21	0.75	pl	0.6	0.33	oblique	1.0	1.0
intfig	bi	0.93	0.72	bi	1.0	0.28	none	1.0	1.0
lown	one	0.5	0.41	thr	0.8	0.57	two	1.0	1.0
opt	pos	0.5	0.44	pos	0.8	0.25	pos	1.0	1.0

TABLE X
Predictability and Predictiveness of Features: Mineral Data

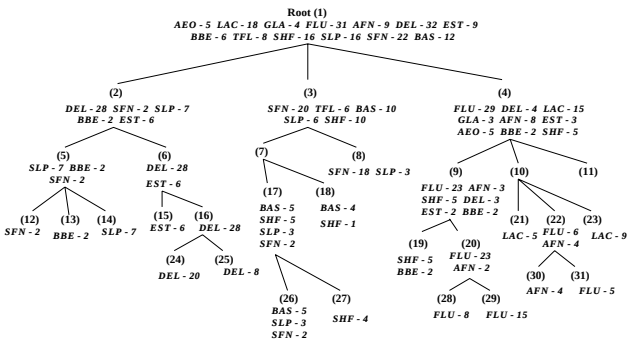


Fig. 2. Rule model for Facies

Case no.	1	2	3	4	5	6	7	8	9	Ave
Query savings	3	2	2	3	3	2	2	2	2	2.33

TABLE XI
QUERY SAVINGS

Case no.	1	2	3	4	5	6	7	8	9	Ave
Without rule models	0	2	3	0	2	4	1	0	0	1.33
With rule models	0	0	1	0	0	1	2	2	0	.67

TABLE XII
NUMBER OF CHANGE-OVERS

generation method appears in [7].

The resulting concept hierarchy is used to focus PLAY-MAKER's evidence gathering scheme during problem solving. Details of the method used for selecting a rule model based on the current hypothesis are presented in [7]. The performance of the system using the ITERATE generated rule models was evaluated on 9 test cases. These test cases were compiled from case books of real data by our geological experts. The results are compared against the previous method of attention focusing [7] which basically chose rules that supported the leading hypotheses. The experiments focused on two issues: (i) how much more efficient is the rule model scheme over the previous scheme, and (ii) are there any structural differences in the consultation dialogues between the two schemes?

Efficiency was measured in terms of the number of queries the system had to ask of the user before it got to its final conclusions. Efficiency improved because the system could identify the more important and relevant features pertaining to the desired conclusions early in the dialog process, and as a result, the unimportant and irrelevant attributes were ignored. This not only reduced the number of queries asked, but is also produced more focused final conclusions in terms of relative ranking by belief values. We determined empirically that a consultation could be terminated if belief values of the relevant hypotheses did not change over the last three queries. The actual number of queries saved for the nine test cases are listed in Table XI.

From an efficiency viewpoint, 2.3 queries/case represents a 14% speedup on the average, which is significant but not a tremendous gain. However, study of the structure of the consultation process revealed another primary difference between the two versions of the system. In the rule model version, sets of evidence (queries) were picked in a particular context defined by the selected rule models. This made it easier for the system to present sets of related queries to the user simultaneously, rather than follow the one-query-at-a-time format. The result was a quicker consultation process, and interactions where the user could focus on pieces of relevant evidence at the same time. During the consultation process, the hypotheses were ranked in terms of their associated belief values. We used a measure, the number of change-overs in the rankings of the primary hypotheses to illustrate the ability of the rule model approach to focus on the right context and conclusions early in the consultation process. Everytime the ranking of a primary (i.e., the top 3 hypotheses) changed during the consultation process, the change-over count was incremented by 1. Table XII lists the results of the experiment. An average 100% performance improvement was observed in the rule model approach.

In addition to providing speedup, the more structured case presentations demonstrate clarity in the reasoning process. This aids the system in providing better explanations of its own reasoning processes. This property is a key to achieving overall reliability and effectiveness in complex decision making systems.

V. Conclusions

The focus of this work has been on developing a concept formation tool useful for data mining tasks. The key issue to be addressed in this unsupervised framework is *interpretation* of the generated partition structure. A partition structure is considered to be more interpretable if it maximizes the traditional concepts of intra-class similarity and inter-class dissimilarity. These were measured by *cohesion* within a class and the *distinctness* between classes, respectively. By performing an analysis of the biases of the concept formation strategy (partition score and incremental control structure), we derived a general data ordering method to exploit the biases in accordance with the goals of our task. A second operator was introduced to manipulate the partition structure and further improve our performance measures.

The result, our conceptual clustering algorithm, ITERATE, uses a hierarchical control structure to determine an effective initial starting point for a partitional optimization scheme, iterative redistribution, for generating interpretable clusters. The ADO algorithm, orders the data to favorably exploit the biases of the category utility function, so that ITERATE generates an initial partition structure with good cohesiveness and separability among the classes. The iterative redistribution operator, as currently implemented, is quite powerful and general, and it can be applied in other situations to optimize initial partition structures.

The experimental results in Section 4 demonstrate that ITERATE achieves statistically significant improvements in within-class cohesion and between-class separability. The mineral data and the PLAYMAKER rule models describe two empirical studies where interpretation of class structure forms the basis for further analysis. The PLAYMAKER results are especially exciting because it demonstrates how the partition structures assist in improving efficiency in problem solving performance. Other extensions to ITERATE that have not been discussed in this paper include modifications to the criteria functions and control structure to handle missing values. Our studies with the PLAYMAKER rule base indicate that systematic missing values play a significant role in the cluster definition process [7], [40].

Further work with the ITERATE algorithm includes extending the capabilities of the algorithm along the following directions:

1. investigate in more detail the relation between clusters and concept definitions associated with these clusters; the goal is to develop formal methodologies for data mining studies,
2. process data sets that have a combination of numeric- and nominal-valued attributes; our initial efforts indicate that this is a difficult task [6], and
3. most important, extend the ITERATE control structure so that we can selectively generate a hierarchy of concepts rather than a flat partition. The need for this is very apparent in some of our current work on analysis of data from real-life data bases.

The overall goal is to use ITERATE as the discovery tool in our toolbox for concept and knowledge discovery in large databases.

Acknowledgments

We acknowledge Gyesung Lee who has performed the PLAYMAKER studies as part of his Ph.D. dissertation research. A number of others: Glenn Koller, Anil Jain, and Cen Li, who have provided valuable input to this research. Tracy Price helped with the final editing. Suggestions for improvement by the anonymous reviewers are also gratefully acknowledged. Ease of access to the UCI repository of machine learning databases and domain theories was also a big help.

References

[1] J.R. Anderson The Adaptive Character of Thought, Lawrence Erlbaum Assoc., Hillsdale, NJ, 1990.

[2] G.H. Ball. Data Analysis in the Social Sciences: What about the details?. *Proc. of the AFIPS Fall Joint Computer Conference*, pp. 533-560, 1965.

[3] G. Biswas, et al. PLAYMAKER: A Knowledge-Based Approach to Characterizing Hydrocarbon Plays. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 4:315-339, 1990.

[4] G. Biswas, J. Weinberg, Q. Yang, and G. Koller. Conceptual Clustering and Exploratory Data Analysis. *Proc. of the Eighth Intl. Workshop on Machine Learning*, Evanston, IL, pp. 591-595, June 1991.

[5] G. Biswas, J. Weinberg, and G. Koller. Data Exploration in Non Numeric Databases. *Advances in Database and Artificial Intelligence*, vol. 1, F. Petry and L. Delcambre, eds., Jai Press, CT, pp. 145-165, 1995.

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

[7] G. Biswas and G. Lee. Knowledge Reorganization: A Rule Model Scheme for Efficient Reasoning, *Proc. Tenth IEEE Conference on AI for Applications (CAIA),* San Antonio, TX, pp. 312-318, March 1994.

[8] W. Buntine. Myths and Legends in Learning Classification Rules. *Proc. AAAI-90*, Boston, MA, 736-742, 1990.

[9] P. Cheesman, et al. AutoClass: A Bayesian Classification System. *Proc. of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, 54-64, 1988.

[10] R. Davis. Use of Meta-level Knowledge in Construction, Maintenance, and use of Large Knowledge Bases. Ph.D. Dissertation, Stanford University, 1976.

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

[12] U. Fayyad, G. Piatetsky, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine,* 17:37-4, 1996.

[13] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and Executing Generalized Robot Plans. *Artificial Intelligence,* 3:251-288, 1972.

[14] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning,* 2:139-172, 1987.

[15] D. Fisher. A computational account of basic level and typicality effects. *Proceedings of the Seventh AAAI,* 233-238, Morgan Kaufmann, San Mateo, CA, 1988.

[16] D. Fisher. Noise-Tolerant conceptual clustering. *Proceedings of the Eleventh IJCAI,* 825-830, Morgan Kaufmann, San Mateo, CA, 1989.

[17] D. Fisher and P. Langley. The Structure and Formation of Natural Categories. In: *The Psychology of Learning and Motivation,* vol. 26, pp. 241-284, 1990.

[18] D. Fisher, L. Xu, and N. Zard. Ordering Effects in Clustering. *Proceedings of the Ninth International Conference on Machine Learning,* pp. 163-168, Morgan Kaufmann, San Mateo, CA, 1992.

[19] D. Fisher, et al. Applying AI Clustering to Engineering Tasks. *IEEE Expert,* vol. 8, pp. 51-60, 1993.

[20] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 3:179-188, 1936.

[21] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus. Knowledge Discovery in Databases: An Overview. *AI Magazine,* vol. 14, no. 3, pp. 57-70, Fall 1992.

[22] J.H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence,* vol. 40, pp. 11-61, 1989.

[23] M. Gluck and J. Corter. Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conf. of the Cognitive Science Society,* 283-287, Irvine, CA, 1985.

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

[25] S.J. Hanson and M. Bauer. Conceptual clustering, categorization, and polymorphy. *Machine Learning,* 3:343-372, 1989.

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

[27] J.E. Laird, P.S. Rosenbloom, and A. Newell. Chunking in SOAR: The anatomy of a general learning algorithm. *Machine Learning,* 1:11-46, 1986.

[28] P. Langley, H.A. Simon, G.L. Bradshaw, and J.M. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Processes,* MIT Press, Cambridge, MA, 1987.

[29] M. Lebowitz. Experiments with incremental concept formation: UNIMEM. *Machine Learning,* 2:103-138, 1987.

[30] C. Li and G. Biswas, "Conceptual Clustering with Numeric-and-Nominal Mixed Data - A New Similarity Based System," in review, *IEEE Trans. on Knowledge and Data Engineering,* July 1996.

[31] K.B. McKusick and P. Langley. Constraints on tree structure in concept formation. *Proc. 12th Intl. Joint Conf. on Artificial Intelligence*, Sydney, Australia, pp.810-816, August, 1991.

[32] R. Michalski and R.E. Stepp. Learning from observation: conceptual clustering. In: *Machine Learning: An Artificial Intelligence Approach,* R. Michalski, J. Carbonell, and T. Mitchell, eds., 331-364, Tioga Press, Palo Alto, CA, 1983.

[33] R. Michalski and R.E. Stepp. Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5:396-409, 1983.

[34] T.M. Mitchell. The Need for Biases in Learning Generalizations. *Rutgers Technical Report*, 1980. Reprinted in *Readings in Machine Learning,* J. Shavlik and T. Dietterich, eds., 184-191, Morgan Kaufmann, San Mateo, CA, 1990.

[35] J.R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81-106, 1986.

[36] A. Silberschatz and A. Tuzhilin. On Subjective Measures of Interestingness in Knowledge Discovery. *Proc. of KDD-95: First International Conference on Knowledge Discovery and Data Mining,* AAAI Press, Menlo Park, CA, 275-281, 1995.

[37] H.A. Simon. *The Sciences of the Artificial.* MIT Press, Cambridge, MA, 1981.

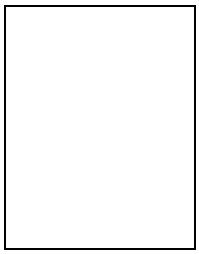
[38] P.E. Utgoff. Machine Learning of Inductive Bias. Kluwer Academic Publishers, 1986.

[39] J.B. Weinberg and G. Biswas. An Analysis of the Inductive Bias of Concept Formation Using the Category Utility as a Criterion Function. Tech. Report, Dept. of Computer Science, Vanderbilt Univ., Nashville, TN.

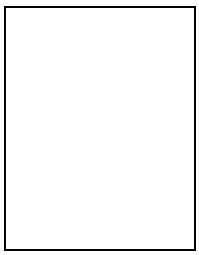
[40] J.B. Weinberg, G. Biswas, and G.R. Koller. Conceptual Clustering with Systematic Missing Values. *Proc. of the Ninth International Workshop on Machine Learning*, Aberdeen, Scotland, pp. 464-469, 1992.

[41] L. Xu. Improving Robustness of the COBWEB Clustering System. M.S. Thesis, Department of Computer Science, Vanderbilt University, Nashville, TN, December, 1991.

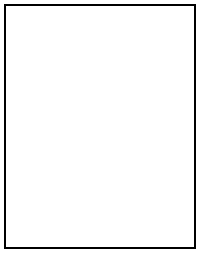
[42] J.M. Zytkow. The KDD land of plenty. *AAAI Workshop Notes – Knowledge Discovery in Databases*, Anaheim, CA, pp. iii-vi, July 14, 1991.



Gautam Biswas is an Associate Professor of Computer Science, Computer Engineering and Management of Technology, and Director of the Computer Engineering program at Vanderbilt University. He has a B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Bombay, India, and M.S. and Ph.D. degrees in Computer Science from Michigan State University, East Lansing. Dr. Biswas conducts research in the design and analysis of Intelligent Systems, and has primary interests in qualitative and model-based reasoning, hybrid modeling systems, monitoring and diagnosis, conceptual clustering and data mining, and cognitive psychology and intelligent learning environments. He has published in a number of journals and contributed book chapters. He is an associate editor of the International Journal of Approximate Reasoning and the Journal of Applied Intelligence, and has served on the Program Committee of a number of conferences. He was co-chair of the 1996 Principles of Diagnosis Workshop, and a senior program committee member for AAAI 97 and AAAI 98. He is a Senior member of the IEEE Computer Society, ACM, AAAI, and the Sigma Xi Research Society.



Jerry Weinberg is an assistant professor of computer science as Southern Illinois University at Edwardsville. His research involves abductive reasoning, inductive learning, medical diagnosis, and the application of conceptual clustering and concept formation to database mining and knowledge discovery. He received a BS in nursing from Indiana State University, a BS in computer science from the University of South Carolina, and his MS and PhD in computer science from Vanderbilt University. He is a member of IEEE Computer Society, AAAI, and ACM.



Doug Fisher is an associate professor of computer science at Vanderbilt University. His current research areas are machine learning, knowledge discovery, and cognitive models of problem solving. Application areas include Space Shuttle telemetry data, printing process control, road traffic control, and software characterization. He received his PhD from the University of California at Irvine in 1987 for the development of the Cobweb clustering system. He is an associate editor of IEEE Expert and of Machine Learning, a governing board member of the Society for Artificial Intelligence and Statistics, and a member of the AAAI, IEEE, and ACM. Doug's email address is dfisher@vuse.vanderbilt.edu.