LEARNING FROM OBSERVATION: CONCEPTUAL CLUSTERING

by

R. S. Michalski
R. Stepp

Chapter In The Book, Machine Learning: An Artificial Intelligence Approach, R. S. Michalski, J. Carbonell And T. Mitchell (Eds.), Tioga Publishing Co., Pp 331-363 Palo Alto, 1983.

ISG 1983-1

Chapter in the book, MACHINE LEARNING: An Artificial Intelligence Approach, R. S. Michalski, T.Mitchel and J. Carbonell (Eds.), TIOGA Pub. Co., Palo Alto, 1983, pp.331-363.

11

LEARNING FROM OBSERVATION: CONCEPTUAL CLUSTERING

Ryszard S. Michalski
Robert E. Stepp
University of Illinois at Urbana-Champaign

ABSTRACT

An important form of "learning from observation" is constructing a classification of given objects or situations. Traditional techniques for this purpose, developed in cluster analysis and numerical taxonomy, are often inadequate because they arrange objects into classes solely on the basis of a numerical measure of object similarity. Such a measure is a function only of compared objects and does not take into consideration any global properties or concepts characterizing object classes. Consequently, the obtained classes may have no simple conceptual description and may be difficult to interpret.

The above limitation is overcome by an approach called *conceptual clustering*, in which a configuration of objects forms a class only if it is describable by a concept from a predefined concept class. This chapter gives a tutorial overview of *conjunctive* conceptual clustering, in which the predefined concept class consists of conjunctive statements involving relations on selected object attributes. The presented method arranges objects into a hierarchy of classes closely circumscribed by such conjunctive descriptions. Descriptions stemming from each node are logically disjoint, satisfy given background knowledge, and optimize a certain global criterion.

The method is illustrated by an example in which the conjunctive conceptual clustering program CLUSTER/2 constructed a classification hierarchy of a large collection of Spanish folk songs. The conclusion suggests some extensions of the method and topics for further research.

11.1 INTRODUCTION

An omnipresent problem in science is to construct meaningful classifications of observed objects or situations. Such classifications facilitate human comprehension of the observations and the subsequent development of a scientific theory. This problem is a form of the very general, well-known principle of "divide and conquer" used in a variety of problem-solving situations. It is also related to the problem of decomposing any large-scale engineering system (for example, an AI system) into smaller components, in order to simplify its design and implementation.

The nature of processes leading to useful classifications remains little understood, despite considerable effort in this direction. From the viewpoint of machine learning, the process of constructing classifications is a form of "learning from observation" ("learning without a teacher"). This form of machine learning has been systematically studied in such areas as cluster analysis and numerical taxonomy. The central notion used there for creating classes of objects is a numerical measure of *similarity* of objects. Classes (clusters) are collections of objects whose intra-class similarity is high and inter-class similarity is low.

A measure of similarity is usually defined as a proximity measure in a multi-dimensional space spanned by selected object attributes. Such a measure is, therefore, meaningful only if the selected attributes are relevant for describing perceived object similarity. The presence of irrelevant attributes will distort this measure. Moreover, all attributes defining the description space are given equal weight in the process of determining classes. The problem, then, becomes one of structuring attributes into classes, in order to determine the most relevant attributes. Factor analysis and multi-dimensional scaling have been used for this purpose, but these methods were designed primarily for numerical variables. They cannot adequately handle the many-valued, nominal (categorical) variables which occur often in human classifications.

The use of numerical measures of similarity for constructing classifications has other disadvantages. Such measures take into consideration only the properties of compared objects without regard to any context or concepts useful for characterizing object configurations. Consequently, the resulting classes do not necessarily have any simple conceptual description and may be difficult to interpret. The problem of determining the *meaning* of the obtained classes is simply left to the researcher. This is a significant disadvantage of traditional methods because a researcher analyzing data typically wants to create classes that are not only mathematically well defined, but that also have a meaningful conceptual interpretation.

This chapter describes an approach to the problem of automatic construction of classifications, in which a configuration of objects forms a class only if it can be closely circumscribed by a conjunctive concept involving relations on selected object attributes. The problem undertaken can be defined as follows:

Given:

- A set of objects (physical or abstract),
- A set of attributes to be used to characterize the objects,
- A body of background knowledge, which includes the problem constraints, properties of attributes, and a criterion for evaluating the quality of constructed classifications.

Find:

- A hierarchy of object classes, in which each class is described by a single conjunctive concept. Subclasses that are descendants of any parent class should have logically disjoint descriptions and optimize an assumed criterion (a *clustering quality criterion*).

Structuring objects into such *conjunctive hierarchies* is called *conjunctive conceptual clustering*. It is a special case of *conceptual clustering* in general, which we define as a process of constructing a *concept network* characterizing a collection of objects, with nodes marked by *concepts* describing object classes, and links marked by the relationships between the classes.

The idea of conceptual clustering and a general method for determining conjunctive hierarchies was introduced by Michalski [1980a]. This chapter is a tutorial overview of conjunctive conceptual clustering and the algorithm implemented in the program CLUSTER/2 (a successor to the earlier program CLUSTER/PAF [Michalski & Stepp, 1981]). The algorithm is illustrated by its application to a practical problem in the area of musicology. The conclusion discusses some possible extensions of the method and suggests topics for future research. To improve the readability of this chapter, Table 11-1 provides a list of basic symbols and operators together with a short explanation.

11.2 CONCEPTUAL COHESIVENESS

In conventional data analysis, the similarity between any two objects is characterized by a single number: the value of a similarity function applied to symbolic descriptions of objects. These symbolic descriptions are vectors, whose components are scores on selected object attributes.

Such measures of similarity are *context-free*, that is, the similarity between any two objects A and B depends solely on the properties of the objects, and is not influenced by any context (the "environment" surrounding the objects). Some authors have introduced *context-sensitive* measures of similarity, where the similarity between objects A and B depends not only on

Table 11-1: A Table of Basic Symbols and Operators

&	conjunction (logical product)
$\vee$	disjunction (logical sum)
e_i	an event (a description of an object)
LEF	a lexicographical evaluation functional
DOM(p)	the domain of variable p
$\delta(e_1,e_2)$	the syntactic distance between events e_1 and e_2
α	a complex
λ-complex	a logical complex
s-complex	a set complex
E	the event space
$s(\alpha)$	number of unobserved events in complex α
$p(\alpha)$	number of observed events in complex α
$t(\alpha)$	total number of events in complex α
E	an event set
k	the number of clusters
$RU(e_1...,\alpha_1...)$	the refunion operator
$GEN(\alpha)$	a generalization of complex α
$COV(E_1 \mid E_2)$	a cover of event set E_1 against E_2
$G(e \mid E_o)$	a star of event e against event set E_o
$RG(e \mid E_o)$	a reduced star of event e against event set E_o
$RG(e \mid E_o,m)$	a bounded reduced star with the bound m

A and B, but also on other objects in the collection to be clustered. One such similarity measure is the reciprocal of *mutual distance* [Gowda & Krishna, 1978]. To determine the mutual distance from object A to object B, objects in the collection are ranked according to the Euclidean distance to A (the closest object gets rank 1) and then according to the Euclidean distance to B. The mutual distance from object A to object B is the sum of the rank of A with respect to B, and the rank of B with respect to A. Thus the similarity between compared objects depends on their relation to other objects.

Taking neighboring objects into consideration solves some clustering problems, but in general is not sufficient. The difficulty lies in the fact that both of the above types of similarity measures are *concept-free*, that is, depend only on the properties of individual objects and not on any external concepts which might be useful to characterize object configurations. Consequently, methods that use such measures are fundamentally unable to capture the "Gestalt properties" of object clusters, that is, properties that characterize a cluster as a whole and are not derivable from properties of individual entities. In order to detect such properties, the system must be equipped with the ability to recognize configurations of objects that correspond to certain "concepts." To illustrate this point, let us consider the problem of clustering the points in Figure 11-1.

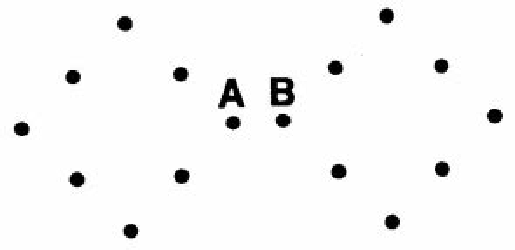


Figure 11-1: An illustration of conceptual clustering.

A person considering Figure 11-1 would typically describe the observed points as "arranged in two diamonds". Thus, the points A and B, although closer to each other than to other points, are placed in separate clusters. Here, human solution involves partitioning the points into groups not on the basis of pairwise distance, but on the basis of *concept membership.* Points are placed in the same cluster if collectively they represent the same concept. In our example, the concept is "diamond".

This idea is the basis of conceptual clustering. From the viewpoint of conceptual clustering, the "similarity" between two points A and B, which we shall call the *conceptual cohesiveness* of A and B, depends not only on those points and surrounding points E, but also on a set of concepts C which are available for describing A and B together:

$$\text{Conceptual cohesiveness}(A,B) = f(A,B,E,C)$$

To illustrate this measure, let us assume that the set of concepts C consists of geometrical figures, such as sequences of straight lines, circles, rectangles, triangles, etc. A measure of conceptual cohesiveness could be defined, for example,[1] as:

$$f(A,B,E,C) = \max_i \left\{ \frac{\#e(i)-1}{\text{area}(i)} \right\}$$

[1]This measure is mentioned solely to illustrate the difference between traditional similarity and conceptual cohesiveness. It is not used in the method of conceptual clustering described here.

where i indexes all geometrical figures that are specified in C and that cover points A and B,

$\#c(i)$ is the total number of data points from E covered by figure i, and

area(i) is the area of figure i.

Note that the constant "-1" in the numerator assures that the conceptual cohesiveness reduces to a conventional similarity measure (a reciprocal of distance) when no context points in E are taken into consideration and C is a straight line of unit thickness linking the data points.

11.3 TERMINOLOGY AND BASIC OPERATIONS OF THE ALGORITHM

This section gives a brief overview of the terminology needed to describe the conjunctive conceptual clustering method. This terminology was introduced by Michalski [1980a].

11.3.1 Variables and Their Types

Let $x_1, x_2, \ldots, x_n$ denote discrete variables that are selected to describe objects in the population to be analyzed. For each variable a domain is defined, containing all possible values the variable can take. We shall assume that the domains of variables x_i, $i = 1,2,\ldots,n$ are finite, and therefore can be represented as:

$\mathrm{DOM}(x_i) = \{0,1,\ldots,d_i\text{-}1\}$, $i = 1,2,\ldots,n$

In general, the domains may differ not only with respect to their size, but also with respect to the structure relating their elements. In the case of numerical variables, this structure is defined by the scale of measurement. We distinguish among *nominal* (*categorical*), *linear* (*quantitative*), and *structured* variables, whose domains are unordered, totally-ordered, and graph-ordered sets, respectively. Structured variables represent generalization hierarchies of related values. We distinguish between two types of generalization hierarchies for structured variables:

1. Unordered – when the leaf values in the hierarchy constitute an unordered set.
2. Ordered – when the leaf values in the hierarchy constitute an ordered set.

Figures 11-2 and 11-3 present an example of an unordered and an ordered generalization hierarchy, respectively. In Figure 11-2, the leaves represent specific shapes, and the internal nodes ("polygon", "ellipsoid", "4-sided") represent generalizations or linguistic equivalents of these shapes. In Figure 11-3, the leaves represent specific quantities, and the internal nodes represent ordered generalizations or linguistic equivalents of these quantities.

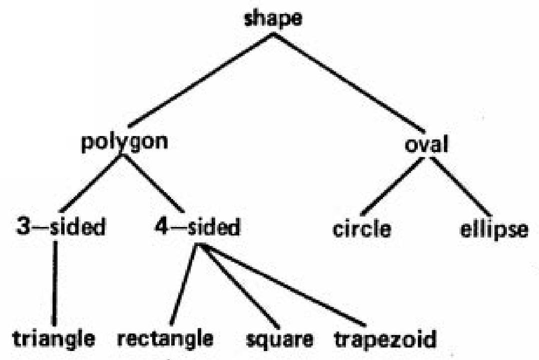


Figure 11-2: An example of an unordered generalization structure.

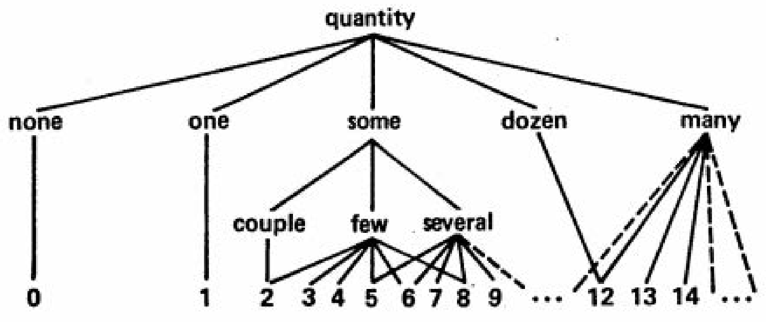


Figure 11-3: An example of an ordered generalization structure.

11.3.2 Event Space

An *event* is an object description in the form of a vector of values of the assumed variables $x_1, x_2,...,x_n$. The *event space* is the space of all possible such events.

11.3.3 Syntactic Distance

The *syntactic distance* $\delta(e_1, e_2)$ between two events e_1 and e_2 is defined as the sum of the syntactic distances between the values of each variable in the events e_1 and e_2. As described by Michalski and Larson [1978], the syntactic distance between two variable values is a number from 0 to 1, determined by a measure which reflects the domain type of the variable. For a nominal variable, the syntactic distance is either 0, if the values taken by the variable in each event are identical, or 1, if the values are not identical. For a linear variable, the syntactic distance is the ratio of the absolute difference between the values to the total span of the domain of the variable. For a structured variable, the evaluation of syntactic distance depends on the type of generalization hierarchy. Since structured variable values in events are leaves of a generalization hierarchy, the syntactic distance between such values for unordered and ordered hierarchies is evaluated the same way as for nominal and linear variables, respectively.

11.3.4 Relational Statements

A *relational statement*[2] (or a *selector*) is a form:

$$[x_i \# R_i]$$

where R_i, the *reference*, is a list of elements from the domain of variable x_i, linked by the *internal disjunction*, denoted by " $\vee$ ".

# stands for the relational operator "$=$" or "$\neq$".

The selector $[x_i = R_i]$ ($[x_i \neq R_i]$) is interpreted as "value of x_i is one of the elements of R_i" ("value of x_i is not an element of R_i"). In the case of linear variables, the notation of a selector can be simplified by using relational operators $\geq$, $>$, $<$, $\leq$, and a range operator "..", as illustrated below. Here are a few examples of a selector, in which variables and their values are represented by linguistic terms:

[2] This form is a special case of a referential selector defined in the annotated predicate calculus (Chapter 4 of this book). This form was first introduced in the variable valued logic system one (VL_1), described by Michalski [1975a].

[length > 2]	(length is greater than 2)
[color = blue $\vee$ red]	(color is blue or red)
[size ≠ medium]	(size is not medium)
[weight = 2..5]	(weight is between 2 and 5, inclusively)

11.3.5 Complexes

A logical product of selectors is called a *logical complex* (λ-complex) $\&_{i \in I}[x_i \# R_i]$, where $I \subseteq \{1,2,...,n\}$. An event e is said to satisfy an λ-complex if values of variables in e satisfy all the selectors in the complex.

For example, event e = (2, 7, 0, 1, 5, 4, 6) satisfies λ-complex $[x_1 = 2 \vee 3][x_3 \leq 3][x_5 = 3..8]$ (concatenation of selectors denotes conjunction). An λ-complex can be viewed as an exact symbolic representation of the events which satisfy it. For example, the above λ-complex is the symbolic representation of all events for which x_1 is 2 or 3, x_3 is smaller than or equal to 3, and x_5 is between 3 and 8.

A collection of events for which there exists an λ-complex satisfied by these events and only by these events is called a *set complex* (s-complex). If the distinction between λ- and s- complexes is not important, then we shall use simply the term *complex.*

11.3.6 Sparseness

Let E be an event space, and $E \subseteq E$ be a set of events representing objects to be clustered. The events in E are called observed events, and events in $E \setminus E$ are called unobserved events. Let α be a complex which covers (includes) some observed events and some unobserved events. The number of observed events (points) in α is denoted by $p(\alpha)$. The number of unobserved events in α is called the *absolute sparseness* of α *in* E and denoted by $s(\alpha)$. The total number of events contained in α is thus $t(\alpha) = p(\alpha) + s(\alpha)$. The *relative sparseness of a complex* is denoted by $r(\alpha)$ and is defined as the ratio of the absolute sparseness of the complex to the total number of events covered by the complex, in other words:

$$r(\alpha) = 1 - \frac{p(\alpha)}{t(\alpha)} \qquad r(\alpha) = 1 - \frac{p(\alpha)}{t(\alpha)}$$

An λ-complex is a generalized description of the observed events contained in the corresponding s-complex. The relative sparseness of a complex can be used as a very simple measure of the degree to which the λ-complex generalizes over (or *fits*) the observed events. If the sparseness is zero, then the description covers only observed events (has zero degree of generalization). As the relative sparseness of the complex increases, so does the degree to which it generalizes over the observed events. The maximum relative sparseness value of 1 is achieved when the complex covers only unobserved events.

The clustering algorithm presented in Section 11.5.1 generates a collection of complexes that are pairwise disjoint. Such a collection, called a *disjoint* *clustering*, describes a partition of all observed events into disjoint classes. The fit between a disjoint clustering and the observed events can be measured by the *relative sparseness* of the clustering, defined as the average of the relative sparsenesses of the complexes in the clustering. Since the complexes in a clustering are disjoint and the total number of observed events is constant, the ranking of clusterings will not change if the relative sparseness measure is replaced by the *absolute* sparseness measure (the sum of absolute sparsenesses of complexes). The latter measure is much simpler computationally and, therefore, is used in the presented clustering algorithm. Henceforth, we shall simply use the term *sparseness* to denote this measure of fit.

An advantage of sparseness as a measure of fit is its simplicity. A disadvantage, however, is that it takes into consideration the whole event space, no matter which variables spanning the space are actually present in the λ-complexes. Therefore, another measure is introduced, called *projected sparseness*, which evaluates a clustering in a subspace of the original event space, defined by specially selected "relevant" variables. To define this measure, let us observe that complexes of a disjoint clustering may involve different subsets of variables. Because complexes are pairwise disjoint, any pair of complexes must contain at least one common variable with disjoint references in both complexes. A variable with this property for any pair of complexes in a clustering is called a *discriminant variable* of the clustering. For example, x_1, x_3, and x_4 are discriminant variables of the clustering:

$$\{[x_1 \geq 3]\,[x_2 = 1 \vee 2]\,[x_3 = 1],\ [x_1 < 3]\,[x_3 = 2 \vee 3]\,[x_4 = 3],\ [x_2 = 1]\,[x_4 \leq 2]\}.$$

The event space spanned over only the discriminant variables is called the *projected event space* of the clustering. The projected sparseness of a clustering is the sum of the absolute sparsenesses of complexes in the projected event space.

11.3.7 Refunion Operator

The *refunion* operator RU transforms a set of events and/or complexes into a single complex covering the events and/or complexes. For each variable, the set of all values the variable takes, in all given events and complexes, is determined. These sets are used as the reference of the variable in the generated complex. For example, given:

$$e_1 = (2,3,0,1)$$
$$e_2 = (0,2,1,1) \text{ and}$$
$$\alpha = [x_1 = 2..3]\,[x_2 = 4]\,[x_3 = 0]\,[x_4 = 2]$$

the refunion complex, $RU(e_1, e_2, \alpha)$, denoted α', is:

$$\alpha' = [x_1 = 0 \vee 2 \vee 3][x_2 = 2 \vee 3 \vee 4][x_3 = 0 \vee 1][x_4 = 1 \vee 2]$$

It can be shown that the refunion complex has the minimum sparseness (absolute or relative) among all complexes covering the given events and/or complexes [Michalski, 1980a].

11.3.8 GEN Operator

The generalizing operator GEN simplifies and generalizes any given complex by applying an appropriate generalization rule (see Section 4.5 in Chapter 4 of this book) to each selector in the complex:

1. To linear selectors, the "closing the interval" rule is applied: The reference is clustered into one or a few disjoint intervals, such that the ratio of the number of unobserved values to the width of the enclosing interval is at or below a certain *sparseness threshold.* For example, the reference $1 \vee 2 \vee 3 \vee 7 \vee 8$ is turned into one interval 1..8, if the assumed threshold is 3/8 or more. If the threshold is less than 3/8, the reference is turned into two intervals $1..3 \vee 7..8$.
2. To structured selectors, the "climbing the generalization hierarchy" rule is applied: A reference with more than one value is replaced by the most specific node in the generalization hierarchy which "covers" the reference.
3. After steps (1) and (2) are completed, the "dropping the condition" rule is applied to all selectors: A selector is removed if the ratio of the number of missing reference values to the number of values in the domain of the variable is below a certain sparseness threshold.

To illustrate the GEN operator, consider the complex α', given above, and assume that variables x_1 and x_2 are linear, variable x_3 is structured, and variable x_4 is nominal, that the domain of x_3 is a generalization hierarchy in which the value "small" is the parent node of values 0 and 1, and that the domain of x_4 contains values 0, 1, 2. Assume also that the sparseness threshold for all variables is 0.5. Then we have:

GEN(α'): $[x_1 \leq 3][x_2 = 2..4][x_3 = \text{small}]$

where the references for x_1 and x_2 are generalized by closing the interval, the reference for x_3 is generalized by climbing the generalization tree, and the selector for variable x_4 is removed by dropping the condition.

11.3.9 Cover

Let E_1 and E_2 be two disjoint event sets, that is, $E_1 \cap E_2 = \phi$. A cover $COV(E_1|E_2)$ of E_1 against E_2 is any set of s-complexes $\{\alpha_j\}_{j\in J}$ such that for each event $e \in E_1$ there is an s-complex α_j, $j \in J$, covering it, and none of the complexes α_j cover any event in E_2:

$$E_1 \subseteq \bigcup_{j\in J} \alpha_j \subseteq E \setminus E_2$$

By representing complexes of the cover as λ-complexes, a cover can be expressed as a logical disjunction of these complexes.

A cover in which all s-complexes are pairwise disjoint is called a *disjoint cover*. If E_1 is a collection to be clustered and $E_2 = \phi$, then a disjoint cover $COV(E_1|\phi)$, or simply $COV(E_1)$, represents a disjoint clustering of events. The algorithm described in Section 11.5 generates a disjoint clustering of events by repetitively constructing a special type of cover, called a *star.*

11.3.10 Star

The *star* $G(e|E_o)$ of event e against event set E_o ($e \notin E_o$) is the set of all maximally general[3] complexes covering the event e and not covering any event in E_o. Informally, it is the set of all maximally general descriptions of event e which do not intersect with set E_o. Figure 11-4 presents a star of event e against events denoted by "•" in the two dimensional space spanned over linear variables. The star consists of complexes α_1, α_2, and α_3. Complex α'_3 is a "reduced" complex α_3, as explained below.

In the algorithm described in the next section, the "theoretical" stars (defined above) are subjected to two major modifications. The first is to minimize the sparseness of complexes in the stars, and the second is to "bound" the stars, that is, to select from them a certain number of "best" complexes, according to a context-dependent criterion. The first modification is performed by procedure Redustar, described below, and the second by procedure Boundstar described in Section 11.5.1.2.

11.3.11 Redustar Procedure

Complexes in stars $G(e|E_o)$ are maximally general, and therefore may describe objects in an overgeneralized way. The Redustar procedure generates a star, and then maximally reduces the sparseness of each complex in it, while preserving the coverage of observed events. For example, complex α'_3 in Figure 11-4 is such a reduced complex obtained from complex α_3. The steps of the procedure are:

---

[3]A complex α is maximally general with respect to a property P if there does not exist a complex α^* with property P such that $\alpha \subset \alpha^*$.

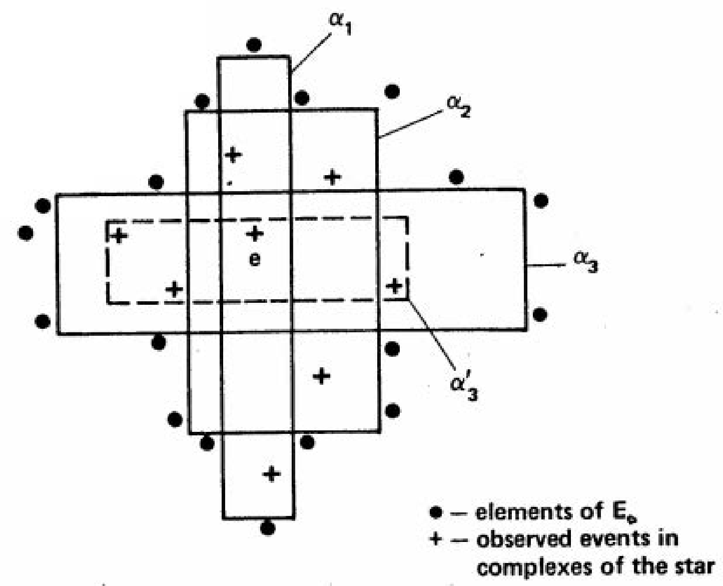


Figure 11-4: An illustration of the star $G(e|E_o)$.

1. Elementary stars, $G(e|e_i)$, $e_i \in E_o$, are determined.

   To generate an elementary star $G(e|e_i)$ of an event e against another event e_i, all variables that have different values in e than in e_i are identified. Suppose, with no loss of generality, that these variables are $x_1, x_2, \ldots, x_g$, and that $e_i = (r_1, r_2, \ldots, r_g, \ldots, r_n)$. The complexes of the star $G(e|e_i)$ are then $[x_j \neq r_j]$, $j = 1, 2, \ldots, g$, because these are the maximally general complexes which cover e and do not cover e_i. The number of complexes in an elementary star is at most n, and, because $e_i \neq e$, at least 1.

2. The complete star $G(e|E_o)$ is determined.

   The star $G(e|E_o)$ is generated by first setting up the logical product & $G'(e|e_i)$, $e_i \in E_o$, where $G'(e|e_i)$ is the disjunction of complexes from the elementary star $G(e|e_i)$. Next, the multiplication of complexes is performed, using absorption laws, until a disjunction of nonredundant complexes is obtained. This multiplication is carried out in steps, each step being a multiplication of a disjunction of complexes by a disjunction of selectors (the elements of consecutive elementary stars). The set of the complexes in the resulting disjunction is $G(e|E_o)$.

3. Complexes in G(e|E_0) are reduced and simplified.

   The sparseness of each complex in the star is reduced as much as possible without "uncovering" any of the observed events. This is done by performing the refunion of all the observed events contained in each complex. The complexes are then generalized and simplified by applying the GEN operator. The resulting set of complexes is a reduced star RG(e|E_0).

The theoretical basis for the above algorithm generating the star G(e|E_0) is described in Michalski [1975b].

11.3.12 NID Procedure

This procedure transforms a set of Nondisjoint complexes Into a set of Disjoint complexes (that is, a disjoint clustering). If input complexes to NID are already disjoint, the procedure leaves them unchanged. The steps of the procedure are:

1. "Core" complexes are determined.

   Observed events covered by more than one complex from the given set are placed on the *multiply-covered event list* (m-list). If the m-list is empty, then the complexes are only weakly intersecting, that is, the intersection area contains only unobserved events. In this case, the procedure terminates with an indication that the combination of complexes is a *weakly intersecting clustering*. Otherwise, each complex is replaced by the Refunion of the observed events contained in the complex that are not on the m-list (i.e., that are singly covered). The obtained Refunions are called "core" complexes.

2. A best "host" complex is determined for each event on the m-list.

   An event is selected from the m-list and is "added" to each of the k core complexes by generalizing each complex to the extent necessary to cover the event. Such a generalization is performed by applying the Refunion operator to the event and the complex. As a result, k modified complexes are obtained. By replacing one of the core complexes in the initial set with the corresponding modified complex, in k different ways, a collection of clusterings is obtained. These clusterings are evaluated according to the assumed clustering quality criterion (see the next section). The best clustering is determined, and the complex in it that covers the given event from the m-list is considered to be the best "host" for this event. The best clustering is retained and the remaining ones are eliminated. By repeating the above operation for every event on the m-list, a set of k disjoint complexes is obtained whose union covers the same observed events as the original set of nondisjoint complexes.

   If an event cannot be "added" to any complex without causing the result to intersect other complexes, then the event is placed on the *exceptions list*.

11.4 A CRITERION OF CLUSTERING QUALITY

The problem of how to judge the quality of a clustering is difficult, and there seems to be no universal answer to it. One can, however, indicate two major criteria. The first is that descriptions formulated for clusters (classes) should be "simple", so that it is easy to assign objects to classes and to differentiate between the classes. This criterion alone, however, could lead to trivial and arbitrary classifications. The second criterion is that class descriptions should "fit well" the actual data. To achieve a very precise "fit", however, a description may have to be complex. Consequently, the demands for simplicity and good fit are conflicting, and the solution is to find a balance between the two.

A number of other measures can be introduced for evaluating clustering quality. CLUSTER/2 uses a combined measure which can include any of the following elementary criteria:

- the fit between the clustering and the events
- the simplicity of cluster descriptions
- the inter-cluster difference
- the discrimination index
- the dimensionality reduction

The fit between a clustering and the data is computed in two different ways, denoted as T and P. The T measure is the negative of the total sparseness of the clustering, and the P measure is the negative of the sum of the projected sparsenesses of the complexes. The reason for using the negative values is to increase the degree of match as the sparseness decreases.

Simplicity of cluster descriptions is defined as the negative of the complexity, which is the total number of selectors in all descriptions.

Inter-cluster difference is measured by the sum of the degrees of disjointness between every pair of complexes in the clustering. The degree of disjointness of a pair of complexes is the number of selectors in both complexes after removing selectors that intersect. For example, the pair of complexes:

- [color = red] [*size = small or medium*] [shape = circle]
- [color = blue] [*size = medium or large*]

has the degree of disjointness 3, because 2 of the 5 selectors intersect (intersecting selectors are italicized). This criterion promotes clusterings with classes having many differing properties, and is analogous to the criterion of requiring maximal distance between clusters, used in conventional methods of clustering.

The *discrimination index* is the number of variables that singly discriminate among all the clusters, that is, variables having different values in every cluster description.

Dimensionality reduction is measured by the negative of the essential dimensionality, defined as the minimum number of variables required to distinguish among all complexes in a clustering. It can be computed by applying to the clustering the variable-valued logic minimization algorithm A^q [Michalski, 1975b]. When the discrimination index is greater than zero, the essential dimensionality is exactly one.

The definitions of the above criteria are such that the increase of any criterion value improves the quality of the clustering. The relative influence of each criterion is specified using the "Lexicographical Evaluation Functional with tolerances" (LEF) [Michalski, 1980b]. The LEF is defined by a sequence of *criterion-tolerance* pairs (c_1,τ_1), (c_2,τ_2), ..., where c_i is an elementary criterion selected from the above list, and τ_i is a *tolerance threshold* ($\tau \in [0..100\%]$). In the first step, all clusterings are evaluated on the first criterion, c_1, and those that score best or within the range defined by the threshold τ_1 are retained. Next, the retained clusterings are evaluated on criterion c_2 with threshold τ_2, similarly to the above. This process continues until either the set of retained clusterings is reduced to a singleton (the "best" clustering) or the sequence of criterion-tolerance pairs is exhausted. In the latter case, the retained clusterings have equivalent quality with respect to the given LEF, and any one may be chosen arbitrarily. The selection of elementary criteria, their ordering, and the specification of tolerances is made by a data analyst.

11.5 METHOD AND IMPLEMENTATION

This section describes the algorithm for conjunctive conceptual clustering implemented in the program CLUSTER/2 (the successor to the program CLUSTER/PAF [Michalski & Stepp, 1981]). The algorithm consists of a *clustering module* and a *hierarchy-building module*, which are described in Sections 11.5.1 and 11.5.2, respectively. Section 11.5.1.4 gives an example illustrating the details of the clustering module.

11.5.1 The Clustering Module

11.5.1.1 The Full-search Version of the Algorithm

The basic algorithm underlying the implementation of the clustering module was introduced in [Michalski, 1980a]. Its goal can be described as follows:

Given:

- A collection of events to be clustered, E

- The number of clusters desired, k, and
- The criterion of clustering quality, LEF

Find:

- A disjoint clustering of the collection of events that optimizes the given criterion of clustering quality LEF.

We shall first describe a straightforward, exhaustive-search version of the algorithm, and then show how this version is modified to increase efficiency. The steps are:

1. Initial seeds are determined.

   From the given collection of events E, k events (the initial seeds) are selected. The seeds may be chosen randomly or according to some criterion. (After this first step, seeds are always selected according to certain rules; see step 5).

2. Stars are constructed for seeds.

   For each seed e_i, a reduced star $RG(e_i|E_o)$ is constructed by procedure Redustar, where E_o is the set of remaining seeds.

3. An optimized clustering (a disjoint cover of E) is built by selecting and modifying complexes from stars.

   Every combination of complexes, created by selecting one complex from each star, is tested to see whether it contains intersecting complexes. If so, the complexes are made disjoint by procedure NID.

4. A termination criterion is evaluated.

   If this is the first iteration, the obtained clustering is stored. In subsequent iterations the clustering is stored only if it scores better than previously-stored clusterings according to the LEF (see Section 11.4). The algorithm terminates when a specified number of iterations does not produce a better clustering (this number is defined by a termination criterion, described below).

5. New seeds are selected.

   New seeds are selected from sets of observed events contained in complexes of the generated clustering, one seed per complex. Two seed-selection techniques are used. One technique selects "central" events, defined as events nearest the geometrical centers of the complexes (as determined by the syntactic distance). The other technique, stemming from the "adversity principle[4]", selects "border" events, defined as events farthest from the centers. Ties for central or border events are broken in favor of events which have not been used recently as seeds. The technique of selecting central events is used repetitively in consecutive iterations as long as the clusterings improve. When the improvement ceases, border events are selected.

After selecting seeds, a new iteration of the algorithm begins from step 2.

The algorithm is summarized by the flow chart in Figure 11-5.

Along with a clustering, the algorithm generates k λ-complexes describing individual clusters, and determines how these complexes score on the evaluation criteria in the LEF. The algorithm stops when the *termination criterion* is satisfied. The termination criterion is a pair of parameters (b,p), where b (the *base*) is a standard number of iterations the algorithm always performs, and p (the *probe*) is the number of additional iterations beyond b performed after each iteration which produced an improved cover. The general structure of the algorithm is based on the so-called dynamic clustering method [Diday & Simon, 1976].

The most computationally costly part of this algorithm is the construction of an optimized clustering, given k seed events (step 3). For an illustration, let us assume that $k=2$ and that k "seeds", e_1 and e_2, have been selected from the collection E. In the first step, stars $G_1 = G(e_1|\text{remaining seeds})$ and $G_2 = G(e_2|\text{remaining seeds})$ are generated. Figure 11-6 presents complexes of these stars as branches of a search tree. Branches from the root represent complexes of star G_1 that are $\alpha_{11}, \alpha_{12}, ..., \alpha_{1m_1}$, and branches at the second level (repeated m_1 times) represent complexes of star G_2 that are $\alpha_{21}, \alpha_{22}, ..., \alpha_{2m_2}$. Each combination of complexes, containing one complex from each star, corresponds to one path in the tree. Because any such combination may contain intersecting complexes, procedure NID is applied to each, and the result is a disjoint clustering. These clusterings are ordered according to the quality criterion LEF, and the best one is selected.

11.5.1.2 Path-Rank-Ordered (PRO) Search Procedure Used in CLUSTER/2

The above strategy for determining a clustering from seeds is very simple, but unfortunately too inefficient for solving any interesting practical problems. This is due to the fact that the stars may contain very many complexes. When there are n variables and k seeds, a star may contain up to n^{k-1} complexes

[4]This principle states that if a border, "near hit" event truly belongs to the given cluster, then when selected as the seed it should produce the same clustering as when a central event is used as a seed.

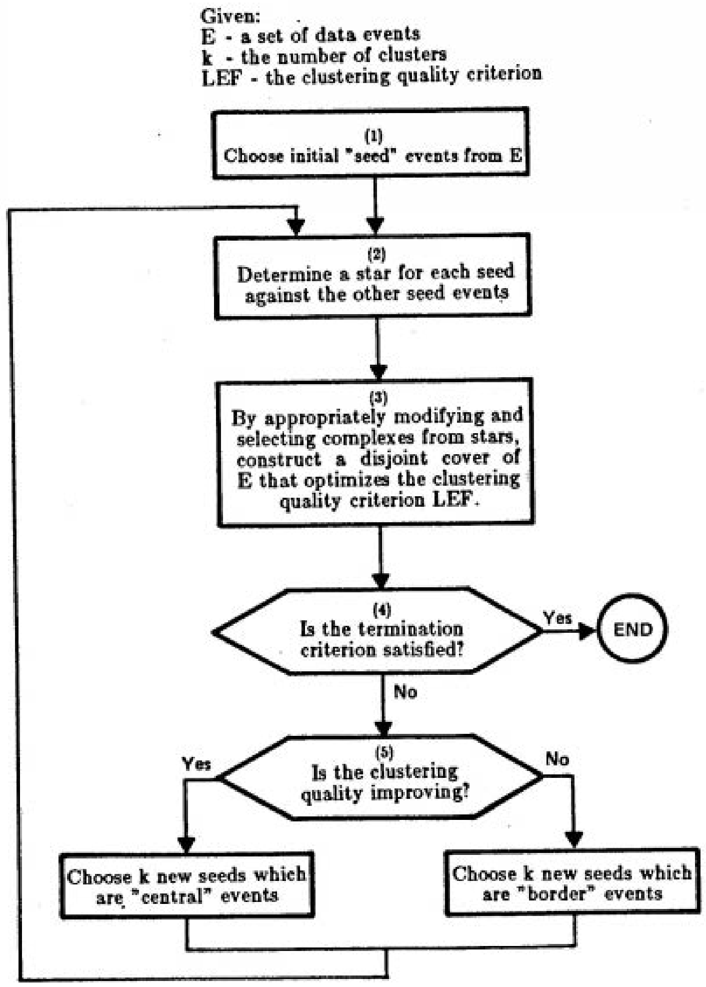
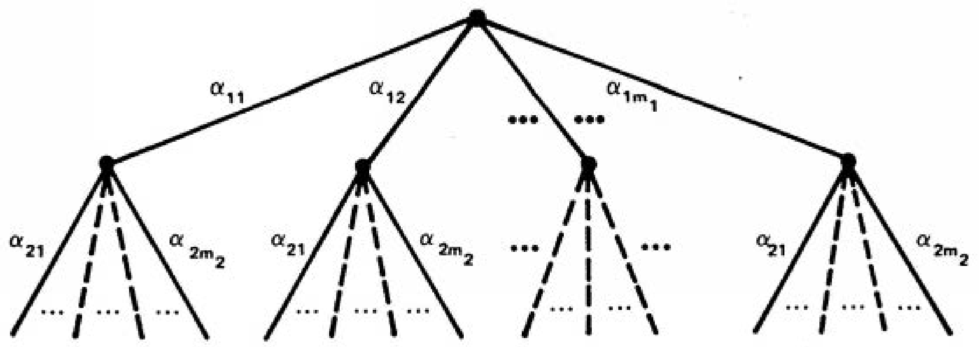
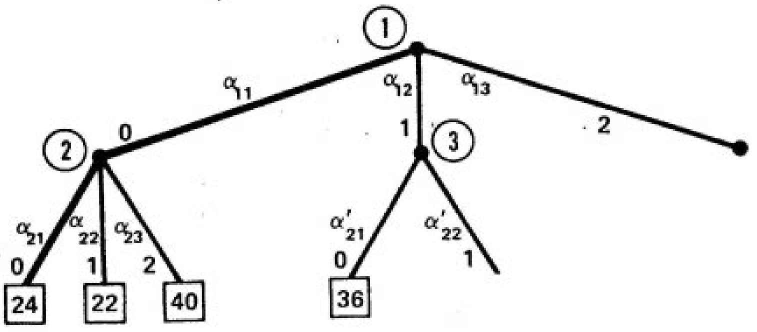

Given:
E - a set of data events
k - the number of clusters
LEF - the clustering quality criterion
(1) Choose initial "seed" events from E
(2) Determine a star for each seed against the other seed events
(3) By appropriately modifying and selecting complexes from stars, construct a disjoint cover of E that optimizes the clustering quality criterion LEF.
(4) Is the termination criterion satisfied?
Yes
END
No
(5) Is the clustering quality improving?
Yes
No
Choose k new seeds which are "central" events
Choose k new seeds which are "border" events

Figure 11-5: The flow chart of the clustering module.



α_{ij} denotes a complex j from star i.

Figure 11-6: The exhaustive-search search tree for $k=2$.



α_{ij} denotes complex j from star i. Integers ①, ②, ... indicate the order of expanding nodes. Integers 0, 1, ... indicate the branch indices. Integers $\boxed{24}$, $\boxed{22}$, ... indicate clustering evaluation scores for each path.

Figure 11-7: The Path-Rank-Ordered search tree for $k=2$ used in CLUSTER/2.

(there are at most n complexes in any of the k-1 elementary stars needed to compute the complete star). Thus, when $n=30$ and $k=3$, there could be up to $n^{k-1}=900$ complexes, and the search tree could have up to 900-way branching at each node, and up to $900^3=729$ million leaves. Absorption laws (as defined in Boolean algebra) will usually eliminate many redundant complexes, but the star may still be too large. Artificial intelligence research on various heuristic search procedures offers various possibilities for reducing the search (for example, Nilsson [1980] or Winston [1977]). To solve this problem, we have adopted some of the known ideas and also developed some new ones. The result is a search procedure called Path-Rank-Ordered (PRO) search that incorporates the following four techniques:

1. Bounding the stars (procedure Boundstar).

   The number of complexes in a star is bounded by a fixed integer m, which assures that the search tree has at most m-way branching. A bounded star contains not just m arbitrary complexes from the initial star, but the m "best" ones.

   At each step of star generation (a multiplication of a set of complexes by the next elementary star; see the Redustar procedure in Section 11.3.11), complexes are first reduced and then arranged in descending order according to the assumed clustering quality criterion LEF. Only the first m complexes are retained for the next multiplication step. This operation is also performed at the end of star generation, so that the final star has at most m complexes. The stars so obtained are called *bounded reduced stars* and denoted $RG(e|E_0,m)$.

   Some elementary criteria measure global properties of a clustering rather than properties of just a single complex (such as the inter-cluster differences). Consequently, when evaluating a complex descending from a node in the search tree that is not the root, the complex is evaluated in the context of complexes associated with the path from the root to this node.

   By bounding the star we gain significantly in efficiency, but give up the assurance that the obtained clustering will be optimal. This is not a significant loss, however, because the clustering obtained at the end of each iteration contributes only the seeds to the next iteration, and thus its optimality is not very important.

2. Generating stars dynamically.

   Because it is necessary to evaluate complexes in the context of previously selected complexes, bounding a star has to be done differently at each node of the search tree. CLUSTER/2 uses a "lazy" strategy, in which a star is generated only when it is needed to expand a node on the path being explored.

3. Searching in order of path rank.

   As we mentioned above, complexes in a bounded star are arranged in descending order according to the LEF. In the search tree, the branch to the best complex is assigned the *branch index* 0, the branch to the next best complex is assigned the branch index 1, and so on, up to the index m-1. The *path index* of a path from the root to a leaf is the sum of the branch indices along the path.

The paths from the root to a leaf represent potential clusterings and are investigated in the ascending order of their path index. Thus, the first path investigated is the one with path index 0, that is, the path containing only the "best" complexes from each star. The next paths considered are those with a path index of one. There are k such paths.

As paths of increasing path index are generated and evaluated, a search termination criterion is applied. This criterion consists of two parameters, *search-base* and *search-probe*. A search-base number of paths is always expanded and evaluated. Then, a search-probe number of additional paths is considered. Each path is processed by NID, and if some complexes are transformed to make them disjoint, the clustering-quality criterion is evaluated again. Whenever a new clustering is better than any previous clustering, it is saved and another search-probe number of additional paths is explored. If the above probing fails to find a better clustering, the search terminates.

4. Tapering the search tree.

The bound of the stars, m, is decreased with the increase of the path index. The search tree is, therefore, more fully developed on the side containing the "higher quality" complexes.

Figure 11-7 shows an example of a search tree generated by CLUSTER/2. The tree is a modification of the tree in Figure 11-6, resulting from the application of the above efficiency-increasing techniques. In Figure 11-6, the maximum value of bound m is set to 3. The root is expanded by constructing the star G(seed$_1$|other seeds,3), whose complexes are α_{11}, α_{12}, and α_{13} (listed in decreasing order of their "quality", as determined by the LEF). The branches representing these complexes are assigned branch indices 0, 1, and 2, respectively. The node attached to branch 0 is expanded next. The star G(seed$_2$|other seeds,3) is generated, creating complexes α_{21}, α_{22}, and α_{23}. Branches corresponding to these complexes are assigned branch indices 0, 1, and 2, respectively. The path 0-0 (having the lowest path index of 0, denoted by heavy lines in Figure 11-7) is considered first. The associated clustering $\{\alpha_{11}, \alpha_{21}\}$ is processed by NID, and the result is saved as the best clustering so far. Next, path 0-1 is considered. The associated clustering $\{\alpha_{11}, \alpha_{22}\}$ is processed by NID and evaluated. If it is better than the previous clustering, it is saved. In order to explore the path 1-0 (the second path with path index 1), the star G(seed$_2$|other seeds,2) is generated. The star contains complexes α'_{21} and α'_{22}. The clustering $\{\alpha_{12}, \alpha'_{21}\}$ associated with the currently investigated path is evaluated. Assuming that the termination criterion has the parameters search-base=2 and search-probe=2, and that the evaluation scores are as shown in Figure 11-7, the tree search terminates after investigating the fourth path 0-2 (since this path exhausts the probing without finding a better clustering). Path 0-1, with the evaluation score of 22, is the best clustering found.

11.5.1.3 Dynamic Modification of Classifications

The obtained clustering partitions all the observed events into disjoint classes. The set-theoretic union of complexes in the clustering does not, however, necessarily cover the whole event space. Consequently, when a given classification is applied to a new event that is "outside" this union, it is not possible to assign this event to any class. In such a case, the classification (clustering) is automatically adjusted to accommodate the new event. This is done by applying the NID procedure (Section 11.3.12), modified as follows. The complexes of the current clustering play the role of "core" complexes, and the new event is treated as an element of the m-list. The event is incorporated into the complex that is the best "host" for it, as determined by NID. As a result, the original complex becomes the Refunion of itself and the event. This way, the initial classification is modified to incorporate the new, unforeseen event. Such a process has a psychological justification, as it is common for people to modify their classifications when some object fails to fit them, by appropriately perturbing the boundaries of the classes.

11.5.1.4 An Example Illustrating the Clustering Module

The following simple example illustrates some further details of the clustering module algorithm. There are ten objects, each described by four variables: x_1, x_2, x_3, and x_4, with three-valued domains, $DOM(x_i) = \{0,1,2\}$, $i=1,2,3,4$. Variables x_1 and x_3 are linear, variable x_2 is structured, and variable x_4 is nominal. The generalization hierarchy of the domain of x_2 is shown in Figure 11-8. Object descriptions (events in the population E) are presented in Figure 11-9. For simplicity, let us assume that the goal is to partition objects into only two classes ($k=2$) using a LEF in which the primary criterion (with tolerance of 0%) is to minimize the total sparseness, and the secondary criterion is to maximize the simplicity of the clustering, (that is, the negative of the number of selectors). Figure 11-10 shows a geometrical representation of the events using a generalized logic diagram [Michalski, 1978]. Each cell in the diagram is labeled by the event it represents. Empty cells represent unobserved events.

The steps of the algorithm follow the diagram in Figure 11-5.

Iteration 1

Step 1 (Figure 11-5, block 1): A subset of $k=2$ observed events (seeds) is selected from the population $E = \{e_i\}$, $i=1,2,...,10$. The seeds can be selected randomly, or they can be chosen as events which are most syntactically distant from each other. In the latter case, as experiments show, the algorithm will usually converge faster. For selecting such "outstanding" events, program ESEL [Michalski & Larson, 1978] is

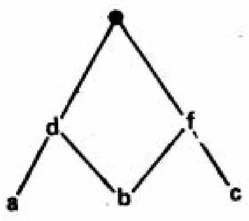


Figure 11-8: The generalization hierarchy of the domain of variable x_2.

Event	x_1	x_2	x_3	x_4
e_1	0	a	0	1
e_2	0	b	0	0
e_3	0	c	1	2
e_4	1	a	0	2
e_5	1	c	1	1
e_6	2	a	1	0
e_7	2	b	0	1
e_8	2	b	1	2
e_9	2	c	0	0
e_{10}	2	c	2	2
Variable Type:	L	S	L	N

(L: linear; N: nominal; S: structured)

Figure 11-9: A data set describing ten objects, using four variables.

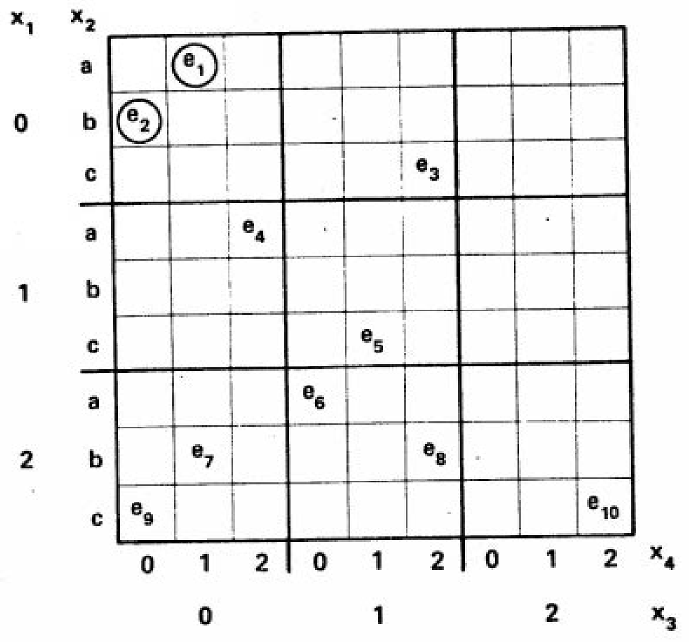


Figure 11-10: A geometrical representation of events e_1 to e_{10}. Encircled events are initial seeds.

used. For the sample problem, let us make a "bad" choice and select two events close to each other, such as e_1 and e_2.

Step 2 (Figure 11-5, block 2): Bounded reduced stars $RG(e_1|e_2,m)$ and $RG(e_2|e_1,m)$, with $m=5$, are generated by procedure Boundstar (described in Section 11.5.1.2):

$$RG(e_1|e_2,m) = \{[x_2 = a][x_3 = 0 \vee 1],\ [x_4 = 1 \vee 2]\}$$
$$RG(e_2|e_1,m) = \{[x_2 = b \vee c],\ [x_4 = 0 \vee 2]\}$$

These stars contain all possible complexes, because m>2. After applying the *closing the interval* and *climbing the hierarchy generalization* rules, the stars become:

$$RG(e_1|e_2,m) = \{[x_2 = a][x_3 = \leq 1], [x_4 = 1 \vee 2]\}$$
$$RG(e_2|e_1,m) = \{[x_2 = f], [x_4 = 0 \vee 2]\}$$

The reference "b $\vee$ c" in the selector for the structured variable x_2 has been replaced by a more general value, f (Figure 11-8).

Step 3 (Figure 11-5, block 3): From each star a complex is selected and appropriately modified, such that the resulting set of k=2 complexes is a disjoint cover of E, and is an optimal or suboptimal cover among all possible such covers, according to the clustering quality criterion. There are four combinations of complexes to consider:

			Sparseness	Complexity
(a)	complex 1:	$[x_2 = a][x_3 \leq 1]$	15	2
	complex 2:	$[x_2 = f]$	47	1
			62	3
(b)	complex 1:	$[x_4 = 1 \vee 2]$	These covers are not disjoint. Procedure NID (Section 11.3.12) is applied to each cover, but the sparseness of resulting clusterings in each case is ≥ 62 and their complexity is 3.	
	complex 2:	$[x_2 = f]$		
(c)	complex 1:	$[x_2 = a][x_3 \leq 1]$		
	complex 2:	$[x_4 = 0 \vee 2]$		
(d)	complex 1:	$[x_4 = 1 \vee 2]$		
	complex 2:	$[x_4 = 0 \vee 2]$		

Cover (a) is selected since it has the minimum total sparseness.

Step 4 (Figure 11-5, block 4): The termination criterion is tested. In our example, the parameters of the termination criterion are: base=2 and probe=2 (Section 11.4). The current iteration is the first of the two base iterations.

Step 5 (Figure 11-5, block 5): A new set of seeds is determined. These new seeds are central events, among the observed events covered by (a). Complex $[x_2 = a][x_3 \leq 1]$ covers the set $\{e_1,e_4,e_6\}$, and complex $[x_2 = f]$ covers the set $\{e_2,e_3,e_5,e_7,e_8,e_9,e_{10}\}$ (notice that value f of x_2 is a generalization of b and c according to Figure 11-8). The central events (as determined by syntactic distance) in these sets are e_4 and e_8, respectively, so they become new seeds.

Iteration 2

Step 2: New stars $RG(e_4|e_8,m)$ and $RG(e_8|e_4,m)$ are generated:

$$RG(e_4|e_8,m) = \{[x_2 = a][x_3 \leq 1], [x_1 \leq 1][x_3 \leq 1], [x_3 = 0]\}$$
$$RG(e_8|e_4,m) = \{[x_1 = 2], [x_2 = f], [x_3 = \geq 1]\}$$

Step 3: All combinations of complexes (obtained by selecting one complex from each star) are subjected to procedure NID and then evaluated. The best clustering is:

		Sparseness	Complexity
complex 1:	$[x_1 \leq 1][x_3 \leq 1]$	31	2
complex 2:	$[x_1 = 2]$	22	1
		53	3

Step 4: This is the last of the base iterations.

Step 5: Complex $[x_1 \leq 1][x_3 \leq 1]$ covers events $\{e_1,e_2,e_3,e_4,e_5\}$ and $[x_1 = 2]$ covers $\{e_6,e_7,e_8,e_9,e_{10}\}$. Since this clustering is an improvement over the previous one (since it has a lower sparseness), the new seeds selected are central events: e_1 and e_8.

Iteration 3

This iteration produces the same clustering as iteration 1.

Step 4: This is the first of the two "probe" iterations.

Step 5: Since the clustering obtained is not better than the previous one, border events are selected as the new seeds: e_2 and e_6.

Iteration 4

This iteration produces a new clustering:

		Sparseness	Complexity
complex 1:	$[x_3 \geq 1]$	49	1
complex 2:	$[x_3 = 0]$	22	1
		71	2

It is the second "probe" iteration. If the obtained clustering was better than the previous best clustering, another probe=2 iterations would be scheduled. Since the sparseness of the clustering obtained in this iteration (71) is not an improvement over the previous best sparseness (53), the termination criterion is satisfied. The best resulting clustering is the one produced in iteration 2:

$$[x_1 \leq 1][x_3 \leq 1]$$

$$[x_1 = 2]$$

Figure 11-11 shows the diagrammatic representation of this solution.

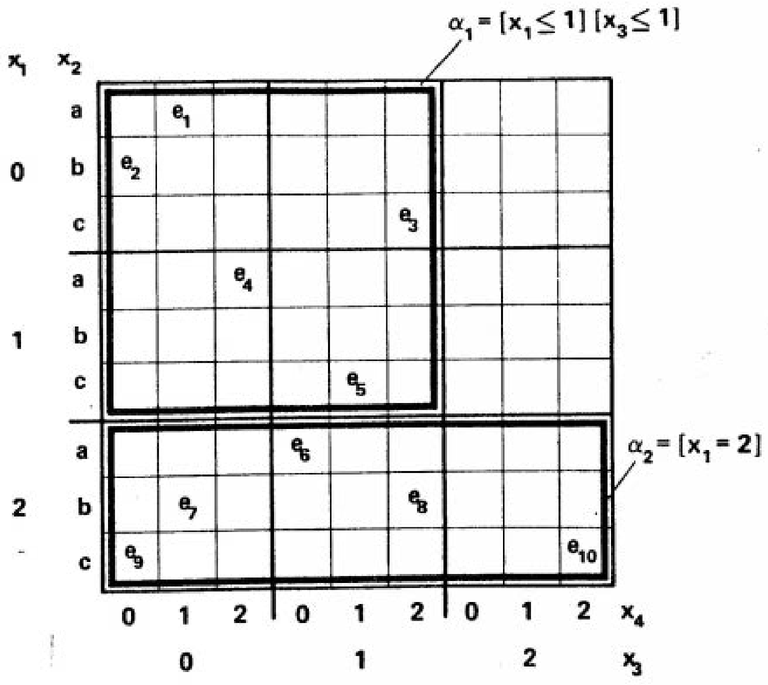


Figure 11-11: A diagrammatic representation of the clustering $\{\alpha_1, \alpha_2\}$.

11.5.2 The Hierarchy-building Module

The hierarchy-building module uses the clustering module to determine a hierarchy of clusters. It performs two loops, one iterative and one recursive. The iterative loop repeats the clustering module for a sequence of values of **k** in order to determine the value for which the most desirable clustering is obtained. Such an approach is computationally acceptable because, in practical applications, most interesting hierarchies will have a relatively small number of branches (that is, a small value of **k**) at each level.

The recursive loop applies the above iterative process at each node of the hierarchy. In the first step, the process is executed for the root, representing the initial event set E. Clusters of E and their conjunctive descriptions are determined. Consecutive steps repeat the same operation for the nodes representing clusters obtained during the previous step. The hierarchy continues to grow from the top down until the "continue-growth" criterion fails to be met. This criterion requires that the fit between the clusters and their descriptions at every level of the hierarchy must be better than at the previous level.

In order to determine the optimal value of k, we must modify the clustering quality criterion so that it can be used to compare clusterings with different numbers of complexes. Such a criterion must reflect the dependency of the fit between the clustering and data on the value of k. As the number of clusters k increases, the fit (measured by the negative of sparseness) will likely increase, since smaller complexes will have smaller sparseness. On the other hand, increasing k increases the complexity of the clustering and therefore is undesirable. A simple criterion that takes into consideration the above trade-off is to require the product

$$\text{Total sparseness} \times (k + \beta)$$

to achieve a minimum value, where β is an experimentally determined parameter balancing the relative effect of the sparseness and the number of clusters k on the solution.

11.6 AN EXAMPLE OF A PRACTICAL PROBLEM: CONSTRUCTING A CLASSIFICATION HIERARCHY OF SPANISH FOLK SONGS

This example presents an application of the above method to the development of a classification hierarchy of 100 Spanish folk songs. The folk songs were characterized by 22 musicological attributes, listed in Figure 11-12. These attributes, as well as other relevant data, were provided by musicologist Pablo Poveda, who studied this problem using traditional methods of numerical taxonomy [Poveda, 1980]. The results obtained by using the traditional methods were not very satisfying, because the generated clusters lacked descriptions, and therefore were difficult to interpret.

The top five levels of the conjunctive hierarchy produced by CLUSTER/2 are presented in Figure 11-13. The criterion of clustering quality was to "minimize the total sparseness". The number of clusters (k) at each level was 2, to meet the requirement of the musicologist.

The top node of the hierarchy corresponds to the whole collection of songs. All the other nodes represent various classes (categories) of songs. The description of each class is a conjunctive statement involving selected folk song attributes. In Figure 11-13, instead of providing the whole cluster description associated with each branch, we show, for simplicity, only the discriminant variables occurring in the given cluster. As it turned out, all nodes in the hierarchy have only one discriminant variable. For example, at the first level, the discriminant variable is the harmonic structure, which takes the value "monophonic" in one cluster and "polyphonic" in the other cluster.

	Variable	Domain
x_1:	Tonal Range	{1..11}
x_2:	Number of Tones	{1..10}
x_3:	Degree of Rubato	{0..5}
x_4:	Degree of Embellishment	{0..5}
x_5:	Degree of Melisma	{0..5}
x_6:	Number of Musical Phrases	{0..5}
x_7:	Degree of Musical Tension	{0..5}
x_8:	Degree of Melodic Line Blending	{0..5}
x_9:	Harmonic Structure	{Monophonic, Polyphonic}
x_{10}:	Religious Setting	{Religious, Secular}
x_{11}:	Sex of Singers	{Same Sex, Mixed Sexes}
x_{12}:	Rhythm	{Weak, Strong, Triple-beat}
x_{13}:	Harmony	{None, Non-Phrygian, Phrygian}
x_{14}:	Homophonic	{Yes, No}
x_{15}:	Instrumental Accompaniment	{Yes, No}
x_{16}:	Female singer	{Yes, No}
x_{17}:	Accompanied by Dancing	{Yes, No}
x_{18}:	A Serenade	{Yes, No}
x_{19}:	A Love song	{Yes, No}
x_{20}:	A Solo	{Yes, No}
x_{21}:	Uses Phrygian Scale	{Yes, No}
x_{22}:	Panegyric	{Yes, No}

Figure 11-12: Variables used to describe 100 Spanish folk songs.

One interesting aspect of the generated hierarchy is that the value sets of some variables have been split into ranges. These ranges can be considered as new (generalized) values of variables. For example, while producing the second level clustering of the monophonic folk songs (the left branch), the range of the degree of "rubato" was split into two ranges 0..3 and 4..5, which can be characterized as "low" and "high", respectively. Similar partitioning of value sets was performed on the degree of embellishment, the degree of melisma, the tonal range, and the number of tones in a song.

The leaf nodes in the hierarchy shown in Figure 11-13, marked by $\alpha_1, \alpha_2, \ldots, \alpha_{11}$ represent groups of songs, whose complete description consists of discriminant variables indicated along the path from the root to the leaf and some additional properties generated by CLUSTER/2, but not shown in Figure 11-13. For example, the group of songs represented by α_1 (8 songs) has the following complete description:

α_1 = [harmonic structure = monophonic] [rubato = low][tonal range = low] [type = secular][instruments used = no]	(discriminant variables along the path from the root to leaf α_1)
&	
[no. of distinct tones = 5..8][dance = no] [panegyric = no][no. of phrases = 1..2] [melisma = 0..2][tension = 1..3]	(additional properties generated by the program at the leaf node)

The hierarchy in Figure 11-13 is a simple and meaningful classification

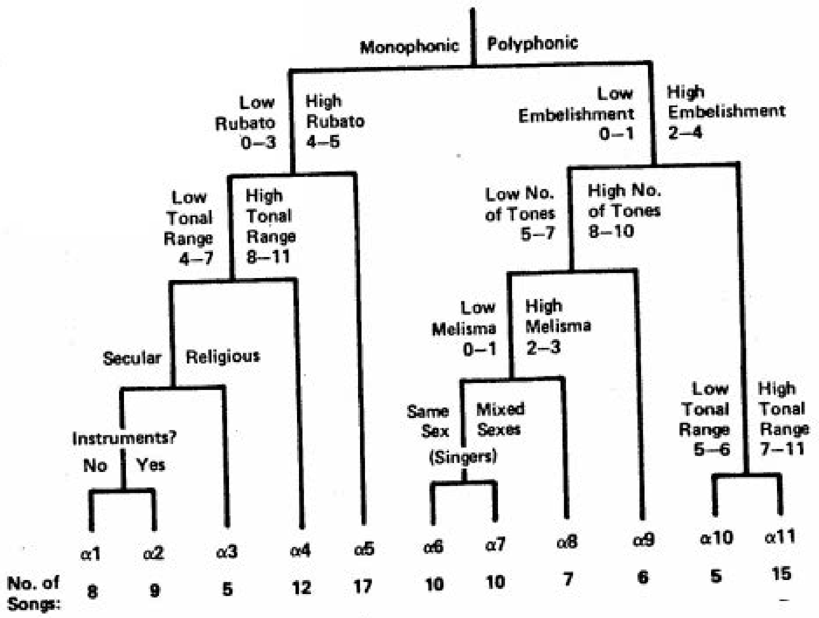


Figure 11-13: A classification hierarchy of Spanish folk songs produced by CLUSTER/2.

of the folk songs. The classes are easy to interpret due to the provided descriptions. If the clustering quality criterion LEF is changed (by selecting different elementary criteria, a different order of the criteria, and/or different tolerances for them) the generated hierarchy may be different. This way the algorithm can generate several alternative hierarchies. The ultimate judgment of which one is the most appropriate for the given application is made by the data analyst.

CLUSTER/2 has also been applied to problems in other domains. One experiment, in the field of agriculture, was to structure a collection of 47 cases of soybean diseases. Each case was described by a vector of 35 components, representing symptoms and characterizations of the diseased plants. CLUSTER/2 "re-discovered" disease classes known to plant pathologists, and provided a description of each class which closely matched the known symptoms of the corresponding diseases [Michalski & Stepp, 1981].

11.7 SUMMARY AND SOME SUGGESTED EXTENSIONS OF THE METHOD

The described method for conjunctive conceptual clustering determines a hierarchy of classes characterizing a collection of objects. Each class has a description in the form of a single conjunctive statement, logically disjoint from descriptions of other classes with the same parent node in the hierarchy, and optimized according to a certain clustering quality criterion. The major difference between this method and methods of numerical taxonomy lies in its extension of the concept of the measure of similarity into a more general notion of "conceptual cohesiveness". Such a measure takes into consideration not only the properties of individual objects, but also their relationship to other objects and, most importantly, their relationship to some predetermined concepts characterizing object collections.

This work represents our early results on the subject of conceptual clustering, and, naturally, many problems remain to be solved. Here are some interesting topics for further research:

- In this method, the variables for describing objects are assumed to be determined *a priori*, and may not be the most appropriate ones for clustering the given objects. A desirable extension of the method would be to implement constructive induction mechanisms able to determine new, more relevant variables during clustering. The use of such variables could lead to simpler and/or more interesting clusterings. A closely related problem of deriving new variables for learning generalized descriptions of concepts from their examples is discussed in Chapter 4.
- The presented method describes object classes solely by conjunctive statements. Although a conjunctive statement is one of the most common descriptive forms used by humans, it is nevertheless a quite limited form. An interesting extension of the work would be to use descriptions which involve additional operators, such as logical implication or equivalence.
- The purpose of building classifications is often to simplify decision making by collecting into one class those situations, observations, or objects that require a similar decision or action. To do this well, the criterion of clustering quality should include knowledge of the goals, purposes, and intentions associated with the problem under investigation.
- In the method described, the classes are organized into a hierarchy. The links of the hierarchy represent just the generalization (set inclusion) relationship between the parent and child nodes. The method could be extended to generate a graph structure (a "classification network") in which links might also represent other relationships between classes. For

example, within such a graph, some links might denote properties that are inherited from parent nodes, and other links might denote properties that differentiate between sibling classes.

- For applications involving clustering visual information, an interesting extension would be to use as conceptual building blocks various standard geometrical shapes, such as circles, ellipses, triangles, rectangles, and so on, and to allow nondisjoint clusterings.
- The problems which are suitable to the CLUSTER/2 algorithm involve objects which can be sufficiently described by variable-value pairs, which are those objects whose internal structure is irrelevant to the problem at hand. When the internal structure of objects is to be considered (when relevant variables include relationships between features of object subparts), the techniques presented here are not adequate (although still applicable, by transforming the structural properties into propositional attributes). An adequate method for clustering such objects requires a richer descriptive language, such as first-order predicate logic or its extension – for example, the annotated predicate calculus described in Chapter 4 of this book.

ACKNOWLEDGEMENTS

The authors thank Mr. Pablo Poveda for providing data used in the musicological experiment, and Professor Richard Selander for providing the numerical taxonomy program NUMTAX. Partial support of this research was provided by the National Science Foundation under Grants Nos. MCS-79-06614 and MCS-82-05166.

REFERENCES

Diday & Simon, 1976.
Diday, E. and Simon, J. C., "Clustering Analysis," in *Communication and Cybernetics 10*, K. S. Fu, ed., Springer Verlag, Heidelberg, New York, 1976.

Gowda & Krishna, 1978.
Gowda, K. Chidananda and Krishna, G., "Disaggregative clustering using the concept of mutual nearest neighborhood," *Man and Cybernetics*, December 1978 , pp. 888-894, Vol. SMC-8, No. 12.

Michalski, 1975a.
Michalski, R. S., "Variable-Valued Logic and Its Applications to Pattern Recognition and Machine Learning," in *Multiple-Valued Logic and Computer Science*, David Rine, ed., North-Holland , 1975.

Michalski, 1975b.
Michalski, R. S., "Synthesis of optimal and quasi-optimal variable-valued logic formulas," *Proceedings of the 1975 International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, May 1975 .

Michalski, 1978.
Michalski, R. S., "A Planar Geometrical Model for Representing Multi-Dimensional Discrete Spaces and Multiple-Valued Logic Function," Tech. report 897, Department of Computer Science, University of Illinois, January 1978.

Michalski, 1980a.
Michalski, R. S., "Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts," *Policy Analysis and Information Systems*, Vol. 4, No. 3, 1980 , pp. 219-244, A Special Issue on Knowledge Acquisition and Induction.

Michalski, 1980b.
Michalski, R. S., "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4, July 1980 .

Michalski & Larson, 1978.
Michalski, R. S. and Larson, J. B., "Selection of most representative training examples and incremental generation of VL_1 hypotheses: the underlying methodology and the description of programs ESEL and AQ11," Tech. report 867, Computer Science Department, University of Illinois, 1978.

Michalski & Stepp, 1981.
Michalski, R. S., and Stepp, R. E., "Concept-based Clustering versus Numerical Taxonomy," Tech. report 1073, Department of Computer Science, University of Illinois, 1981.

Nilsson, 1980.
Nilsson, Nils T., *Priciples of Artificial Intelligence*, Tioga Publishing Co., 1980.

Poveda, 1980.
Poveda, P., "Classification of Folksongs According to the Principles of Numerical Taxonomy," 1980. Unpublished Report, School of Music, University of Illinois at Urbana-Champaign.

Winston, 1977.
Winston, P. H., *Artificial Intelligence*, Addison-Wesley, 1977.