HOW TO STRUCTURE STRUCTURED OBJECTS

by

Ryszard S. Michalski
Robert Stepp

Proceedings of the International Machine Learning Workshop, June 22-24, 1983.

Proceedings of the

International Machine Learning Workshop

June 22-24, 1983

Allerton House
Monticello, Illinois

Sponsored
by
The Office of Naval Research
under Grant No. N00014 83G0030
and
The Department of Computer Science
University of Illinois at Urbana-Champaign

HOW TO STRUCTURE STRUCTURED OBJECTS

Ryszard S. Michalski
Robert E. Stepp

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

ABSTRACT

Past research on automated construction of classifications has been concerned with structuring objects that are characterized solely by attribute-value pairs. The methods stemming from that research did not take into consideration the structural information about the objects nor did they utilize general or problem-specific background knowledge in the process of creating a classification. This paper considers problems of generating classifications of structured objects using knowledge about the inter-relationships of various concepts relevant to describing individual objects as well as object configurations.

1. INTRODUCTION

Creating a classification is typically the first step in understanding and formulating a theory about a collection of observations or phenomena. This process is a form of learning from observation (learning without a teacher) and its goal is to structure given observations into a hierarchy of meaningful categories. Past work on this problem was done mostly under the heading of numerical taxonomy and cluster analysis. It was based on the application of some mathematical measure of similarity between objects, defined over a finite, a priori given set of object attributes. Classes of objects were taken to be collections of objects with high intra-class and low inter-class similarity. The methods assumed that objects are characterized by sequences of attribute-value pairs and that this information is sufficient for creating a classification. The methods did not take into consideration any background knowledge about the relationships among object attributes or global concepts that could be used for characterizing object configurations. Nor did they take into consideration possible purposes of classification.

Since the basis for creating classes was purely mathematical, the resulting classifications were often difficult to interpret conceptually. The problem of interpreting the results was left to the data analyst. In addition, since descriptions of objects are simply attribute-value pairs, the methods were inadequate for creating classifications of structured objects, i.e., objects whose appropriate description involves not only object attributes but also relationships among the object parts as well as attributes of the parts.

The recently developed method of conjunctive conceptual clustering [Michalski and Stepp, 1983] overcomes some of the above problems. Given a set of object descriptions, the method constructs a hierarchy of classes and their descriptions which are in the form of conjunctive concepts. The siblings of any node in the hierarchy are optimized according to a certain global criterion of classification "quality." The method, however, still relies on the attribute-value descriptions of objects. It also has only limited ability to take into consideration the background knowledge of various concepts. This paper addresses the problems of extending the above method in two directions:

(1) how to form classifications of structured objects,

(2) how to formulate and utilize background knowledge in creating classifications.

In the following sections, the above problems are characterized by simple examples and a methodology for their solution is outlined.

2. STRUCTURING STRUCTURED OBJECTS

In order to describe this problem in greater detail, lets us consider a simple example based on rephrasing the problem known as "East- and Westbound trains" [Larson, 1977; Michalski, 1980a] shown in Figure 1. In the old formulation of the problem, given are two collections of trains, those that are "Eastbound" (A to E) and those that are "Westbound" (F to J). These trains are highly structured, each consisting of a sequence of cars of different shapes and sizes. The individual cars carry a variable number of loads of different shapes. In this formulation, this is a typical problem of *learning from examples* (or *concept acquisition*) in which the task is to automatically formulate a simple rule distinguishing between the two classes of objects.

Suppose now the class labels are removed (as in Figure 1) and the problem is to create a meaningful classification of the collection of trains. This is a form of a *learning from observation* (or *concept formation*) problem. To illustrate a possible solution for the concept formation problem, let us go back to the concept acquisition formulation of the problem and consider discriminant descriptions of Eastbound and Westbound trains found by the program INDUCE/2 [Hoff, Michalski, Stepp, 1983]. These descriptions were:

Eastbound trains:

"A train is Eastbound if it contains a short, closed car."

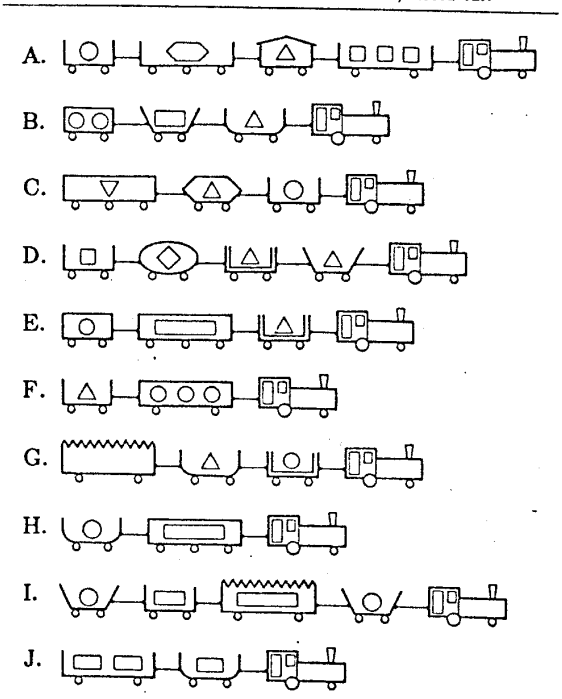

Figure 1. How would you classify these trains?

Westbound trains:

"A train is Westbound if there are just two cars in a train or if a train has a car with a jagged top."

To illustrate the description language used, below is the actual output from the INDUCE/2 program that corresponds to the above descriptions:

Eastbound: $\forall$(train) [contains(train,car1)][length(car1)=short] & [shape(car1)=closed]

Westbound: $\forall$(train) [num-cars(train)=2] v [contains(train,car1)][shape(car1)=jagged top]

Various aspects of learning structural descriptions from examples are given in [Winston, 1977] and [Dietterich and Michalski, 1983]. The problem of concern here is to develop a general method that when applied to the collection of trains without class labels (Figure 1) could potentially restore the above classification or invent one of equal conceptual appeal. In addition it should generate descriptions of the same kind as above for the created classes, with a possible proviso that each class is described by a simple conjunction only (in this case we would have three classes of trains: those with a short closed car, those with a jagged-top car, and those with exactly two cars).

Such classification construction problems occur when one wants to organize and classify observations that require structural descriptions. Problems of this type include classifying physical or chemical structures, analyzing genetic sequences, building taxonomies of plants or animals, and characterizing visual scenes. In Sec. 4 we propose two methods for solving this type of problem, one data-driven and one model-driven. Before discussing these methods we turn to the problem of how to use background knowledge in creating a classification.

3. BACKGROUND KNOWLEDGE

It can be observed that when people create a classification of some observations or partition a system into subsystems, they employ knowledge about various concepts and relationships relevant to describing the observations or the system. Let us consider a simple example. Suppose that we are observing a typical restaurant table on which there are such objects as food on a plate, a salad, utensils, salt and pepper, napkins, a vase with flowers, a lamp, etc. Suppose a person is asked to build some meaningful classification of the objects on the table. One way to create a classification is to perform the following chains of inferences:

- salt and pepper are seasonings
  seasonings are used to add zest to food
  seasoned food is something to be eaten
  things which are to be eaten are edible
  salt and pepper are edible

- salad is a vegetable
  vegetables are food
  food is something to be eaten
  things which are to be eaten are edible
  salad is edible

A similar chain of inferences applied to "meat on a plate" or "cake on a dessert plate" will also lead to the concept "edible." On the other hand, a lamp is not food and is therefore not edible. A vase containing flowers is not food and is therefore not edible. Consequently, one meaningful classification of objects on the table is simply "edible" vs. "inedible." The problem that we pose is this: suppose we are given descriptions of objects on the table in terms of their attributes (including their structural attributes) and we want to have a program that on the basis of these descriptions would create the above classification into edible vs. inedible objects. Obviously, such a program would have to be equipped (among other things) with the above inference rules and with the ability to use them.

4. PROPOSED SOLUTION

This section describes two methods for solving problems of the kind posed above, one data-driven and one model-driven. The data driven method is based on our previous work and reduces the problem of building a classification into a sequence of concept acquisition problems (specifically, problems of determining discriminant descriptions of objects with given class labels). Thus the method will be called RD for "Repeated Discrimination."

The model-driven method is based on selecting *classifying attributes* (attributes that are used to define classes) either from the initially given pool of attributes or from attributes generated by applying inference rules provided by background knowledge. This method will be called CA for "Classifying Attributes."

The common components of the methods are that they both use general and domain-specific knowledge in the process of constructing a classification and that they both utilize a similar criterion for measuring the "quality" of generated candidate solutions. Therefore, before we launch into a description of the methods, we will first briefly discuss our language for describing generated classes and expressing background knowledge inference rules, and then we will discuss a criterion for measuring the quality of proposed classifications.

4.1. The representation language

In order to build a classification of structured objects, we must have an adequate language for describing such objects as well as their classes. This requirement suggests that we use a predicate calculus or some modification of it. Here we have chosen a language called *annotated predicate calculus (APC)*. APC is an extension of predicate calculus that uses several novel forms and attaches to each predicate, variable, and function an *annotation* [Michalski, 1983]. The annotation is a store of information about the given predicate or atomic function. Together with all forms found in predicate calculus, the language also uses a special kind of predicate called a *selector*. A simple selector is in the form:

[atomic-function REL value-of-atomic-function]

where REL (relation) stands for one of the symbols $= \neq < > \leq \geq$. An example of such a selector is

[weight(box) > 2kg]

which means "the weight of the box is greater than 2 kg." A more complex selector may involve *internal disjunction* (disjunction of values of the same atomic function) or *internal conjunction* (the conjunction of atomic-functions having the same value). These two operators are illustrated by the two corresponding examples:

[color(box) = red v purple]	"the color of the box is either red or purple."
[color(box1 & box2) = red]	"the color of box1 and box 2 is red."

Selectors can be combined by standard logical operators to form more complex expressions. For example, a statement "there is a green circle on top of a blue or red square" can be expressed as

$\exists(p_1,p_2)$ [ontop(p1,p2)][color(p1)=green][shape(p1)=circle] & [color(p2)=blue v red][shape(p2)=square]

(Logical multiplication is denoted by the symbol & or by concatenating selectors).

The background knowledge is expressed as a set of APC implicative rules:

CONDITION ⇒ CONSEQUENCE

where CONDITION and CONSEQUENCE are conjunctions of selectors. If CONDITION is satisfied, then CONSEQUENCE is asserted. To illustrate the implicative statement, consider the assertion "vegetables are food" from the example in Sec. 3. It can be expressed:

[is-vegetable(object1)] ⇒ [is-food(object1)]

An alternative way to express it is

[type(object1) = vegetable] ⇒ [type(object1) = food]

In this expression "vegetable" and "food" are treated as elements of the structured domain of the attribute "type." This implication expresses a generalization inference rule called *climbing the generalization tree* (the domain of the attribute "type" is assumed to have a tree structure). Further details on the APC language are given in [Michalski, 1983].

4.2. Measuring the quality of a classification

Creating a classification is a difficult problem because there are usually many potential solutions with no "correct" or "incorrect" answers. The choice of a classification can be made according to some perceived goal of classification or some measure of the quality of the classification. One way to measure classification quality that has been successful in both INDUCE/2 and CLUSTER/2 is to define various elementary, easy to measure criteria specifying desirable properties of a classification and to connect them together into one general criterion called the *Lexicographical Evaluation Functional* (LEF) [Michalski, 1983]. The LEF consists of an ordered sequence of elementary criteria along with tolerances which control to what extent different solutions are considered equivalent. The elementary criteria measure certain aspects of the generated descriptions such as the fit between the classification and the objects, the simplicity of the descriptions (the reciprocal of the number of selectors), the discrimination index (the number of attributes that singly discriminate between all classes), and the dimensionality reduction (the number of attributes that are necessary to classify the objects).

4.3. How to use background knowledge

Building a meaningful classification relies on finding good classifying attributes (attributes that define classes). The data-driven and model-driven methods described in the next section both use background knowledge rules in the search for such attributes. The background knowledge rules enable the system to perform a chain of inferences that add new attributes to object descriptions during the course of classification construction. These new attributes are tested to determine if they are good classifying attributes.

The background knowledge rules can represent both general knowledge managed by the classification program and available to all problems, and problem-specific knowledge provided each time by the data analyst. In either case, the knowledge is supplied in the form of an inference rule (called a background rule, or *b-rule*). Special types of b-rules include expressions of arithmetic relationships *(a-rules)*, such as

girth(object) = length(object) + width(object)

and implicative rules that specify logical relationships *(l-rules)* such as:

[above(p1,p2)][above(p2,p3)] ⇒ [above(p1,p3)]

Each kind is associated with a condition indicating the situations to which the rule is applicable.

Background knowledge is handled somewhat differently in the data-driven and model-driven approaches. This distinction will be characterized below.

4.4 How to form classifications of structured objects

4.4.1. A data-driven approach: concept formation via concept acquisition

This section explains how a problem of concept formation (here, building a classification) can be solved via a sequence of controlled steps of concept acquisition (learning from examples). To begin with, let us briefly describe the program INDUCE/2 which solves concept acquisition tasks involving structured objects.

Given a set of objects described by annotated predicate calculus and arranged into two or more classes, INDUCE/2 generates a description of each class (in the form of annotated predicate calculus expressions) that covers all the objects in the described class and no objects in any other class. This is accomplished in the following manner.

The objects are divided into two sets: objects belonging to the class being described (called set F1), objects belonging to any other class (called set F0). One object at a time is selected from set F1 (the "focus of attention") and a "star" is built that covers the selected object against all objects in set F0. A star is a set of all alternative descriptions that are maximally general, cover the "focus of attention" object (the *seed*) and possibly other objects from F1, and cover no objects from F0 [Michalski, 1983]. To control computational explosion, a parameter MAXSTAR restricts the number of alternative descriptions that are retained during the star generation process. The "best" description in the star (according to a LEF criterion) is selected, and it becomes part of the solution. The objects covered by the selected description are removed from set F1 and the process is repeated by selecting another seed object (from among those not yet covered) and building a star for it. When all objects in the set F1 have been covered, the solution is complete.

The INDUCE/2 algorithm can be adapted for solving classification construction problems. Given a set of unclassified objects, k objects are selected and treated as individual representatives of k imaginary classes. The INDUCE/2 algorithm then generates descriptions of each representative that are maximally general and do not cover any other representative object.

These descriptions are then used to determine the most representative object in each class (defined as the set of objects satisfying the class description). The representative objects are then used as new seeds for the next iteration. This process stops when either consecutive iterations converge to some stable solution, or when a specific number of iterations will not improve a classification (from the viewpoint of the criterion LEF).

The control layer that provides the representatives to the concept acquisition process must be able to pick good representatives or else the resulting classifications will be arbitrary without revealing underlying patterns in the data. One technique that has been used in a variety of clustering situations is to pick representatives at random in the first step. In the next steps *central* representatives or *extreme* representatives are selected alternately [Michalski and Stepp, 1983].

This approach requires the selection of a defined number of representative objects (which corresponds to the number of classes). Since the best number of classes to form is usually unknown, two techniques are used:

(1) varying the number of classes, and

(2) composing the classes hierarchically.

Since the classification to be formed should be easy to understand by humans, we assume an upper limit on the number of classes that stem from any node of the classification hierarchy. This limit is assumed to be in the range of 4 to 7. Since this limit is small it is computationally feasible to repeat the whole process for

each number of classes. The solution that optimizes the score on the LEF (with appropriate adjustment for the effects of the number of classes on the score) indicates the best number of classes to form at this level of the hierarchy.

To give an example solution to the "Trains" problem from Figure 1, we show in Figure 2 a classification created by a data-driven approach (using an earlier method described by Stepp [1978]). We find this kind of solution appealing because the difference between classes is striking, yet not obvious by casual inspection.

4.4.2. A model-driven approach

In contrast to the data-driven method described in the preceding section, we will now describe a model-driven approach. The underlying goal of the model-driven approach is to find one or more classifying attributes whose observed value sets can be split into ranges that characterize each class. The important aspect of this approach is that the classifying attribute can be derived through a chain of inferences from the initial attributes during the course of constructing the classification. The classifying attributes sought are the ones that lead to classes of objects maximizing the global criterion of classification quality, (e.g., classes representing simple concepts and/or containing objects that share values for all attributes considered important).

The importance of an attribute can be determined either on the basis of the classification goal or by the fact that it implies many other attributes. For example, if the goal of the classification is "finding food," the attribute "edibility" from Sec. 3 is the important classifying attribute. The second way of determining the importance of an attribute can be illustrated by the problem of classifying birds. The question of whether "color" is a more important classifying attribute than "is-waterbird" is answered in favor of "is-waterbird" because it implies more secondary attributes than does the attribute "color" (e.g., is-waterbird implies "can-swim," "has-webbed-feet," "eats-fish").

There are two fundamental processes that operate alternately to generate the classification. The first process searches for the classifying attribute whose value set can be partitioned to form

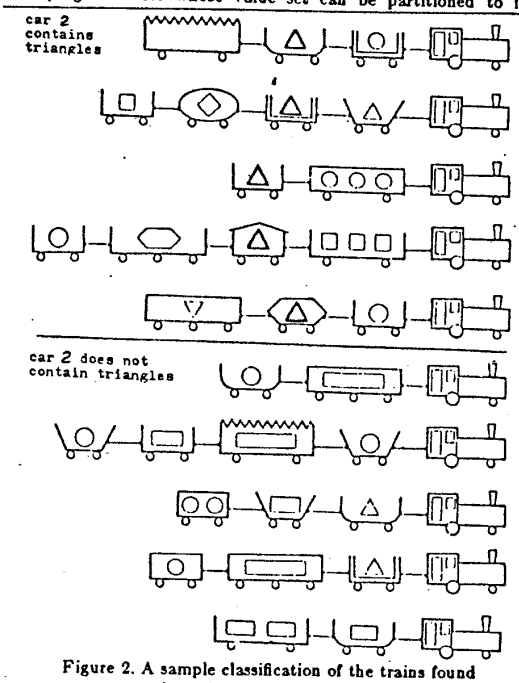

Figure 2. A sample classification of the trains found by a data-driven approach.

classes such that the produced classification scores best according to the LEF. The second process generates new attributes from logical and arithmetic combinations of other attributes, as guided by background knowledge inference rules.

The attribute search can be performed in two ways. When the number of classes to form (k) is known in advance, the process searches for attributes having k or more different values occur in the descriptions of the objects to be classified. These values are called the *observed values* of the attribute. Attributes with the number of observed values smaller than k are not be considered.

For attributes with observed value sets larger than k, the choice of the mapping of value subsets to classes depends on the resulting LEF score for the classification produced and the type of the value set.

For attributes with nominal (unordered) value sets, any combinations of values can be mapped to classes. For attributes with linearly ordered value sets, only non-overlapping closed intervals of values are mapped to classes. For attributes with generalization-tree ordered value sets [Michalski, 1983], sibling values in the generalization tree are mapped to classes.

For example, if k is 4 and the two attributes x_1 and x_2 can have 7 and 3 values, respectively, only x_1 is a candidate for determining the classification. The selection of which of the 7 values denote which of the 4 classes depends on the type of the value set. If we assume that the value set is linearly ordered, one possible classification involving 4 classes could be

$[x_1 = 0..1]$ (class 1)
$[x_1 = 2..3]$ (class 2)
$[x_1 = 4]$ (class 3)
$[x_1 = 5..6]$ (class 4)

This classification would be selected if it is best according to the quality criterion LEF (which also takes into consideration values of other attributes).

When the number of classes to form is not known, the above technique is performed for a range of values of k. The best number of classes is indicated by the classification that is best according to the LEF.

The attribute generation process constructs new attributes from combinations of existing attributes. Certain heuristics of attribute construction are used to guide the process. For example, two attributes that have linearly ordered value sets can be combined using arithmetic operators. When the attributes have numerical values (as opposed to symbolic values such as "small," "medium," and "large") a trend analysis can be used to suggest appropriate arithmetic operators as in BACON 4 [Langley, Bradshaw, Simon, 1983]). Predicates can be combined by logical operators to form new attributes through l-rules. For example, the rule

[cold-blooded(a1)][offspring birth(a1)=egg] $\Rightarrow$ [type(a1)=reptile]

yields a new attribute "type" with a specified value "reptile." Using this rule and similar ones, one might classify some animals into reptiles, mammals, and birds (even though the type of each animal is not stated in the original data).

5. SUMMARY

We have discussed problems of creating a classification of structured objects and have outlined a data-driven method and a model-driven method. The first method transforms concept formation into a sequence of concept acquisition tasks. The second method forms classes by partitioning the value set of one or more classifying attributes that were determined as most appropriate according to a classification quality criterion (LEF). The classifying attributes are either selected from the initially given ones or derived through an application of inference rules defined in the problem background knowledge. Both approaches make extensive use of

attribute constructions through inference rules provided by general and problem-specific background knowledge. The capability to incorporate background knowledge in classifying structured objects adds a new dimension to the tasks of concept formation and data analysis.

REFERENCES

Dietterich, T.G. and Michalski, R.S., "A Comparative Review of Selected Methods for Learning from Examples," chapter in *Machine Learning*, R.S. Michalski, J. Carbonell and T. Mitchell (Editors), Tioga Publishing Company, 1983.

Hoff, W.A., Michalski, R.S., Stepp, R.E., "INDUCE 2: A Program for Learning Structural Descriptions from Examples," Technical Report, Department of Computer Science, University of Illinois, Urbana, Illinois, September, 1983.

Larson, J., "Inductive Inference in the Variable-Valued Predicate Logic System VL_{21}: Methodology and Computer Implementation," Ph.D. Thesis, Report No. 869, Department of Computer Science, University of Illinois, Urbana, Illinois, 1977.

Langley, P., Bradshaw, G.L., Simon, H.A., "Rediscovering Chemistry With the BACON System," chapter in *Machine Learning*, R.S. Michalski, J. Carbonell ad T. Mitchell (Editors), Tioga Publishing Company, 1983.

Michalski, R.S., "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis ad Machine Intelligence*, Vol. PAMI-2, NO. 4., pp 349-361, July, 1980a.

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

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

Michalski, R.S., "A Theory and Methodology of Inductive Learning," chapter in *Machine Learning*, R.S. Michalski, J. Carbonell and T. Mitchell (Editors), Tioga Publishing Company, 1983.

Michalski, R.S. and Stepp, R.E., "Learning From Observation: Conceptual Clustering," chapter in *Machine Learning*, R.S. Michalski, J. Carbonell and T. Mitchell (Editors), Tioga Publishing Company, 1983.

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