88-11

Machine Learning from Structured Objects

ROBERT E. STEPP (STEPP@UICSL.CSL.UIUC.EDU)

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 U.S.A.

Abstract

Machine learning techniques applied to structured objects frequently use a predicate calculus representation to model the world. Unless careful attention is given to the semantics of this model, the results of inductive inference over descriptions of structured objects have unanticipated interpretations. In this paper, a motivation is given for the importance of careful attention to the semantics that underly descriptions of structured examples and categories of such examples. Particular attention is given to the use of *must not* clauses and the ability to determine relevant attributes. An example from INDUCE/NE is used to illustrate *must not* clauses with the INDUCE algorithm. An example from CLUSTER/CA is used to illustrate the use of knowledge about relevant attributes in learning.

1. Introduction

The domain of structured entities is naturally of interest to beings that live in a 3-D (rather than a 1-D) world. Even though the early machine learning systems used structured objects (e.g., Winston, 1975), relatively few subsequent projects have worked with structured examples. A problem is the complexity of dealing with structured representations. A natural representation scheme is first order logic, but there are frequently difficulties with handling quantification in a way that provides a good model of the real world semantics of structured objects and that holds up under our notions of closed world assumptions.

The INDUCE system (Larson, 1977; Hoff, Michalski, & Stepp, 1983) is able to learn disjunctive rules describing classes of structured objects using variable-valued logic (Michalski, 1983). The CLUSTER/CA system (Mogensen, 1987) is able to discover concepts describing structured objects through knowledge based conceptual clustering. Both approaches represent departures from Winston's learning system that formulates conjunctive concepts by adding links to a semantic network representation (Winston, 1975).

1.1. A problem of representation

Structured objects are described by semantic networks or equivalent first order logic statements involving attribute relationships for

(1) attributes of the entire structure (e.g., total mass),
(2) attributes of some identifyable part (e.g., color of part1), or
(3) attributes of interrelationships between parts (e.g., part1 is ontop of part2).

Here the use of predicates, semantic nets, and Michalski selectors are considered equivalent.

Consider structured object x in Fig. 1. One would say that it has three parts, two of which are upright supports and one of which is a horizontal triangular piece that is on top of both supports. It is possible that physical dimensions of parts, their mass, their composition, their surface texture, their color, etc. could be specified as well. An overlooked subliminal issue is "how to specify the abstract idea of a part?"

Let us assume that any real object can have multiple alternative structural descriptions, with different ways to decompose the entity into parts. For example, it is a matter of problem requirements whether a screwdriver is viewed as one single part or decomposed into handle and blade. Given a particular view of the decomposition of the object into parts, those parts can be represented either as constants or as variables. For example, the description of object x in Fig. 1 could be

contains(x,p100,p110,p133) • **upright**(p100) • **upright**(p110)
horizontal(p133) • **ontop**(p133,p100) • **ontop**(p133,p110)

in which p100, p110, and p133 represent unique entities within the entire universe. People generally reserve this type of representation for those cases where an arbitrary concept of identification is to be used. For example, as in a recognition rule such as

is(body,body542) → id(body,R2D2).

A droid may look like R2D2, but it is not R2D2 unless it is precisely that particular robot. Such a concept would cover at most one object at a time.

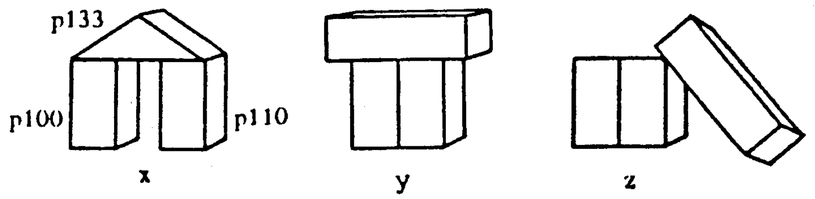


Fig. 1. Some structured objects.

Rules of this type do not generalize.

In cases for which generalization based learning can be applied, one may assume that parts with exactly the same attributes are equivalent and need not be distinguished. Structured objects described using such parts would have descriptions such as this description for x:

$$\exists\, p1,p2,p3\ \mathbf{contains}(x,P1,P2,P3) \cdot \mathbf{upright}(P1) \cdot \mathbf{upright}(P2)$$
$$\mathbf{horizontal}(P3) \cdot \mathbf{ontop}(P3,P2) \cdot \mathbf{ontop}(P3,P1)$$

P1, P2, P3 are existentially quantified variables that take on only mutually distinct values. Note that P1 and P2 are distinct but completely equivalent parts (equivalent here except for left-right role reversal). This method of representing structured objects includes a buried generalization step for changing constants into variables.

1.2. A problem of semantics

It is interesting to look at the semantics behind the representation. Arbitrary symbols (like p100) used to identify parts are themselves without any semantic interpretation. The semantics are captured by the other forms: the quantification, predicates over part symbols, and relations on values returned by functions on part symbols. Thus, we expect concepts for structured objects to have semantic interpretation not by referring to just certain part constants, but by revealing the pattern of attributes that any equivalent part must fulfill. The concept of an arch as "something that contains p100, p110, and p133" is not interesting. The concept of "something that contains two support-like parts with a lentil-like part on top" is more useful, with "support-like" and "lentil-like" being lower level concepts of part categories.

The reason for pointing out the obvious semantic utility for variables is that this approach immediately leads to problems. Consider again example x of Fig. 1. described using existentially quantified variables as above. The description of the arch is a maximally specific concept that describes this one example in the representation space of attributes sufficient for expressing the relevant characteristics of this kind of arch. Since the concept used to describe this one example uses existential quantification, it actually describes infinitely many variations of structures that have the requisite parts.

Suppose a tiny "arch" is placed atop a huge building. Is this new object an arch? The problem is that it matches with the "maximally specific" concept of the arch example from Fig. 1 but we would hardly call the building an arch just because it has nearly irrelevant arch-like parts. Somehow an arch must not have parts that are forbidden or that make the canonical arch-like parts irrelevant. For an arch, having a huge building attached makes the whole example not an arch. Having a speck of dust attached however does not matter.

Concepts learned from structured objects can have two very different uses. Given a set of structured examples, one can build a concept that makes a statement about what the example *is* (e.g., it *is* an arch with all the appropriate deductive entailments that stem from this finding), or one can build a concept that makes a statement about what the example *contains* (e.g., it *contains* one or more instances of "arch" with all appropriate deductive entailments). This distinction is often overlooked. For example the INDUCE program works in a way permitting it to infer statements about what examples contain, but the interpretation usually given to the rules it produced is a statement about what the example is.

Structured training and testing examples are always collections of parts extracted from some larger whole. Obviously any real arch rests on the ground or is in some other context. Having a structured object is therefore the result of some object detachment process that purposefully breaks the attachments between the borders of the structured object and its environment. This *detached* object is what we mean by the word "object." It is described under a given representation scheme and used either to learn structural descriptions, or to test them. Rules that deduce what the object *is* work with the whole of the detached structure. Rules that deduce what the object *contains* work with just selected parts.

1.3. IS versus CONTAINS

For "contains" semantics, the normal existential quantification approach is appropriate: any example that has the characteristic parts is covered by the description, regardless of what other parts there are. Using "is" semantics things need to work differently: any example that has the characteristic parts and only *allowable* auxiliary parts should be covered by the description. The auxiliary parts can be checked in at least four different ways:

(1) allowable auxiliary part attributes can be learned from inductive inference—any allowable auxiliary part must be covered by some maximally specific generalization from positive events. For example, an arch is two supports with a lentil on top and an optional vine or an optional spider or an optional street sign, etc. As more positive events are seen, the concepts used to describe optional parts would become more and more general.

(2) allowable parts must be plausible according to some problem specific theory—some process of plausible inference must decide that the auxiliary parts are reasonable extensions of the object. For example, any optional part that is less than 1% the size of the arch or any part that could be used to build an arch is permitted as a part of an arch.

(3) a part that is irrelevant to the intended use of the concept is allowable. For example, the attachment of ornaments is irrelevant to an arch

considered for its load-bearing design.

(4) there must not be any auxiliary part that is forbidden. For example, an arch must not have a bridge across a river attached to it (the object then would be considered a bridge with an arch part).

Systems that try to generate class identification rules (*is* rules) but which actually build superclass membership rules (*contains* rules) suffer from a common problem: examples from different classes can be glued together to build a new example that is guaranteed to be recognized by rules for both classes. Thus, rules produced by such systems are inherently non-disjoint and always intersect at some virtual event. To properly apply semantics for learning *is* rules, learning algorithms need to be able to infer *must not* conditions and goal-oriented attribute and part relevancy. Some recent work on these two topics is presented below.

2. Inferring Concepts Based on MUST and MUST NOT

Most people would think that an arch with a vine growing on it would still be an arch unless it were given as a negative example. If given such a negative example, Winston's algorithm would identify a MUST NOT condition (an arch must not have a vine growing on it) while the INDUCE algorithm would simply fail (it would report inconsistent data) being unable to specialize the description of the arch in any way to exclude the arch with a vine on it. In this situation, the INDUCE program needs a way to use attributes from negative events in building concepts.

Winston's system for learning the concept of "arch" accepts a sequence of positive and negative structured examples in semantic net form, marking certain links in the semantic net as MUST or MUST NOT links. Positive events trigger the generalization of the concept within the constraints of the MUST and MUST NOT links. The effectiveness of the learning algorithm is greatly influenced by the order in which negative and positive events are presented: this system practices a form of incremental rather than batch learning.

Consider the examples found in Fig. 1 and the learning that takes place from observing the first object. It would seem that learning first from a negative example (such as z) would lead to disjunctive concepts such as "an arch MUST NOT have supports that touch" or "an arch MUST NOT have a leaning lentil", and that learning first from a positive example (such as x) would lead to the simpler concept "an arch can be a horizontal lentil on top of two supports." Of course there is a fallacy here. Just as one never knows in which of the various ways example z truly fails to be an arch, one is equally uninformed about in which ways example x truly is an arch. The whole point of learning about structured examples is trying to find what is relevant and what is irrelevant, while generalizing the concept for the sake of conciseness.

Consider a learning algorithm that only finds "MUST" constraints. Such an algorithm operating on positive example x and negative example y could note that example y fails to have the supports that are apart and therefore they MUST be apart. Some part of the structured representation for x would have to include the statement *apart(support-1,support-2)*. Were this fact to be omitted from the description for x but with y having the extra fact *touching(support-1,support-2)*, the problem would be insurmountable. There is nothing x MUST have that y doesn't have and thus there is nothing to be learned. This is the approach taken by previous versions of the INDUCE algorithm. For some problems, MUST constraints alone appear to be sufficient, and the success of INDUCE would seem to support this view.

The problem outlined in the preceding paragraph is remedied by introducing MUST NOT constraints in the concept language, or through the application of closed world assumptions that could, for example, support the assumption *apart* for example x because its description does not specifically state that the supports are *touching*. Note that the choice of which closed world assumption to make depends on what negative example is being considered: seeing example y for which *touching* holds, and seeing no information about *touching* in x, one then concludes the opposite for x. This is not very appealing: the semantics for determining whether a concept covers an example shift from showing that the example possesses the necessary attribute relations to showing that it possesses no contradictory relations, i.e., a thing is an arch unless it has some kind of part that arches cannot have.

The more useful approach is the one Winston took: to have both MUST and MUST NOT conditions for a concept. Using the representation vehicle of predicate logic, the MUST conditions map to existentially quantified statements and the MUST NOT conditions map to negated existentially quantified statements.

Fig. 2 shows some blocks world examples given to INDUCE/NE

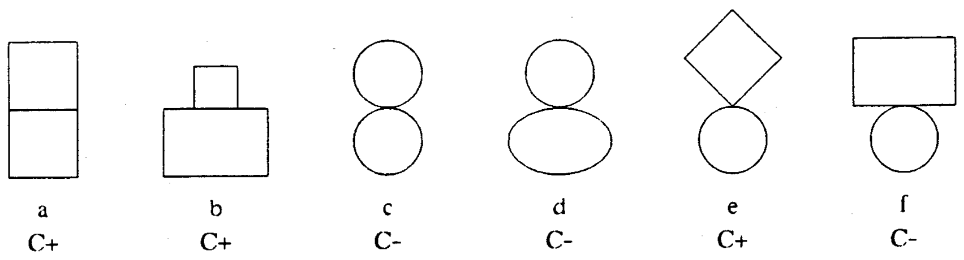


Fig. 2. Blocks-world examples for INDUCE/NE.

(Whitehall, 1987). After being shown examples a to d belonging to classes C+ or C- as indicated in Fig. 3, the program produced the descriptions

$$\exists\, P1\ [shape(P1)=curved] \rightarrow C-$$
$$\exists\, P1,-P2\ [shape(P1)=polygon]\ [shape(P2)=curved] \rightarrow C+$$

where -P2 denotes a negative existentially quantified variable that acts as a censor if bound. That is, the expression is satisfied if P1 can be bound so that all selectors involving only P1 are satisfied and P2 either cannot be bound or all selectors involving P2 fail to be satisfied. Thus the last rule reads: "infer C+ if there exists a polygon shaped part and no curved part." The rule for C- that was learned from the first four examples also covers example e. When example e is presented as an example in class C+, the system realizes that e is incorrectly covered and specializes the *must not* part of the description for C+ and the *must* part in the description for C-. The result produced by INDUCE/NE is

$$\exists\, P1,P2\ [shape(P1)=curved][ontop(P1,P2)] \rightarrow C-$$
$$\exists\, P1,-P2\ [shape(P1)=polygon][shape(P2)=curved][ontop(P1,P2)] \rightarrow C+$$

Finally, after processing example f the rules become

$$\exists\, P1,-P2\ [shape(P1)=curved][ontop(P1,P2)][shape(P2)=square,diamond] \rightarrow C-$$
$$\exists\, P1\ [shape(P1)=square,diamond] \rightarrow C+$$

The current version of INDUCE/NE does not do plausible inference and cannot generate clauses that mean "no other parts permitted." These features are the target of future research. Without them, maximally general *must not* parts of rules are often the same as the *must* part of other rules. With negative existential quantification INDUCE/NE is able to handle incremental and batch learning for example sets that regular INDUCE cannot handle. The extended algorithm of INDUCE/NE is important for most structured object learning situations.

3. Inferring Attribute Relevancy

Another problem with learning algorithms for structured objects is not knowing whether to generalize over the structure, over values of attributes of subparts, or some combination of the two. The size of the space of potential descriptions is immense. This can be managed by heuristic preferences provided by background knowledge and/or structural simplification including substructure decomposition into derived structural macros.

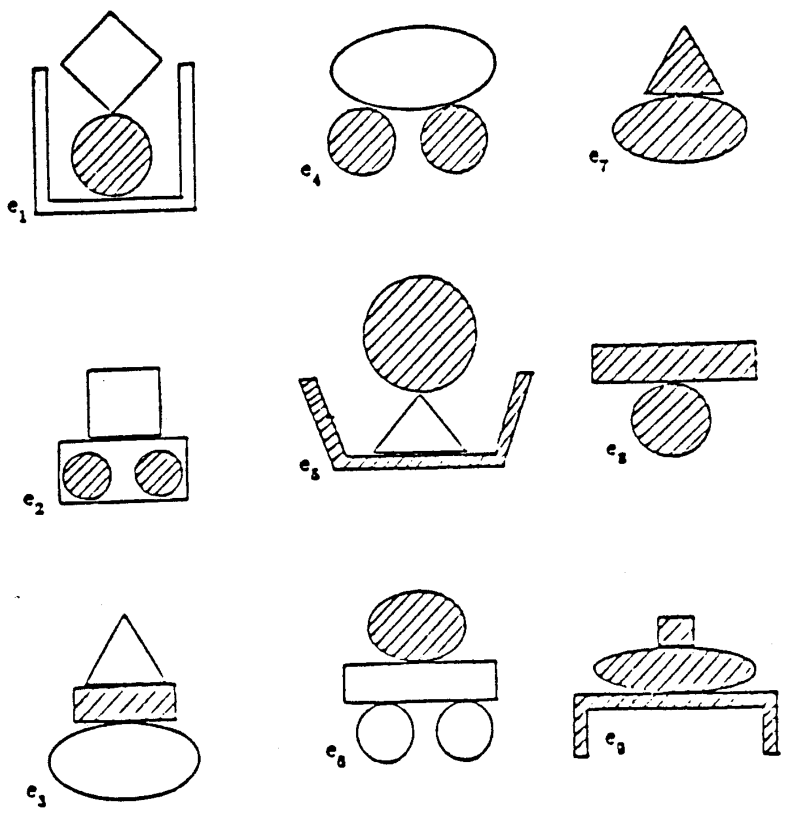
The task of clustering structured objects is one for which the relevance of attributes has a great effect. An algorithm for performing conceptual clustering in a knowledge directed way using background knowledge consisting of deductive inference rules and a goal dependency network for determining estimated attribute relevancy has been described by Stepp & Michalski (1986). This algorithm has been implemented in a program CLUSTER/CA developed by Mogensen (1987) that is derived from approaches and algorithms used by INDUCE (Larson, 1977; Hoff, Michalski & Stepp, 1983), CLUSTER/S (Stepp, 1984), and an algorithm described by Stepp & Michalski (1986). This research provides additional steps towards knowledge guided inference and the formulation of concepts that are generated based on the a goal oriented determination of which attributes and parts are relevant and which are irrelevant.

The CLUSTER/CA algorithm performs conceptual clustering given a set of structured examples, a goal for clustering (problem specific or general), and background knowledge for deriving new attributes and inferring attribute relevancy from the goal. The relevancy of attributes is determined by the use of a Goal Dependency Network that links goals to subgoals and subgoals to relevant attributes. This knowledge is used to determine which of the given attributes are relevant and to trigger the application of additional knowledge to derive attribute values when they can be inferred from the given data.

As an illustration of this algorithm, consider the set of figures shown in Fig. 3. Let there be a goal dependency network that contains a high level goal of "do simple shape recognition" that indicates that the more relevant attributes include number of parts, shapes, sizes, and textures; the shape, size, and texture of object parts; and particular focus on the top and bottom parts.

The system is provided with background knowledge for inferring part counts and the count of parts with specific attributes, such as the number of curved parts; and also whether a shape is polygon or curved. The objects in Fig. 3 are described by low-level attributes such as size, shape, texture, mass, etc. of each part; and ontop, inside, and next-to relations. The CLUSTER/CA system takes the goal "do simple shape recognition" and infers subgoals (none are needed for this simplified example) and then relevancy scores for attributes found in the goal dependency network. Several clusterings based on conceptual descriptions built from combinations of attributes with high relevancy scores are considered and the best n (a parameter) clusterings are reported. For the demonstration problem presented in Fig. 3, there are four alternative results, as shown in the figure.

A study of this approach (Mogensen, 1987) shows that CLUSTER/CA is much less computationally expensive than the non-goal-directed method used in CLUSTER/S. The penalty of CLUSTER/CA is the need to acquire and maintain a possibly large amount of background knowledge. Relevancy of attributes and application heuristics for inference rules used to derive new attributes constitute problem meta-knowledge. Such meta-knowledge needs to be incorporated into learning algorithms that are to handle structured objects in meaningful ways. When category identification rules (*is* rules) are generated,



Best clusterings produced by CA:

Number of objects is "Two" | "Three" | "Four"
Number of circles is "Zero" | "One" | "Two"
Shape of the top object is "Polygon" | "Curved"
Number of different shapes is "Two" | "Three"

Fig. 3. CLUSTER/CA applied to structured objects.

meta-knowledge about attribute and part relevancy needs to be bound to the rule for use in deciding if auxiliary parts of object instances are allowable under the learned concept. CLUSTER/CA is but a beginning in following this lead.

4. Conclusion

When moving from learning about zeroth order phenomenon ("flat" examples that can be represented in propositional logic) to learning about first order phenomenon (structured examples that can be represented in first order logic) important semantic issues arise. The higher order brings a choice of two kinds of rules over structured examples, identifying classes of structured entities (what the object *is*) versus identifying superclasses of parts (what the object *contains*). Rules about what objects *contain* can be expressed by existentially quantified statements such as those produced by INDUCE. Rules about what an object *is* require a different semantic interpretation using exact quantification in which each part of a candidate object is accounted for, by inclusion, by non-exclusion, or by plausible inference.

The INDUCE algorithm is a robust mechanism for conducting inductive inference over structured examples. The system INDUCE/NE demonstrates one dimension of added capability for providing a mechanism to handle inductive inferences over negatively existentially quantified statements that provide *must not* semantics. The system CLUSTER/CA demonstrates another capability needed to handle structured examples by guiding the learning through the use of strong preferences based on constructed attributes and inferred attribute relevancy. Portions of a more complete solution that remain to be developed include a rule mechanism for plausible inference of part relevancy and a way to bind meta-knowledge from the learning into rules for use in-checking for allowable auxiliary object parts.

Acknowledgements

The author wishes to thank Bradley Whitehall and Robert Reinke for their assistance and suggestions concerning the ideas in this paper. The program INDUCE/NE was developed by Bradley Whitehall. The program CLUSTER/CA was developed by Brian Mogensen. Partial support for this research was provided by the National Science Foundation under grants NFS IST 85-11170 and NFS DCR 84-06801, and the Department of Defense Advanced Research Projects Agency under grant N00014-85-K-0078.

References

Dietterich, T.G., & Michalski, R.S. (1981). Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence*, *16, July*, (pp. 257-294).

Hoff, W., Michalski, R.S., & Stepp, R.E. (1983). INDUCE 2: A program for learning structural descriptions from examples. (Technical Report No. UIUCDCS-F-83-904). Urbana, IL: University of Illinois, Department of Computer Science.

Larson, J.B. (1977). Inductive inference in the variable-valued predicate logic system VL_{21}: Methodology and computer implementation (Technical Report 869). Doctoral dissertation, Department of Computer Science, University of Illinois, Urbana, IL.

Michalski, R.S. (1983). A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga,

Michalski, R.S., & Stepp, R.E. (1983). Learning from observation: Conceptual clustering. In R.S. Michalski, J.G. Carbonell & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga.

Mogensen, B. (1987). Goal-oriented conceptual clustering: The classifying attribute approach. Master's thesis, Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL.

Stepp, R.E., & Michalski, R.S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R.S. Michalski, J.G. Carbonell & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Whitehall, B.L. (1987). Incremental learning with INDUCE/NE (Working Paper 84). Urbana, IL: AI Research Group, University of Illinois Coordinated Science Laboratory.

Winston, P.H. (1975). Learning structural descriptions from examples. In P.H. Winston (Ed.), *The psychology of computer vision.* New York: McGraw Hill.