

AN ANALYSIS OF GENERALIZATION AS A SEARCH PROBLEM

Tom M. Mitchell

Computer Science Department
Rutgers University
New Brunswick, NJ 08903

The problem of concept learning, or forming a general description of a class of objects given a set of examples and non-examples, is viewed here as a search problem. Existing programs that generalize from examples are characterized in terms of the classes of search strategies that they employ. Several classes of search strategies are then analyzed and compared in terms of their relative capabilities and computational complexities.

1 Introduction

"Learning" is a broad term covering a wide range of processes. One process central to many kinds of learning is the process of generalization or concept learning; that is, characterizing a class of specific observations by abstracting the important features common to members of that class. *

This problem of generalizing from a set of training instances has been studied by many researchers over the last two decades (e.g., [1], [10], [13], [4], [2], [12], [3], [6]). The results so far have been tantalizing: partially successful programs have been written for problems ranging from learning fragments of spoken English to learning rules of chemical spectroscopy. But comparing and understanding alternate strategies has been difficult because of differences in data representations, terminology, and problem characteristics.

The goal of this paper is to compare alternate approaches to generalization in terms of a single framework. Toward this end, generalization is cast as a search problem, and alternate methods for generalization are characterized in terms of the search strategies that they employ. This characterization uncovers similarities among approaches, and leads to a comparison of relative capabilities and computational complexity of alternate approaches.

* An extended version of this paper is available as Rutgers Department of Computer Science technical report DCS-TR-78. This work has been supported in part by the National Institutes of Health under grant RR 00612-07 and by the Advanced Research Projects Agency under contract DAHC 15-73-C-0435.

2 The Problem

The class of concept learning, or generalization problems considered here may be characterized as follows:

Generalization Problem:

- Given:
1. A language in which to describe concepts.
 2. A set of positive and negative training instances of some "target concept" (concept to be learned).
 3. A matching predicate that matches concept descriptions to instances.

Determine: Concept descriptions within the provided language that are consistent with the presented training instances.

Here a concept description is considered to be consistent with a set of training instances if and only if it matches every positive instance and no negative instance in the set. This strict requirement of consistency is imposed in order to obtain concrete results in comparing alternate approaches. Although several of the approaches to be considered have been extended to deal with inconsistent training data, an analysis of performance in such cases is beyond the scope of this paper.

Generalization as defined above can conveniently be viewed as a search problem, in which the language of allowed concept descriptions defines

the domain of concepts that the program might learn, or the space of possible "solutions" to the concept learning problem. The program must examine this solution space, subject to constraints imposed by the training instances, to determine valid descriptions of the target concept.

2.1 The Partial Ordering

A key characteristic of the above generalization problem is that there is a useful structure to the solution space of possible concept descriptions. This structure is a partial ordering based upon the "more-specific-than" relation defined as follows*:

More-specific-than relation: Given two concept descriptions, CD1 and CD2, we say that CD1 is "more-specific-than" CD2 if and only if CD1 matches a proper subset of the instances that CD2 matches.

As an example, consider the following three concept descriptions which characterize classes of block structures:

CD1: "A structure including a green slab supported by two posts."

CD2: "A structure including a slab."

CD3: "A structure including a red slab."

Here, CD1 is more-specific-than CD2: any instance which matches CD1 must also match CD2. Similarly, CD3 is more-specific-than CD2. In contrast, CD3 and CD1 are not comparable descriptions in the partial ordering - neither is more specific than the other.

This more-specific-than relation imposes a general-to-specific partial ordering over the concept descriptions in the solution space of any generalization problem as defined above. It is important because it allows organizing the examination of the solution space in an efficient, complete manner. At the same time, it provides this paper with a basis for comparing alternate generalization strategies in a way that is independent of the particular concept description languages used.

* This ordering has been described previously for individual concept description languages [10], [5], [4], [12].

3 Model-driven and Data-driven Strategies

If concept learning is viewed as a search problem, then concept learning methods can be characterized in terms of the search strategies that they employ. Two broad classes of search strategies that have been employed for concept learning problems may be called model-driven and data-driven search strategies. These terms have been used to refer to procedures for examining spaces of possible hypotheses for a variety of problems outside concept learning [11], [9].

In a model-driven search (e.g., [2], [3]), hypotheses (concept descriptions) in the search are generated according to a predetermined model based upon prior knowledge of the problem. The generated hypotheses are then tested against the set of available data (training instances), to prune the search. In contrast, data-driven search uses the data prospectively to generate new hypotheses. Discrepancies between the current hypothesis and available data drive the generation of new hypotheses. In practice, model-driven strategies tend to consider all available training instances at each step to test the methodically generated hypotheses, whereas existing data-driven strategies for concept learning consider the training instances one at a time to generate new hypotheses.

The model-driven search performed by the RULEGEN portion of the Meta-DENDRAL program [2] evaluates the model-generated hypotheses in terms of their performance over the entire set of training instances. This strategy can accommodate quite severe errors in the training data because it judges the generated hypotheses by their performance over a large set of instances rather than making decisions based upon single training instances. This observed robustness of model-driven search with respect to data errors is consistent with the observations of other researchers considering hypothesis formation tasks outside concept learning [9], [11].

In contrast, existing data-driven search procedures (such as [13], [10], [5], [4], [12], [6]) base the generation of new hypotheses on differences between the current hypotheses and a single new training instance.

Because of this incremental use of the data, such strategies tend to be more sensitive to errors in individual training instances, while being better suited to adapting current results when new data becomes available.

An interesting combination of the advantages of model-driven and data-driven search procedures is found in the Meta-DENDRAL program. The RULEGEN portion of the program, mentioned above, conducts a coarse, model-driven search to form approximate rules of mass spectroscopy from highly unreliable training instances. These approximate rules are then refined by a data-driven strategy which conducts a more detailed search, and takes advantage of additional data [6]. Thus, the advantages of model-driven search for dealing with inconsistent data are blended with the advantages of data-driven search for incremental use of the data.

4 Three Data-Driven Strategies

The majority of concept learning programs reported in the literature employ data-driven search strategies. Although no two of these programs employ exactly the same strategy, it is informative to group them into classes whose members employ similar strategies and therefore possess similar performance characteristics. The aim of this section is not to compare alternate concept learning programs, but rather alternate classes of data-driven strategies that existing programs implement in various ways, for various concept description languages. We describe three such classes of search strategies here. The capabilities and efficiency of the classes are then compared.

4.1 Depth-first Search

One common data-driven strategy for examining the solution space of possible concept descriptions is depth-first search. Programs that can be characterized in this way include [13] and parts of the RULEMOD portion of the Meta-DENDRAL program [2]. In this strategy, a single concept description is chosen as the current best hypothesis for describing the identity of the target concept. This current hypothesis is then tested against each newly presented training instance, and altered as needed so that the resulting concept description is consistent with the new instance. Each such alteration yields a new current hypothesis, and corresponds to one step in a depth-first search through the solution space.

There are two awkward characteristics of this depth-first search strategy. First, each alteration to the current hypothesis must be tested to determine whether it is consistent with past training instances. Some systems (e.g., [13]) sacrifice assured consistency with past instances by not reexamining them when the current hypothesis is altered. Others (e.g., [2]) test past instances, and therefore require progressively larger computation time for each successive training instance. Second, once the program has determined a set of acceptable alterations to the current concept description, it must choose one of these as the new current hypothesis, and be prepared to backtrack if an incorrect choice has been made.

4.2 Specific-to-General Breadth-first Search

In contrast to depth-first search programs, programs which employ a breadth-first strategy maintain a set of several alternate hypotheses. Systems which fall into this class include those reported in [10], [4], and [12]. Each of these programs takes advantage of the general-to-specific partial ordering of concept descriptions to efficiently organize the breadth-first search. Starting with the most specific concept descriptions consistent with the first positive instance, the search is organized to follow the branches of the partial ordering so that progressively more general concept descriptions are considered each time the current set must be modified. The set of alternate plausible hypotheses computed by this specific-to-general breadth-first search is the set (which we shall call S) of maximally specific concept descriptions consistent with the observed training instances; that is

$$S = \{s \mid s \text{ is a concept description that is consistent with the observed instances, and there is no concept description which is both more specific than } s, \text{ and consistent with the observed instances}\}$$

One advantage of this strategy over depth-first search stems from the fact that the set S represents a threshold in the solution space. Concept descriptions that are more specific than this threshold are not consistent with all the observed positive instances, whereas those more general than this threshold are. Thus, when a concept description in S must be revised, it must be made more general, and the revision need not be tested for consistency with past positive instances. It must still, however, be tested against previous negative instances.

4.3 Version Space Strategy

The version space strategy for examining the solution space entails representing and revising the set of all hypotheses consistent with the observed training instances. This set of concept descriptions forms a convex set with respect to the partial ordering, and is referred to as the version space of the target concept (since it contains all plausible versions of the emerging concept). This strategy begins by representing the version space of concept descriptions consistent with the first positive training instance. Concept descriptions found inconsistent with subsequent instances are then eliminated from consideration. Programs that implement this strategy for two different concept description languages are described in [6].

The version space approach is feasible because the general-to-specific ordering of concept descriptions allows a compact representation for version spaces. In particular, a version space can be represented* by two sets of concept descriptions: the set S as defined above, and a complementary set G, of maximally general concept descriptions consistent with the observed instances:

$G = \{g \mid g \text{ is consistent with the observed instances, and there is no concept description which is both more general than } g, \text{ and consistent with the instances}\}$

The set G can be computed by conducting a breadth-first search analogous to that for computing S, but proceeding instead from general to specific concept descriptions. Since the sets S and G together delimit the version space, the version space strategy can be viewed as an extension of the above breadth-first search strategy into a bi-directional search.

The power of the version space strategy lies in the fact that the set G summarizes the information from the negative training instances that bounds how general the "correct" concept description might be, while the set S summarizes the information from positive instances that limits how specific it might be. Thus, testing whether a given concept description is more specific than some element of G and more general

than some element in S, is logically equivalent to testing whether it is consistent with all observed training instances.

Since the sets S and G summarize the information in the training instances, the version space strategy does not require reexamining or saving previously processed training instances. This method of delimiting the version space leads to increased capabilities as well as efficiency.

4.4 Capabilities

In comparing alternate strategies for concept learning, the central issue is relative capabilities. What are the limits of each approach in terms of questions to which it can provide answers? The major differences in capabilities among the above three data-driven strategies derive from the number of plausible concept descriptions carried along at each step, and from the use of the partial ordering in guiding the search. Below we consider two desirable capabilities: (1) the ability to determine when a concept has been fully learned, and to use incompletely learned concepts in a reasonable manner, and (2) the ability of the learning program to direct the presentation of training instances in an intelligent manner.

4.4.1 Using Imprecisely Learned Concepts

Recognizing the degree to which a concept has been learned, and using incompletely learned concepts in a reasonable manner are essential capabilities for practical learning systems. It is rare that the available training instances precisely describe the target concept.

The version space strategy provides a straightforward method for detecting the point at which a concept is precisely described by a set of training instances, with respect to the given concept description language. The concept is completely learned (assuming reliable training instances, and a sufficient concept description language) if and only if the version space contains exactly one concept description; that is, if S and G contain one and the same concept description. In contrast, it is difficult to recognize this condition when maintaining only a single current concept description, as with the depth-first search strategy.

In the case where a partially learned concept must be used, the sets S and G again provide

* The version space is "represented" in the sense that it is possible to generate and recognize concept descriptions in the version space by examining the representation.

useful information. Consider the case in which the version space derived from the provided training instances contains a range of plausible concept descriptions, no additional training instances are available, and a new instance is presented to be classified.

In such a case, although the exact identity of the target concept is not fully determined by the training instances, it is known that the correct description of the target concept lies somewhere within the version space delimited by S and G. Therefore, if the new instance matches every concept description in the version space (equivalently, if it matches each element of S), then it can be classified as a positive instance with the same certainty as if the concept had been uniquely described by the training instances. Alternately, if the instance matches no concept description in the version space (i.e., it matches no element of G), then it is certain that the instance does not match any description of the target concept that would be chosen by examining additional training instances. Thus, for such instances it is possible to use the partially learned concept to obtain classifications that are as reliable as if the learned concept had been completely described by the training instances.

In contrast, instances that match some, but not all concept descriptions in the version space cannot be reliably classified until further training instances are available. Of course, by considering outside knowledge and examining the proportion of concept descriptions in the version space which match the instance, one might still estimate the classification of such instances.

Because the specific-to-general breadth-first strategy also computes the set S, this strategy allows reliably classifying the same positive instances as the version space strategy. Since it does not compute the set G, however, it cannot distinguish between instances which the version space strategy would reliably classify as negative instances, and those which cannot be reliably classified.

4.4.2 Selecting Future Training Instances

A further capability afforded by computing the sets S and G is the selection of informative new training instances. Consider the following problem: After processing some sequence of training instances, a program is provided a set of new instances, without their classifications

as positive or negative instances, and is allowed to request the correct classification of any one of them. The instance whose classification will be most informative (on the average) is the one that comes closest to matching one half of the concept descriptions in the version space. Regardless of its classification, such an instance would allow rejecting one half of the currently plausible concept descriptions. Thus, by testing each instance to determine what proportion of the concept descriptions in the version space it matches, the most informative training instance can be selected (a more complete discussion of this issue may be found in [6] and [8]).

The breadth-first strategy also provides some information for selecting new training instances. The strategy of selecting instances which match half the concept descriptions in the computed set S is reasonable, although less complete than the strategy which takes into account the entire version space.

4.5 Complexity and Efficiency

The overall efficiency of each approach is determined by a complex of factors, including the order of presentation of training instances, the chosen concept description language and the branching of the associated partial ordering, the cost of matching concept descriptions to training instances, and the amount of space needed to store concept descriptions and observed instances.

A complete analysis is beyond the scope of this paper, but it is possible to characterize the rates of growth of time and space requirements as a function of the number of training instances, under reasonable assumptions. Under the assumption that positive and negative instances are distributed uniformly throughout the sequence of training instances, bounds on the time and space complexity of the data-driven strategies described earlier are summarized in table 1. Here p indicates the number of positive training instances, n indicates the number of negative training instances, |S| indicates the largest size obtained by the set S, and |G| represents the largest size obtained by the set G.

Strategy	Processing Time	Storage Space
Depth-first search	$O(pn)$	$O(p+n)$
Specific-to-general, Breadth-first	$O(S pn)$	$O(n+ S)$
Version space strategy	$O(S G (p+n))$	$O(S + G)$

Table 1: Bounds on processing time and maximum storage costs.

The differences in Table 1 reflect differences in the way in which revised hypotheses are tested for consistency with past instances. In the depth-first strategy this must be done by going back to the instances themselves. In the specific-to-general breadth first strategy, only past negative instances need be reexamined, since S summarizes the constraints imposed by past positive instances. The tradeoff is one of calculating and using S and G , versus returning to the original data. A justification of these results is given in [7].

In interpreting the above results it is important to know how the sizes of the sets S and G vary over the training sequence. For the two concept description languages for which the version space strategy has been implemented, these sets were observed to first grow in size, then level off, and finally decrease in size as the version space collapsed toward the correct description of the target concept. An interesting problem is to characterize the behavior of these sets for various forms of concept description languages, and study ways in which careful presentation of training instances can reduce their size.

5 Summary and Conclusions

The problem of concept learning, or generalization, can be viewed as a search problem involving a large solution space of possible concept descriptions. Concept learning then involves examining this space under constraints imposed by the training instances (data constraints), as well as prior knowledge and expectations (model-based constraints). In this light, it is informative to characterize

alternate approaches to concept learning in terms of the strategy that each employs in examining this solution space.

Two classes of strategies, data-driven and model-driven, differ in terms of which of these two kinds of constraints is used to drive the search. General characteristics of these two classes that appear in concept learning problems correlate with characteristics noted by others in hypothesis formation problems in signal analysis and molecular genetics.

A general-to-specific partial ordering gives structure to the search space for the class of generalization problems considered here. Several data-directed strategies for concept learning are described and compared in terms of the way in which they organize the search relative to this partial ordering. This examination leads to a comparison of relative capabilities and computational complexity.

6 Acknowledgments

Thanks to the following people who commented carefully on various drafts of these ideas: Bruce Buchanan, John Burge, Ed Feigenbaum, Cordell Green, Rick Hayes-Roth, Abe Lockman, and Richard Pantell.

References

1. J.S. Bruner, et al., *A Study of Thinking*. Wiley, 1956.
2. B.G. Buchanan and T.M. Mitchell, Model-directed learning of production rules. In *Pattern-Directed Inference Systems* (Waterman and Hayes-Roth, Eds.), Academic Press, 1978.
3. T.G. Dietterich and R.S. Michalski, Learning and generalization of characteristic descriptions. *IJCAI6*, Tokyo, 1979.
4. F. Hayes-Roth, Schematic classification problems and their solution. *Pattern Recognition*, 6, pp. 105-113 (1974).
5. R.S. Michalski, AQVAL/1 - Computer implementation of a variable valued logic system VLI and examples of its application to pattern recognition. *IJCPRI*, 1973, pp. 3-17.
6. T.M. Mitchell, Version Spaces: An approach to concept learning. Ph.D. thesis, Stanford University, December, 1978.
7. T.M. Mitchell, An Analysis of Generalization as a Search Problem. Rutgers Computer Science Technical Report DCS-TR-78. January, 1979.
8. T.M. Mitchell, Generalization Using Version Spaces. In preparation.
9. H.P. Nil and E.A. Feigenbaum, Rule-based understanding of signals. In *Pattern-Directed Inference Systems* (Waterman and Hayes-Roth, Eds.), Academic Press, 1978.
10. G.D. Plotkin, A note on inductive generalization, *Machining Intelligence 5* (Meltzer and Michie, Eds.), Edinburgh University Press, 1970, pp. 153-163.
11. M.J. Stefik, Inferring DNA structures from segmentation data. *Artificial Intelligence*, 11, August, 1978.
12. S.A. Vere, Inductive learning of relational productions. *Pattern-Directed Inference Systems* (Waterman and Hayes-Roth, Eds.), Academic Press, 1978.
13. P.H. Winston, (Ed.), *The Psychology of Computer Vision*, McGraw-Hill, 1975.