2.4 Theory

The Need for Biases in Learning Generalizations

Tom M. Mitchell
Computer Science Department
Rutgers University
New Brunswick, NJ 08904

May, 1980

Abstract

The ability to make an appropriate "inductive leap" when generalizing from a small set of training instances is possible only under a priori biases for choosing an appropriate generalization out of the many possible. Understanding the origins and justification of such biases is critical to further progress in the field of machine learning. The notion of an UNbiased learner is defined, then the notion of bias, its usefulness, and some classes of justifiable biases are considered.

Originally published as a Rutgers Technical Report.

1 Introduction

Learning involves the ability to generalize from past experience in order to deal with new situations that are "related to" this experience. The inductive leap needed to deal with new situations seems to be possible only under certain biases for choosing one generalization of the situation over another. This paper defines precisely the notion of bias in generalization problems, then shows that biases are necessary for the inductive leap. Classes of justifiable biases are considered, and the relationship between bias and domain-independence is considered.

We restrict the scope of this discussion to the problem of generalizing from training instances, defined as follows:

<u>The Generalization Problem</u>

<u>Given:</u>

1. Language of instances.
2. Language of generalizations.
3. Matching predicate for matching generalizations to instances.
4. Sets of positive and negative training instances.

<u>Determine:</u>

Generalization(s) consistent with the training instances.

As a concrete example of the above generalization problem, consider the task addressed by Winston's program for learning classes of block structures [Winston, 1975]. Here, the language of instances is the representation used to describe example block structures. The language of generalizations is the language in which learned concepts (e.g., arch, tower) are described. The matching predicate specifies whether a given generalization applies to a given instance (e.g., whether the inferred description of an arch is satisfied by a specific block structure).

This paper addresses a deep difficulty with the generalization problem as defined above: If consistency with the training instances is taken as the sole determiner of appropriate generalizations, then a program can never make the inductive leap necessary to classify instances beyond those it has observed. Only if the program has other sources of information, or biases for choosing one generalization over the other, can it non-arbitrarily classify instances beyond those in the training set.

In this paper, we use the term <u>bias</u> to refer to <u>any basis for choosing one generalization over another, other than strict consistency with the observed training instances.</u>

2 What is an UNbiased Generalizer?

If generalization is the problem of guessing the class of instances to which the positive training instances belong, then an unbiased generalizer is one that makes no a priori assumptions about which classes of instances are most likely, but bases all its choices on the observed data. Two common sources of bias in existing learning systems are (1) the generalization language is not capable of expressing all possible classes of instances, and (2) the generalization procedure that searches through the space of expressible generalizations is itself biased.

2.1 An Unbiased Generalization Language

In considering bias in the generalization language, it is useful to view each generalization as denoting the set of instances that it matches. In figure 1, for example, g1 and g2 are two generalizations expressible in some generalization language, and each matches a different subset of the instances.

Relative to a given language of instances, an unbiased generalization language is then one which allows describing every possible subset of these instances. In short, an unbiased generalization language corresponds to the power set of the given instance language.

The impact of using a biased generalization language is clear: each subset of instances for which there is no expressible generalization is a concept that could be presented to the program, but which the program will be unable to describe and therefore unable to learn. If it is possible to know ahead of time that certain subsets of instances are irrelevant, then it may be useful to leave these out of the generalization language, in order to simplify the learning problem. For example, generalization languages that allow only conjunctions of constraints on features of the instance (e.g., [Winston, 1975], [Buchanan, 1978]) introduce a strong bias of this kind, which reduces considerably the complexity of the generalization problem.

The strength of the bias introduced by generalization languages restricted to conjunctive constraints on features, is illustrated by a simple example. Consider an instance language of binary feature vectors containing 5 features, and a generalization language that allows constraining each feature value to be 1, 0, or "don't care". Some simple arithmetic shows that only about one out of every 10^7 subsets of instances is expressible in the generalization language! This proportion worsens quickly as the number of features and the number of allowed values per feature increases.

2.2 An Unbiased Generalization Procedure

The generalization procedure searches for expressible generalizations that denote sets of instances, each of which includes all of the positive but none of the negative training instances. Of course, there may be many such generalizations. In figure 1, for instance, if the observed positive instances are contained in the intersection of the instance sets of g1 and g2, and the observed negative instances are outside both sets, then both g1 and g2 are consistent with the observed instances.

An unbiased generalization procedure is one which shows no preference for one expressible generalization over another, except on the basis of consistency with the training instances.

Following [Mitchell, 1977], we define the *version space* relative to a particular generalization language, and a given set of training instances, as the set of *all* expressible generalizations consistent with the training instances. Then an unbiased generalization procedure must compute the version space relative to the observed training instances, and the provided generalization language. Such a generalization procedure is described in [Mitchell, 1978], and has been implemented as part of the Meta-DENDRAL program [Buchanan, 1978].

In order to consider the consequences of an unbiased generalization procedure, it is necessary to consider how a computed version space can be used to classify subsequent instances as either positive or negative. An unbiased classification method classifies the new instance as a positive instance if and only if *every* generalization in the version space matches it. The instance is classified as a negative instance if and only if *no* generalization in the version space matches it. If some, but not all, of the generalizations in the version space match the instance, then the instance cannot be classified with certainty. The program could, however, give an estimated classification based upon the proportion of generalizations within the version space, that match and do not match the instance.

3 *The Futility of Removing Biases*

For a generalization system to be unbiased - for it to consider equally all possible subsets of instances as the possible identity of the class being learned - it must employ an unbiased generalization language, and compute the version space relative to that language and the presented training instances.

Although removing all biases from a generalization system may seem to be a desirable goal, in fact the result is nearly useless. An unbiased learning system's ability to classify new instances is no better than if it simply stored all the training instances and performed a lookup when asked to classify a subsequent instance.

To see that this is the case, consider the procedure described above for using the computed version space to classify new instances. For an unbiased generalization language, the new instance will match every generalization consistent with the observed instances if and only if it is identical to one of the observed positive instances. Similarly, to be classified as a negative instance, the new instance must be identical to one of the observed negative instances. Furthermore, any instance that has not yet appeared as a training instance will match exactly half the generalizations in the version space. As a result, even the scheme described above for estimating the classification will fail to produce useful results.

In retrospect, it is not surprising that an unbiased generalization system cannot make classifications of instances other than the training instances. An unbiased system is one whose inferences logically follow from the training instances, whereas classifications of new instances do not logically follow from the classifications of the training instances.

4 Useful Classes of Biases

There is an important conclusion to the above discussion: If totally unbiased generalization systems are incapable of making the inductive leap to characterize new instances, then the power of a generalization system follows directly from its biases - from decisions based on criteria other than consistency with the training instances. Therefore, progress toward understanding learning mechanisms depends upon understanding the sources of, and justification for, various biases. This section classifies certain kinds of biases that have been used by learning programs.

Factual knowledge of the domain. In learning generalizations for a particular purpose, it may be possible to limit the generalizations considered, by appealing to knowledge about the task domain. The Meta-DENDRAL program [Buchanan, 1978] forms general rules that characterize molecular bonds that fragment in a mass spectrometer. Here, general knowledge of the domain, such as "double bonds rarely break", can be used to constrain the search for appropriate generalizations. Similarly, in a program that learns the rules of baseball [Soloway, 1978], general knowledge about competitive games constrains the number of generalizations considered. This kind of prior knowledge can provide a strong, justifiable constraint on the generalizations considered. In such a case, the goal of the generalization system becomes "determine generalizations consistent with the training instances, _and with other known facts about the task domain_".

Intended use of the learned generalizations. Knowledge of the intended use of learned generalizations can provide a strong bias for learning. As a simple example, if the intended use of the learned generalizations involves a much higher cost for incorrect positive than for incorrect negative classifications, then the learning program should prefer more specific generalizations over more general alternatives that are consistent with the same training data.

Knowledge about the source of training data. Knowledge about the source of the training instances can also provide a useful constraint on learning. For instance, in learning from a human teacher, we seem to take advantage of many assumptions about the existence of an organized curriculum to constrain our search for appropriate generalizations. In an organized curriculum, our attention is carefully focussed on particular features of instances, in a way that removes ambiguity about which of the possible generalizations is most appropriate. The use of this kind of knowledge to constrain generalization is currently under study in both CAI and machine learning research. For example, [Smith, D., 1980] and [Smith, R., 1980], address issues such as shared assumptions between the teacher and learner, what constitutes a good curriculum, and how the student's generalization strategy can take advantage of cues in the curriculum.

Bias toward simplicity and generality. One bias that humans seem to use is a bias toward simple, general explanations. Some learning programs also incorporate this bias. For example, the RULEGEN portion of the Meta-DENDRAL program searches through the space of possible generalizations, beginning with the most general possible, and considering more specific generalizations only when the training data indicates that more general explanations are unacceptable. This bias toward simplicity and generality has been discussed in the philosophical literature, and its justification has been debated there.

Analogy with previously learned generalizations. If a system is learning a collection of related concepts, or generalizations, then a possible constraint on generalizing any one of them is to consider successful generalizations of others. For example, consider the task of learning structural descriptions of blocks-world objects, such as "arch", "tower", etc. After learning several concepts, the learned descriptions may reveal that certain features are more significant for describing this class of concepts than are others. For example, if the generalization language contained features such as "shape", "color", and "age" of each block in the structure, the system may notice that "age" is rarely a relevant feature for characterizing structural classes, and may develop a bias in favor of other features. The justification for this learned bias must be based upon some presumed similarity in the intended use of the concepts being learned.

5 Conclusions

Unbiased generalization programs that use consistency with the training instances as their only source of information, cannot outperform programs that use rote learning. Additional information or biases are therefore critical to the ability to classify instances that are not identical to the training instances. This fact has significant implications for research on machine learning.

If biases and initial knowledge are at the heart of the ability to generalize beyond observed data, then efforts to study machine learning must focus on the combined use of prior knowledge, biases, and observation in guiding the learning process. It would be wise to make the biases and their use in controlling learning just as explicit as past research has made the observations and their use.

6 Acknowledgments

The following people have provided thoughtful comments on various drafts of this paper, and have contributed their own ideas: Saul Amarel, George Drastal, N.S. Sridharan, and Paul Utgoff. This work was supported by NIH grant RR-643-09, and by an award from the Rutgers Research Council.

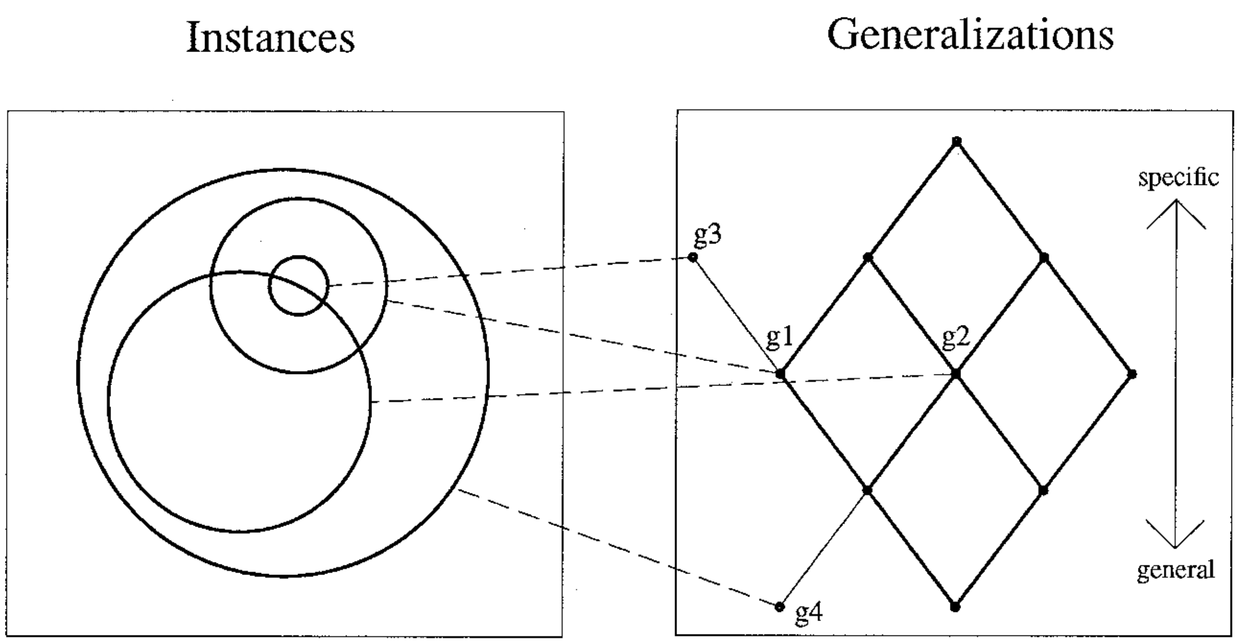

Figure 1. Relationships among Instances and Generalizations

(This figure was missing from the original publication and added in 1990.)

References

Buchanan B. G., and T. M. Mitchell, Model-directed learning of production rules, In Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

Mitchell, T. M., Version Spaces: A candidate elimination approach to rule learning. IJCAI5, MIT, Cambridge, MA, August 1977, pp. 305-310.

Mitchell, T. M., Version Spaces: An approach to concept learning. Ph.D. thesis, Stanford University, December, 1978. Also Stanford CS report STAN-CS-78-711, HPP-79-2.

Smith, D.E., FOCUSER: a strategic interaction paradigm for language acquisition. submitted to 1st AAAI, 1980.

Smith, R.L. Jr., Modelling student acquisition of problem solving skills. Submitted to 1st AAAI, 1980.

Soloway E. M., and E. M. Riseman, Knowledge-directed learning. Pattern-Directed Inference Systems (D.A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

Winston, P. H., (Ed.), The Psychology of Computer Vision, McGraw-Hill, New York, 1975.