BACON: A PRODUCTION SYSTEM THAT DISCOVERS EMPIRICAL LAWS

Patrick W. Langley
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Introduction

In recent years researchers have produced a number of programs capable of scientific-like behavior. Each of the systems, DENDRAL(Feigenbaum and Lederberg, 1971), meta-DENDRAL(Buchanan, Feigenbaum and Sridharan, 1972), MYCIN (Davis, Buchanan and Shortliffe, 1975) and AM (Lenat, 1976) could arrive at rules that explained observed data. In this paper I discuss another system, BACON, which discovers simple empirical laws like those found by early physicists.

Task Environment

BACON's task environment is an artificial two-dimensional universe in which labeled objects interact. The program has direct access to a set of primitive attributes (such as the x-coordinate, color and velocity) specified by the programmer. The values of these attributes change across objects and across time according to a set of laws determined by the programmer. Such a law might be that the third differential of each object's distance from the center of the universe is a constant (with a different constant for each object).

Program Structure

BACON is implemnted in the production sytem language OPS (Forgy and McDermott, 1976). The production system framework's advantage in this task is that it allows one to write a small set of general "regularity detectors". These search in parallel through the data the program has collected (in the form of attribute-object-time-value 4-tuples) and which are still in working memory. If one of the described regularities is present in the data, it "leaps out" at the system and appropriate action is taken. If the regularity is a constancy, a generalization across the relevant dimension (time, objects, or both) is made and tested. If the value of an attribute consistently increases or decreases over time, a new attribute defined in terms of the old attribute's change over time is constructed (e.g., acceleration is constructed from velocity) and considered further. If the value of one attribute goes up as the value of another goes down, a new attribute defined as their product is considered. Eventually, a higher level attribute employing all the attributes mentioned in the programmer's law is constructed and discovered to have a constant value; this is BACON's version of the law.

Along the way, whenever some generalization about the value of an attribute is confirmed, BACON adds productions that enable prediction of the value and, if the attribute is nonprimitive, prediction of some of its component's values In terms of other component's values. The system can also qualify its hypotheses. If counterexamples are found to a generalization, restrictions can be added in the form of exceptions if the generalization was across objects and in the form of exceptions and upper and lower bounds if the generalization was across time. In fact, once an hypothesis is formed, it is never rejected. It is always stored in the permanent production memory, though perhaps in greatly qualified form.

Initially, the data collection process is unintelligent. Attributes are generated randomly and their values examined across a number of objects and times. However, once regularities have been detected, these regularities serve to direct the search.

Conclusions

In summary, BACON discovers empirical laws by using two complementary techniques. The first is the detection of regularities in data, for which production system formalisms seem ideally suited. This technique leads to useful generalizations, but it also directs the second technique, the construction of new attributes in terms of more primitive ones. Together the two lead the search for more data which could lead to further generalizations.

As implemented, BACON has a number of limitations as a law discoverer. First, it cannot formulate conditional laws, in which the conditions are restrictions on the values of related attributes; however, it seems plausible that the regularity detection technique could be used here as well. Second, the system has special knowledge that time and objects are useful dimensions to generalize over; a more general discovery program would be able to do without such help. Finally, the current system cannot deal with noise in its data; however, a minor change in the OPS pattern matching routine for numbers would deal with this. In conclusion, although the present system has some drawbacks, it seems easily extendable and shows promise of more interesting things to come.

References

Buchanan, B.C., Feigenbaum, E.B, and Sridharan. Heuristic theory formation.In D.Michie (ed.), Machine Intelligence 7. New York: American Elsevier Publishing Co.,1972,pp.267-290.

Davis, R., Buchanan, B. and Shortliffe, E. Production rules as a representation for a knowledge-based representation program. Stanford AT Laboratory Memo AIM-266, 1975.

Feigenbaum, E.B. and Lederberg, J. On generality and problem solving: A case study using the DENDRAL program. In B.Meltzer and D.Michie (eds.), Machine Intelligence 6. New York: American Elsevier Publishing Co., 1971, pp.165-190.

Forgy, C. and McDermott, J. OPS Manual. Pittsburgh, Pa.: Carnegie-Mellon University, Department of Computer Science, 1976.

Lenat, D.B. AM: An artificial intelligence approach to discovery in mathematics as heuristic search. Unpublished doctoral dissertation, Department of Computer Science, Stanford University, 1976.