ADJUSTING BIAS IN CONCEPT LEARNING *

Paul E. Utgoff**

Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

ABSTRACT

We approach concept learning as a heuristic search through a space of concepts for a concept that satisfies the learning task at hand The heuristics represent bias that the concept learning program employs when forming an inductive generalization. We present a model of bias adjustment and report our experience with an implementation of the model.

I ROLE OF BIAS

A major objective in research on inductive concept learning is creation of a program that can accept training data, apply knowledge, and form inductive hypotheses of the concept, all without human intervention. The learning program searches a space of hypotheses for those that are consistent with the observed examples, and which classify the unobserved instances as indicated by heuristics. The heuristics, which we call *bias,* determine the inductive generalizations that the program will form, given some set of training examples. The initial bias may be appropriate for one learning task, yet inappropriate for another. In many concept learning programs to date, e.g. (Michalski 83), (Mitchell 77), the search for appropriate bias is done by hand. We examine an approach to mechanizing that search.

Vere (Vere 80) and Lenat (Lenat 83) have programs that adjust their bias. Vere uses the set difference operator to construct a new description C = A-B when no other consistent description is available. Lenat's EURISKO learns heuristics that lead the program to discover interesting concepts. EURISKO is an advance from Lenat's AM where the search for appropriate bias was done by hand.

II AN APPROACH TO ADJUSTING BIAS

We represent bias as a restricted search space of concepts that we call the *concept description language.* We use Mitchell's Candidate Elimination Algorithm (Mitchell 77) to maintain a version space of all concept descriptions in the concept description language that are consistent with the training instances. When the trainer supplies a positive instance, all concept descriptions that exclude the instance are refuted and removed from the version space. Similarly, when the trainer supplies a negative instance, all concept descriptions that include the instance are refuted and removed from the version space. As shown in Figure 1, as the version space shrinks, the complementary set of refuted hypotheses grows. If the version space becomes empty, then all descriptions in the concept description language have been refuted. A refuted concept description language indicates a refuted bias, and is sufficient justification for adjusting bias. We do not yet know stronger justifications The role of bias adjustment is to enlarge the concept description language so that the version space becomes nonempty and search can continue. It may be that certain concepts are not describable within a particular formalism. If so, shift to a more appropriate formalism may be necessary.

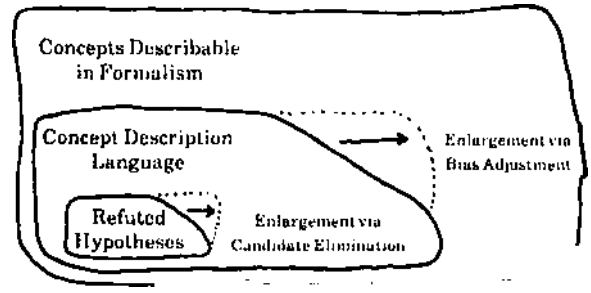

Figure 1 Model of Bias

We approach bias adjustment as a three step process:

1. Determine a membership specification for the new concept.
2. Translate the membership specification into a description that uses the formalism of the concept description language.
3. Assimilate the new concept description into the hierarchy of the concept description language.

A *membership specification* is itself a concept description, but is expressed in an unbiased language so that the heuristics for determining the specification are not compromised by what is expressible. Because the membership specification may be expressed in an alternate formalism, we require the ability to translate the membership specification into a concept description in the formalism of the concept description language. The translation step may require creation of new descriptive terms that correspond to membership specifications that are not describable in the current concept description language. A new concept must then be assimilated into the concept description language by defining its location in the hierarchy of descriptions.

III AN EXAMPLE

We have implemented a program that models bias adjustment as part of the LEX program (Mitchell 83) LEX is a concept learning program that learns heuristics that suggest whether a given operator should be applied to a given problem state. Each operator heuristic is represented as the concept "set of states to which this operator should be applied."

LEX currently learns heuristics in the domain of integral calculus. The program is initially given a set of problem-solving operators. Each operator has a domain of applicability, a rewrite rule, and a range of producible states. For each operator, the legal domain of applicability describes states to which the

* This work was supported by National Science Foundation Grant GMCS80-08889, Rutgers University Laboratory for Computer Science Research, and Siemens Research and Technology Laboratories.

** Current address is Siemens Research and Technology Laboratories, 105 College Road East, Princeton, New Jersey 08540.

operator can be applied. For each operator, the heuristic domain of applicability describes states to which the operator should be applied.

LEX solves a problem by producing a state that contains no integral. The solution tree is then criticized Each operator application along the minimum solution path is labelled as a positive instance showing a state to which the given operator should be applied. Each operator application that leads away from the minimum solution path is labelled as a negative instance showing a state to which the given operator should not be applied. The training instances are then used to update the heuristics being learned for the operators For each operator, Mitchell's Candidate Elimination Algorithm is used to maintain a version space of all candidate versions of the heuristic that have not yet been refuted If a version space becomes empty, then the bias adjustment module is invoked

A A Heuristic for Membership

Deciding the membership of a new concept is the essence of induction. Which instances should be included, and which should be excluded? We employ a heuristic:

> The domain of an operator sequence that leads to a solution should be describable in the concept description language.

We choose this heuristic because of our a priori knowledge that the operator heuristic we hope to learn will be the disjunction of all useful sequences that start with that operator To describe the disjunction, we can start by being able to describe the disjuncts To compute a specification of the domain of an operator sequence that produces a state in a given range, in this case the set of solved problems, we apply a deduction procedure that we call *constraint back-propagation.*

Constraint back-propagation is a procedure for deducing the domain of a macrooperator that produces some constrained range of states. Consider an operator sequence of length 1 that uses operator Op,, as shown in Figure 2. If we constrain.the range of the Op, to some set of states RCRange(Op,), and then apply the inverse operator Op^{1}; to R, then we deduce a constrained set DC I)omain(Op_{1},) such that Op_{i}(D) = R. For example, an operator r->lr-2] has the set of real numbers as its domain and also as its range. If the domain is constrained to the set of integers, then the range is constrained to the set of even integers.

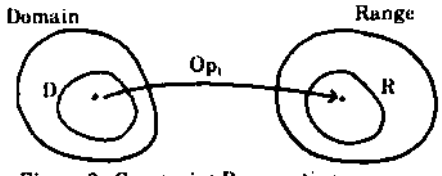

Figure 2 - Constraint Propagation

If we know that Op_1 can produce a state that is a solved problem, we can compute Op_i(IntersecUSolvedProblems, Range(Opj))) to deduce a constrained set D. D is then the set of states to which application of Op_1 will produce a solved problem. For operator sequences of length greater than 1, the deduction step is simply applied sequentially to the result of the previous deduction.

B. A Translation Method

A nested expression of intersections and inverse-operator applications, as shown above, is a specification of concept membership, but is not useful as a concept description because operator applications are required to evaluate whether an instance matches the description. The nested expression must be translated from a membership specification into a structural concept description that uses the formalism of the concept description language. Our translation method is to evaluate the membership specification To let this work, we have defined a description of the set of solved problems, an Intersection function, and inverse operator definitions. New concepts are created and assimilated during evaluation of the membership specification,.

When an operator only restructures a state via a simple rewrite rule, e.g a-b-»b-a, creation of new concepts is not required because the range of the operator is expressible by a description in the concept description language. When an operator requires a computation, e.g. multiplication by 2, the range may be describable in the operator language, but not the concept description language. For example, consider an operator r->[r-2]. The [r-2] indicates that the bound value of r should be multiplied by 2, and the result used in the rewritten state If an operator is to be applied to a set of states, as with constraint back-propagation, then correct specification of the operator's domain and range becomes very important. Application of r->|r-2] to the set of integers k produces a description f k-2 J. This is a description in the operator language, not the concept description language!

Our method of translating a description implied by a computation y = f(x) to a term in the concept description language is to first create a definition {yl(A(y)(match f-'(y) x))} and then search for a term with the same definition If a term with that definition is not found, then a new term is created and defined as per the constructed definition, For example, the definition that corresponds to |k-2] is {yl(A(y)(match y/2 'k))}(which we could associate with a new term "even "

C An Assimilation Method

Assimilation is the process of inserting a new concept description into the hierarchy of description in the concept description language Assimilation is necessary so that we can use the match predicate to evaluate whether one concept is more specific or equal to another.

Our present method is imprecise. In general, if we are to assimilate z = {yl(A(y)(match f^{1} y) x))}, then we make z a specialization of y. This method is correct only in the sense that no false match relation is asserted. For example, our method may assert that "even integers" is a specialization of "real numbers", but fail to assert that "even integers" is also a specialization of "integers". At the same time, the method does indicate y as a most general bound on where z should be assimilated. If the set of possible assimilations is small, we would expect that a mechanical procedure could prune these by empirical refutation, but we have not pursued this. Below we show an example of deducing the correct assimilation

D Experiments

We present three experiments with the implementation. Several problems came to light.

1. Experiment #1

As in (Utgoff 82), LEX found a solution for $Jcos^{7}(x)\,dx$ similar to that shown in Figure 3. LEX was then given $Jcos^{5}(x)dx$, for which the same solution sequence, and $/cos^{6}(x)dx$ for which the same solution sequence did not work because the polynomial in the integrand was not an integer. Because the concept description language did not contain a description that included $Jcos^{7}(x)dx$ and $Jcos^{5}(x)dx$ and excluded $Jcos^{6}(x)$-dx, the heuristic for Op51 could not be described. Bias adjustment was invoked, and constraint back-propagation was expected to proceed as shown in Figure 3. The implementation was successful for the last three propagation steps through $Op^{1}52$, Op-150, and $Op^{1}51$. Back-propagation through $Op^{1}13$ failed because our formalism, a modified context-free grammar, does not permit specifying that whatever matches f be the derivative of whatever matches f. This example shows that the formalism of the concept description language also implies bias. The ability to identify the kind of bias that is associated with a given formalism is an important open problem.

Back-propagation of the set of integers k through the exponent in Op 150 caused a new description {xI(A(x)(match *xI2* 'k))} to be constructed, the set of even integers. Because the operator could divide any real exponent by 2, the constructed set description was assimilated as a specialization of the set of real numbers. It would have been better to assimilate the set of even

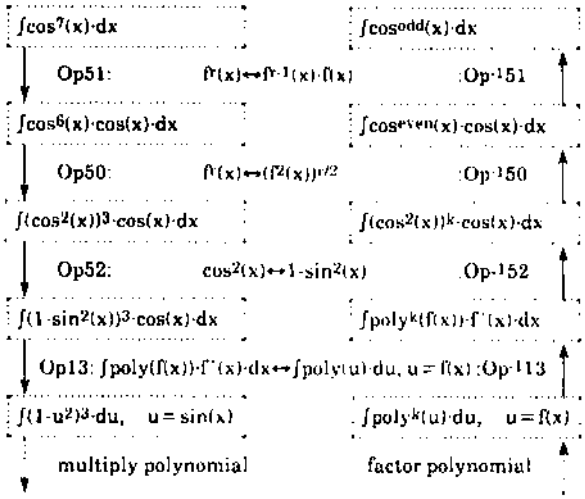

Figure 3 - Solution and propagation for $\int \cos^7(x) \cdot dx$

integers as a specialization of the set of integers. Precise assimilation could have been determined with the following reasoning: An alternate definition of even integers is {y|(∃x∈Integers)y=2·x)}. Because x and 2 are integers, and because we know that integers are closed under multiplication, we prove that every y is also an integer. The critical piece of knowledge for the proof is that integers are closed under multiplication. We have not implemented a mechanism for executing such proofs. Proposing and then proving subset relationships is an open problem of great interest. The backward constraint propagation method is useful for deducing which conjectures are worth proving.

2 Experiment #2

In this experiment, the set of real numbers "r" was to be back-propagated through Op22 as shown in Figure 4 . The result of the back-propagation step was $r_4 \cdot (r_3/r_4)$. Because this set was describable by an existing set description, it was not necessary to add any new descriptions to the search space.

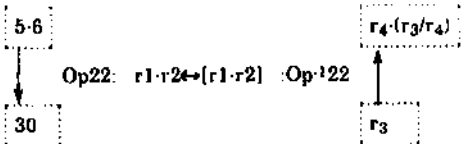

Figure 4 - Evaluating Product with Op22

Nevertheless, the back-propagation failed because the deduced set $r_4(r3/r_4)$ did not satisfy a safety check requiring that a deduced set, in this case r_4-(r_3/r_4), match the state that occurred in the original solution sequence, in this case 5-6. Although r_4 matched 5, r_3/r_4 did not match 6. The concept description language did not include knowledge that r_3/r_4 is equivalent to the set of rational numbers, and that 6 is an instance of a rational number. The inverse operator had inserted an implicit computation; division!

3. Experiment #3

A problem Jx^2dx was given to LEX, and was solved by Op2 as shown in Figure 5. Other instances were also provided that caused the bias adjustment module to be invoked. The program needed to back-propagate the set of real numbers *"r"* through the exponent in Op2 [1]. The code created a description {xKA(x)(match x+1 'r))|, and assimilated it us a specialization of r. This produced a description with infinite recursion of the form "x is a real number if x + 1 is a real number". If the program could have applied knowledge that the real numbers are closed under addition, in this case by 1, then it would have proven that adding 1 to an x will not alter whether that x is a real number.

It is important to consider not only where to assimilate a new description, but whether to assimilate a new description. To circumvent superfluous descriptions, we added an ad hoc procedure for computing closures; if the description $D_{Mt,vv}$ is intended to be a specialization of an existing description $D_{n}i,i$, but $D_{IH\cdot W}$ is more general or equal to D_0m, then $D_{m\,w}$ is superfluous

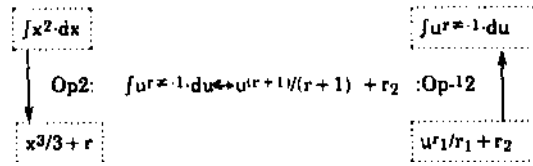

Figure 5 - Solution and propagation for Jx2-dx

IV SUMMARY

We suggested that heuristics used for induction should be found by the learning program itself, not by hand. We presented a model of a concept learning program that adjusts its own bias, thereby altering the inductive generalizations it will make. We implemented an example of the model by adding a bias adjustment module to the existing LEX concept learning program. Experience with the implementation uncovered examples where the formalism was weak, or additional knowledge was needed to determine whether or where to assimilate a newly constructed description. Where these problems were not a factor, the program was able to automatically adjust its own bias.

ACKNOWLEDGEMENTS

I thank Tom Mitchell, Rich Keller, and Donna Nagel for discussions of ideas presented here. I thank Jack Mostow and Bruce Ladendorf for helpful comments on a draft.

REFERENCES

[11 Lenat, D. B., "The Role of Heuristics in Learning by Discovery," *Machine Learning,* Michalski, Carbonell, Mitchell (eds.), Tioga, Palo Alto, 1983, pp. 243-306.

[21 Michalski, R. S., "A Theory and Methodology of Inductive Learning," *Machine Learning,* Michalski, Carbonell, Mitchell (eds.), Tioga, Palo Alto, 1983, pp. 83-134.

[31 Mitchell, T. M., "Version Spaces: A Candidate Elimination Approach to Rule Learning," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* Cambridge, Mass., 1977, pp. 305-310.

[4] Mitchell, T. M., UtgofT, P. E., and Banerji, R. B. "Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics," *Machine Learning,* Michalski, Carbonell, Mitchell (eds), Palo Alto, Tioga, 1983, pp. 163-190.

[51 UtgofT, P. E. and Mitchell, T. M., "Acquisition of Appropriate Bias for Inductive Concept Learning," *Proceedings of the Second National Conference on Artificial Intelligence,* Pittsburgh, August 1982, pp. 414-417.

[61 Vere, S. A. "Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions," *Artificial Intelligence,* 14:2, September 1980, pp. 138-164.