Learning Concept Classification Rules Using Genetic Algorithms

Kenneth A. De Jong
George Mason University
Fairfax, VA 22030 USA
kdejong@aic.gmu.edu

William M. Spears
Naval Research Laboratory
Washington, D.C. 20375 USA
spears@aic.nrl.navy.mil

Abstract

In this paper we explore the use of an adaptive search technique (genetic algorithms) to construct a system GABIL which continually learns and refines concept classification rules from its interaction with the environment. The performance of the system is measured on a set of concept learning problems and compared with the performance of two existing systems: ID5R and C4.5. Preliminary results support that, despite minimal system bias, GABIL is an effective concept learner and is quite competitive with ID5R and C4.5 as the target concept increases in complexity.

1 Introduction

An important requirement for both natural and artificial organisms is the ability to acquire concept classification rules from interactions with their environment. In this paper we explore the use of an adaptive search technique, namely Genetic Algorithms (GAs), as the central mechanism for building a system which continually learns and refines concept classification rules from its interaction with the environment. We show how concept learning tasks can be represented and solved by GAs, and we provide empirical results which illustrate the performance of GAs relative to more traditional methods. Finally, we discuss the advantages and disadvantages of this approach and describe future research activities.

2 Concept Learning Problems

Supervised concept learning involves inducing concept descriptions for the concepts to be learned from a set of positive and negative examples of the target concepts. Examples are represented as points in an n-dimensional feature space which is defined *a priori* and for which all the legal values of the features are known. Concepts are therefore represented as subsets of points in the given n-dimensional space.

A concept learning program is presented with both a description of the feature space and a set of correctly classified examples of the concepts, and is expected to generate a reasonably accurate description of the (unknown) concepts. Since concepts can be arbitrarily complex subsets of a feature space, an important issue is the choice of the concept description language. The language must have sufficient expressive power to describe large subsets succinctly and yet be able to capture irregularities. The two language forms generally used are decision trees [Quinlan, 1986] and rules [Michalski, 1983].

Another important issue arises from the situation that there is a large (possibly infinite) set of concept descriptions which are consistent with any particular finite set of examples. This is generally resolved by introducing either explicitly or implicitly a bias (preference) for certain kinds of descriptions (e.g., shorter or less complex descriptions may be preferred).

Finally, there is the difficult issue of evaluating and comparing the performance of concept learning algorithms. The most widely used approach is a *batch mode* in which the set of examples is divided into a training set and a test set. The concept learner is required to produce a concept description from the training examples. The validity of the description produced is then measured by the percentage of correct classifications made by the system on the second (test) set of examples during which no further learning takes place.

The alternative evaluation approach is an *incremental mode* in which the concept learner is required to produce a concept description from the examples seen so far and to use that description to classify the next incoming example. In this mode learning never stops, and evaluation is in terms of learning curves which measure the predictive performance of the concept learner over time.

This incremental and continuous model of concept learning matches more closely the kind of concept learning that an organism performs as it explores a complex and changing world. Consequently, we use predictive learning curves as our evaluation methodology.

3 Genetic Algorithms and Concept Learning

In order to apply GAs to a particular problem, we need to select an internal representation of the space to be searched and define an external evaluation function which assigns utility to candidate solutions. Both components are critical to the successful application of the GAs to the problem of interest. [1]

3.1 Representing the Search Space

The traditional internal representation used by GAs involves using fixed-length (generally binary) strings to represent points in the space to be searched. However, such representations do not appear well-suited for representing the space of concept descriptions which are generally symbolic in nature, which have both syntactic and semantic constraints, and which can be of widely varying length and complexity.

There are two general approaches one might take to resolve this issue. The first involves changing the fundamental GA operators (crossover and mutation) to work effectively with complex non-string objects [Rendell, 1985]. This must be done carefully in order to preserve the properties which make the GAs effective adaptive search procedures (see [DeJong, 1987] for a more detailed discussion). Alternatively, one can attempt to construct a string representation which minimizes any changes to the GAs.

We are interested in pursuing both approaches. Our ideas on the first approach will be discussed briefly at the end of the paper. In the following sections we will describe our results using the second approach in which we try to apply classical GAs with minimal changes.

3.2 Defining Fixed-length Classifier Rules

Our approach to choosing a representation which results in minimal changes to the standard GA operators involves carefully selecting the concept description language. A natural way to express complex concepts is as a disjunctive set of (possibly overlapping) classification rules. The left-hand side of each rule (disjunct) consists of a conjunction of one or more tests involving feature values. The right-hand side of a rule indicates the concept (classification) to be assigned to the examples which match its left-hand side. Collectively, a set of such rules can be thought of as representing the (unknown) concepts if the rules correctly classify the elements of the feature space.

If we allow arbitrarily complex terms in the conjunctive left-hand side of such rules, we will have a very powerful description language which will be difficult to represent as strings. However, by restricting the complexity of the elements of the conjunctions, we are able to use a string representation and standard GAs, with the only negative side effect that more rules may be required to express the concept. This is achieved by restricting each element of a conjunction to be a test of the form:

> return true if the value of feature i of the example is in the given value set; return false otherwise.

For example, a rule might take the following symbolic form: "if (F2 = large) and (F5 = tall or thin) then it's a widget". Since the left-hand sides are conjunctive forms with internal disjunction, there is no loss of generality by requiring that there be *at most one test for each feature* (on the left hand side of a rule).

With these restrictions we can now construct a fixed-length internal representation for classifier rules. Each fixed-length rule will have N feature tests, one for each feature. Each feature test will be represented by a fixed-length binary string, the length of which will depend of the type of feature (nominal, ordered, etc.). For simplicity, the examples used in this paper will involve features with nominal values. In this case we use k bits for the k values of a nominal feature. So, for example, if the legal values for F1 are the days of the week, then the pattern 0111110 would represent the test for F1 being a weekday.

As an example, the left-hand side of a rule for a 5 feature problem would be represented internally as:

F1	F2	F3	F4	F5
0110010	1111	01	111100	11111

Notice that a feature test involving all 1's matches any value of a feature and is equivalent to "dropping" that conjunctive term (i.e., the feature is irrelevant). So, in the above example only the values of F1, F3, and F4 are relevant. For completeness, we allow patterns of all 0's which match nothing. This means that any rule containing such a pattern will not match (cover) any points in the feature space. While rules of this form are of no use in the final concept description, they are quite useful as storage areas for GAs when evolving and testing sets of rules.

The right-hand side of a rule is simply the class (concept) to which the example belongs. This means that our "classifier system" is a "stimulus-response" system with no message passing.

3.3 Evolving Sets of Classifier Rules

Since a concept description will consist of one or more classifier rules, we still need to specify how GAs will be used to evolve sets of rules. There are currently two basic strategies: the Michigan approach exemplified by Holland's classifier system [Holland, 1986], and the Pittsburgh approach exemplified by Smith's LS-1 system [Smith, 1983]. Systems using the Michigan approach maintain a population of *individual rules* which compete

[1] Excellent introductions to GAs can be found in [Holland, 1975] and [Goldberg, 1989].

with each other for space and priority in the population. In contrast, systems using the Pittsburgh approach maintain a population of *variable-length rule sets* which compete with each other with respect to performance on the domain task.

Very little is currently known concerning the relative merits of the two approaches. In this paper we report on results obtained from using the Pittsburgh approach.[2] That is, each individual in the population is a variable-length string representing an unordered set of fixed-length rules (disjuncts). The number of rules in a particular individual is unrestricted and can range from 1 to a very large number depending on evolutionary pressures.

Our goal was to achieve a representation that required minimal changes to the fundamental genetic operators. We feel we have achieved this with our variable-length string representation involving fixed-length rules. Crossover can occur anywhere (i.e., both on rule boundaries and within rules). The only requirement is that the corresponding crossover points on the two parents "match up semantically". That is, if one parent is being cut on a rule boundary, then the other parent must be also cut on a rule boundary. Similarly, if one parent is being cut at a point 5 bits to the right of a rule boundary, then the other parent must be cut in a similar spot (i.e., 5 bits to the right of some rule boundary).

The mutation operator is unaffected and performs the usual bit-level mutations.

3.4 Choosing a Payoff Function

In addition to selecting a good representation, it is important to define a good payoff function which rewards the right kinds of individuals. One of the nice features of using GAs for concept learning is that the payoff function is the natural place to centralize and make explicit any biases (preferences) for certain kinds of concept descriptions. It also makes it easy to study the effects of different biases by simply making changes to the payoff function.

For the experiments reported in this paper, we wanted to minimize any *a priori* bias we might have. So we selected a payoff function involving only classification performance (ignoring, for example, length and complexity biases). The payoff (fitness) of each individual rule set is computed by testing the rule set on the current set of examples and letting:

$$payoff\,(individual\ i) = (percent\ correct)^2$$

This provides a non-linear bias toward correctly classifying all the examples while providing differential reward for imperfect rule sets.

[2] Previous GA concept learners have used the Michigan approach. See [Wilson, 1987] and [Booker, 1989] for details.

3.5 The GA Concept Learner

Given the representation and payoff function described above, a standard GA can be used to evolve concept descriptions in several ways. The simplest approach involves using a batch mode in which a fixed set of examples is presented, and the GA must search the space of variable-length strings described above for a set of rules which achieves a score of 100%. We will call this approach GABL (GA Batch concept Learner).

The simplest way to produce an incremental GA concept learner is to use GABL incrementally in the following way. The concept learner initially accepts a single example from a pool of examples. GABL is used to create a 100% correct rule set for this example. This rule set is used to predict the classification of the next example. If the prediction is incorrect, GABL is invoked to evolve a new rule set using the two examples. If the prediction is correct, the example is simply stored with the previous example and the rule set remains unchanged. As each new additional instance is accepted, a prediction is made, and the GA is re-run in batch if the prediction is incorrect. We refer to this mode of operation as batch-incremental and we refer to the GA batch-incremental concept learner as GABIL.

4 Evaluating Concept Learning Programs

As suggested in an earlier section, there are many ways to evaluate and compare concept learning programs: in either batch or incremental modes. An incremental concept learner will make a prediction for each new instance seen. Each prediction is either correct or incorrect. We are interested in examining how an incremental system changes its predictive performance over time. Suppose each outcome (correct or incorrect) is stored. We could look at every outcome to compute performance, but this would only indicate the global performance of the learner (a typical batch mode statistic). Instead, we examine a small window of recent outcomes, counting the correct predictions within that window. Performance curves can then be generated which indicate whether a concept learner is getting any better at correctly classifying new (unseen) examples. The graphs used in the experiments in this paper depict this by plotting at each time step (after a new example arrives) the percent correct achieved over the last 10 arrivals (recent behavior).

5 Initial Experiments

The experiments described in this section are designed to demonstrate the predictive performance of GABIL as a function of incremental increases in the size and complexity of the target concept. We invented a 4 feature world in which each feature has 4 possible distinct values (i.e., there are 256 instances in this world). This means that rules map into 16-bit strings and the length of individual

rule sets is a multiple of 16.

In addition to studying the behavior of GABIL as a function of increasing complexity, we were also interested in comparing its performance with an existing algorithm. ID5R [Utgoff, 1989], which is a well-known incremental concept learning algorithm, was chosen for comparison. ID5R uses decision trees as the description language and always produces a decision tree consistent with the instances seen.

We constructed a set of 12 concept learning problems, each consisting of a single target concept of increasing complexity. We varied the complexity by increasing both the number of rules (disjuncts) and the number of relevant features per rule (conjuncts) required to correctly describe the concepts. The number of disjuncts ranged from 1 to 4, while the number of conjuncts ranged from 1 to 3. Each target concept is labeled as nDmC, where n is the number of disjuncts and m is the number of conjuncts.

Each target concept is associated with one experiment. Within an experiment the number of disjuncts and conjuncts for the target concept remains fixed. The variation in target concept occurs between experiments. For each of the concepts, a set of 256 unique, noise free examples was generated *from the feature space* and labeled as positive or negative examples of the target concept. For the more complex concepts, this resulted in learning primarily from negative examples.

For each concept, the 256 examples were randomly shuffled and then presented sequentially as described above. This procedure was repeated 10 times for each concept and for each learning algorithm. The performance curves presented are the average behavior exhibited over 10 runs.[3]

Figures 1 and 2 present the comparative results of applying both GABIL and ID5R to the 1D1C and 4D1C concepts. The remainder of the graphs are not shown due to space limitations. They represent intermediary results between the 1D1C and 4D1C extremes illustrated in Figures 1 and 2. Recall that each point on a curve represents the percentage of correct predictions achieved over the previous 10 instances presented (and averaged over 10 runs). Note that this implies that the learning curves can be and are, in general, non-monotonic. In particular, they will remain at 100% indefinitely only when the algorithms have correctly learned the target concept.

The graphs indicate that, on the simpler concepts, the predictive performance of ID5R improves more rapidly than that of GABIL. However, ID5R degrades in performance as the target concept becomes more complex, with significant deterioration in predictive power

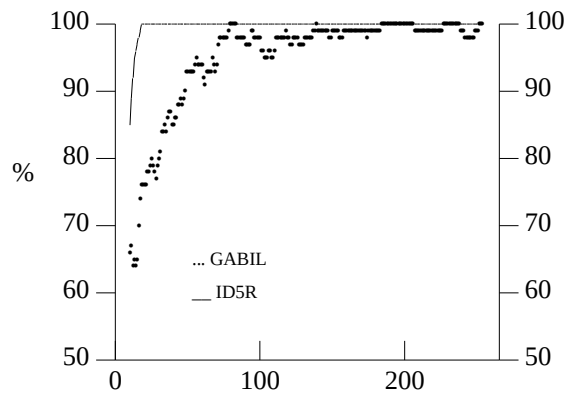

Fig 1. 1D1C

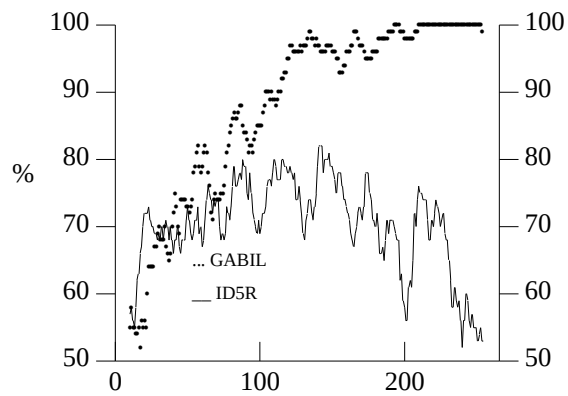

Fig 2. 4D1C

seen as the number of conjuncts and disjuncts are increased. The performance of GABIL, on the other hand, is relatively insensitive to the increase in concept complexity, resulting in significantly better predictive capability than ID5R already on 4 disjunct concepts. The analysis below suggests that this trend will continue with even larger numbers of disjuncts and conjuncts.

We were surprised to see ID5R suffer the most on the 4D1C target concept, since syntactically the concept is only moderately complex. The target concept is of the form:

if (F1 = 0001) or (F2 = 0001) or (F3 = 0001) or
(F4 = 0001) then it's positive

Although it is natural to expect that a simple target concept (from a syntactic viewpoint) would have a small decision tree representation, this is only a rough generalization. This target concept is represented by ID5R as a decision tree of over 150 nodes. In fact, each negative example is represented by a unique leaf node in the

[3] It is not always possible for ID5R to make a prediction based on the decision tree. If it cannot use the tree to predict, we let ID5R make a random prediction.

decision tree. For this reason, ID5R cannot generalize over the negative examples, and has a good chance of predicting any negative example incorrectly. Furthermore, even the positive examples are not generalized well, resulting in prediction errors for positive examples. It is clear that the decision tree representation is poor for representing this particular concept. Target concept 4D1C represents a worst case, which explains why the difference between GABIL and ID5R is greatest for this concept. A similar situation occurs for target concepts 3D1C, 4D2C, and 4D3C, although to a lesser degree.

ID5R relies upon Quinlan's information theoretic entropy measure to build its decision trees. The information theoretic measure favors those concepts in which individual features clearly distinguish target class membership. ID5R's biases also favor concepts that can be represented with small decision trees. The experiments presented above indicate the effect of these built-in biases: the predictive power of ID5R can vary dramatically depending on how well-matched the concept is to these biases.

GABIL, however, performs much more consistently on target concepts of varying complexity. GABIL is not significantly affected by the number of conjuncts, since with our fixed-length rule representation, large conjunctions are no more difficult to find than small ones. There is also no built-in bias towards a small number of disjuncts, although this could be achieved if desired by changing the payoff function, rather than GABIL itself. The overall effect is similar to what has been noted in other GA applications, namely that the overhead of using an adaptive search process is quite evident on simpler problems, but the payoff is clearly seen as the problem complexity increases (see, for example, [Spears and De Jong, 1990]).

6 Further Analysis and Comparisons

Having characterized the behavior of GABIL in this controlled concept world, we have begun to extend the analysis to more complex and challenging problems. One of our first steps was to look at the family of multiplexor problems introduced to the machine learning community by Wilson [Wilson, 1987]. Multiplexor problems fall into the general area of trying to induce a description of a k-input boolean function from input/output examples. Because no single individual input line is useful in distinguishing class membership, information-theoretic approaches like Quinlan's ID3 system have a particularly hard time inducing decision trees for multiplexor problems. Wilson's work indicated that his GA-based classifier system Boole did not have such difficulties. Some of these issues were addressed by Quinlan in the development of his C4 system. Quinlan subsequently reported that C4 outperforms Boole on the multiplexor problems [Quinlan, 1988].[4]

Since we had access to C4.5 (a successor to C4 [Quinlan, 1989]), we felt that a direct comparison of GABIL and C4.5 on multiplexor problems would be enlightening. Since C4.5 is a batch-mode system, we have to run it in a batch-incremental mode in the same manner as GABIL in order to provide meaningful comparisons. This can be achieved by running C4.5 in batch mode for every new instance seen, and using the resulting decision tree to predict the class of the next instance.

The 6-input multiplexor problem has 6 features in which each feature has 2 possible distinct values (i.e., there are 64 instances in this world). This means that rules map into 12-bit strings and the length of individual rule sets is a multiple of 12. For this concept we randomly generated a set of 200 examples from the feature space, each example labeled positive or negative. Since there are only 64 possible unique examples, the set does not contain unique examples, although they are noise free. This methodology allows for direct comparison with Quinlan's reported results [Quinlan, 1988].

The set of 200 examples was randomly shuffled and then presented sequentially. This procedure was repeated 10 times for both learning algorithms. The performance curves presented in Figure 3 are the average behavior exhibited over 10 runs.

GABIL clearly outperforms C4.5 on the 6-input multiplexor problem. As noted above, the weaker performance of C4.5 is not due to the choice of representation (decision tree). In fact, a compact decision tree can be created to describe the concept. The problem lies with the information theoretic bias itself, which makes it hard to find this compact tree. Preliminary results suggest similar performance differentials on larger multiplexor problems.

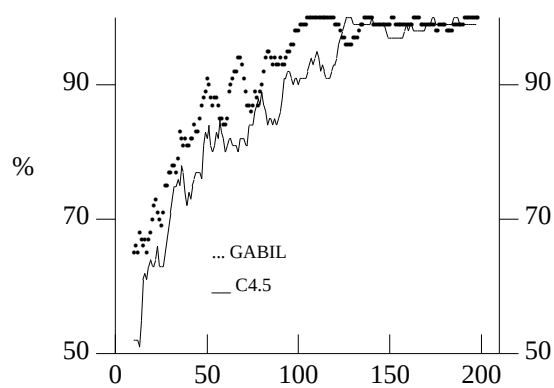

Fig 3. GABIL vs C4.5

[4] However, dramatic improvements to Boole have been reported by [Sen, 1988].

The concept description language and the search algorithm constitute strong biases for any concept learner. The above experiments indicate that ID-like systems can suffer both from their decision tree language bias (see Fig. 2) and from their information theoretic search bias (see Fig. 3). When the biases are appropriate, ID-like systems perform quite well. GABIL, however, due to its minimal system bias, performs uniformly well on target concepts of varying complexity. These initial results support the view that GABIL can be used as an effect concept learner although it may not outperform more strongly biased concept learning algorithms whose bias is appropriate for learning simpler target concepts.

7 Conclusions and Future Research

This paper presents a series of initial results regarding the use of GAs as the key element in the design of a system capable of continuously acquiring and refining concept classification rules from interactions with its environment. It is interesting to note that reasonable performance is achieved with minimal *a priori* bias. The initial results support the view that GAs can be used as an effective concept learner although they may not outperform algorithms specifically designed for concept learning when simple concepts are involved.

This paper also sets the stage for the design of three additional GA-based concept learners. First, we wish to implement a variation of the current system that is truly incremental. Second, we are also very interested in understanding the difference between using the Pittsburgh approach and the Michigan approach in this problem domain. The current fixed-length rule representation can be used directly in Michigan-style classifier systems. Third, we noted early in the paper that there were two basic strategies for selecting a representation for the concept description language. In this paper we developed a representation which minimized the changes to standard GA implementations. We also plan to explore the alternative strategy of modifying the basic GA operators to deal effectively with non-string representations. We feel that the development and analysis of such systems is an important direction the research community should follow in order to develop additional results on these and other problems of interest.

Acknowledgements

We would like to thank Diana Gordon for her support and for many discussions on the biases in supervised concept learning systems. Diana was also instrumental in helping us design our experimental methodology. We would also like to thank John Grefenstette and Alan Schultz for many useful comments about GABIL and crossover, J. R. Quinlan for C4.5, and Paul Utgoff for ID5R.

References

Booker, Lashon B. (1989). Triggered Rule Discovery in Classifier Systems, *Proc. 3rd Int'l Conference on Genetic Algorithms and their Applications*.

De Jong, Kenneth A.(1987). Using Genetic Algorithms to Search Program Spaces, *Proc. 2nd Int'l Conference on Genetic Algorithms and their Applications*.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley Publishing Company, Inc.

Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press.

Holland, John H. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Morgan Kaufmann Publishers, Los Altos, CA.

Michalski, R. (1983). A Theory and Methodology of Inductive Learning. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 1). Tioga Publishing Co., Palo Alto, CA.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, Volume 1, Number 1.

Quinlan, J. R. (1988). *An Empirical Comparison of Genetic and Decision-Tree Classifiers*, Proc. of the 5th Int'l Conference on Machine Learning, June, 1988, Morgan Kaufman Publishers.

Quinlan, J. R. (1989). Documentation and User's Guide for C4.5. (unpublished).

Rendell, L. (1985). Genetic Plans and the Probabilistic Learning System: Synthesis and Results. *Proc. 1st Int'l Conference on Genetic Algorithms and their Applications*.

Sen, S. (1988). Classifier System Learning of Multiplexer Function, Unpublished report available from Sandip Sen, Department of Electrical Engineering, University of Alabama, Tuscaloosa, AL 35486.

Smith, S. F. (1983). Flexible Learning of Problem Solving Heuristics Through Adaptive Search, *Proc. 8th IJCAI*, August 1983.

Spears, W. M. and K. A. De Jong (1990). Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems, *International Joint Conference on Neural Networks*.

Utgoff, Paul E. (1989). Improved Training via Incremental Learning, *Proc. of the 6th Int'l Workshop on Machine Learning*, 62-65.

Wilson, S.W. (1987). Quasi-Darwinian Learning in a Classifier System, *Proc. of the 4th Int'l Workshop on Machine Learning*, June, 1987, Morgan Kaufman Publishing.