Using Genetic Algorithms for Supervised Concept Learning

WILLIAM M. SPEARS
Navy Center for Applied Research in AI
Naval Research Laboratory
Washington, D.C. 20375, USA

and

KENNETH A. DE JONG
Computer Science Department
George Mason University
Fairfax, VA 22030, USA

ABSTRACT

Genetic Algorithms (GAs) have traditionally been used for non-symbolic learning tasks. In this chapter we consider the application of a GA to a symbolic learning task, supervised concept learning from examples. A GA concept learner (GABL) is implemented that learns a concept from a set of positive and negative examples. GABL is run in a batch-incremental mode to facilitate comparison with an incremental concept learner, ID5R. Preliminary results support that, despite minimal system bias, GABL is an effective concept learner and is quite competitive with ID5R as the target concept increases in complexity.

1. Introduction

There is a common misconception in the machine learning community that Genetic Algorithms (GAs) are primarily useful for non-symbolic learning tasks. This perception comes from the historically heavy use of GAs for complex parameter optimization problems. In the machine learning field there are many interesting parameter tuning problems to which GAs have been and can be applied, including threshold adjustment of decision rules and weight adjustment in neural networks. However, the focus of this chapter is to illustrate that GAs are more general than this and can be effectively applied to more traditional symbolic learning tasks as well. [1]

To support this claim we have selected the well-studied task of supervised concept learning. [14,15,17,20] Although there have been a number of applications of genetic algorithms to supervised concept learning, they have largely avoided notice. One of the

[1] For an introduction to Genetic Algorithms, please see Goldberg. [9]

earlier efforts merged genetic algorithms with an existing concept learner.[19] Other efforts have concentrated on the combination of genetic algorithms and classifier systems to concept learning.[3,25] Finally, both Greene[10] and Janikow[13] have studied genetic algorithms (in isolation) as mechanisms for solving concept learning problems. In this chapter, we reemphasize this theme, showing how concept learning tasks can be represented and solved by GAs, and we provide empirical results which illustrate the performance of GAs relative to a more traditional method. Finally, we discuss the advantages and disadvantages of this approach and describe future research activities.

2. Supervised Concept Learning Problems

Supervised concept learning involves inducing concept descriptions from a set of examples of a target concept (i.e., the concept to be learned). Concepts are represented as subsets of points in an n-dimensional feature space which is defined *a priori* and for which all the legal values of the features are known.

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[17] and rules[15].

Another important issue arises from the problem 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 with no further learning.

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.

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.

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. This representation maps well onto parameter optimization problems and there is considerable evidence (both theoretical and empirical) as to the effectiveness of using GAs to search such spaces. [6,9,11,22] 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. [19] This must be done carefully in order to preserve the properties which make the GAs effective adaptive search procedures (see De Jong [7] for a more detailed discussion). Alternatively, one can attempt to construct a string representation which minimizes any changes to the GAs without adopting such a convoluted representation as to render the fundamental GA operators useless.

We are interested in pursuing both approaches. Our ideas on the first approach will be discussed briefly at the end of the chapter. In the following sections we will describe our results using the second approach.

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 (DNF). 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, else return false.

For example, rules might take the following symbolic forms:

> if F1 = blue then it's a block

or

> if (F2 = large) and (F5 = tall or thin) then it's a widget

or

> if (F1 = red or white or blue) and (10 < F4 < 20) then it's a clown

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 nominal features with *k* values we use *k* bits, 1 for each value. 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.

Intervals for features taking on numeric ranges can also be encoded efficiently as fixed-length bit strings, the details of which can be seen in Booker.[2]. For simplicity, the examples used in this chapter will involve features with nominal values.

So, for 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 internal memory.

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, [12] and the Pittsburgh approach exemplified by Smith's LS-1 system. [21] Systems using the Michigan approach maintain a population of *individual rules* which compete 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. As discussed in a later section, one of our goals is to use the domain of concept learning as a testbed for gaining more insight into the two approaches. In this chapter we report on results obtained from using the Pittsburgh approach. 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 chapter, 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:

$$payoff\,(individual\ i) = (percent\ correct)^2 \tag{1}$$

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

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).

Due to the stochastic nature of GAs, a rule set with a perfect score (i.e., 100% correct) may not always be found in a fixed amount of time. So as not to introduce a strong bias, we use the following search termination criterion. The search terminates as soon as a 100% correct rule set is found within a user-specified upper bound on the number of generations. If a correct rule set is not found within the specified bounds or if the population loses diversity (> 90% convergence), [5] the GA simply returns the *best* rule set found. This incorrect (but often quite accurate) rule set is used to predict (classify) future examples.

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 rerun 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. Empirical Studies

The experiments described in this section are designed to compare the predictive performance of GABIL with ID5R as a function of incremental increases in the size and complexity of the target concept.

4.1. *Evaluating Concept Learning Programs*

As suggested in the introduction, there are many ways to evaluate and compare concept learning programs: in either batch or incremental modes. We tend to favor incremental learning systems since the world in which most learning systems must perform is generally dynamic and changing. In this context we prefer the use of learning curves which measure the change in a system's performance over time in a (possibly) changing environment.

In the domain of supervised concept learning, this means that we are interested in situations in which examples are accepted one at a time. In this mode, a concept learner must use its current concept descriptions to classify the next example. The concept learner then compares its classification with the actual class of the example. Based on this comparison the concept learner may add that example to the existing set and attempt to reformulate new concept descriptions, or it may leave the current descriptions unchanged.

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 chapter depict this by plotting at each time step (after a new example arrives) the percent correct achieved over the last 10 arrivals (recent behavior).

4.2. *Implementation Details*

All of our experiments have been performed using a C implementation of the GAs. In all cases the population size has been held fixed at 100, the variable-length 2-point crossover operator has been applied at a 60% rate, the mutation rate is 0.1%, and selection is performed via Baker's SUS algorithm. [1]

4.3. *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 our GA-based concept learner (GABIL) as a function of increasing complexity, we were also interested in comparing its performance with an existing algorithm. Utgoff's ID5R, [24] 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 labelled as *n*D*m*C, 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. [2]

ID5R and GABIL use significantly different approaches to concept learning. Therefore, we expect their performance behaviors to differ. As the number of disjuncts and conjuncts increases, the target concept (viewed syntactically as a logical DNF expression) becomes more difficult. In general, a more complex target concept requires a larger decision tree (although this is not always true). ID5R relies upon Quinlan's information theoretic entropy measure to build its decision trees. This measure works well when individual features are meaningful in distinguishing an example as positive or negative. As the number of disjuncts and/or conjuncts increases, individual features become less informative, resulting in larger decision trees and poorer predictive performance. ID5R's information theoretic biases will therefore perform better on simpler target concepts.

GABIL, however, should perform uniformly well on target concepts of varying complexity. GABIL should not be affected by the number of conjuncts, since with our fixed-length rule representation, large conjunctions are no more difficult to find than

---

[2] 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.

small ones. There is also no bias towards a small number of disjuncts. Given these biases (and lack of biases), then, it is natural to expect that while ID5R will outperform GABIL on the simpler concepts, there will exist a frontier at which the situation will reverse.

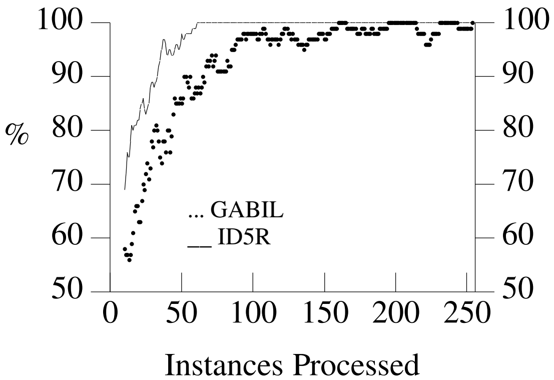
For the sake of brevity we present graphs of 7 of the 12 experiments. Figure 1 depicts the comparative results on target concept 2D1C. It is representative of the results on all the 1 and 2 disjunct concepts. Figures 2 - 7 present the comparative results of applying both GABIL and ID5R to the more difficult concepts (3 and 4 disjuncts). Recall that each point on a curve represents the percent correct achieved over the previous 10 instances (and averaged over 10 runs).

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, and GABIL starts to win on the 4 disjunct concepts. We expect this trend to continue with even larger numbers of disjuncts and conjuncts.

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. 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

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 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 (which is also a bias) is poor for representing this



Instances Processed
Fig 1. 2D1C

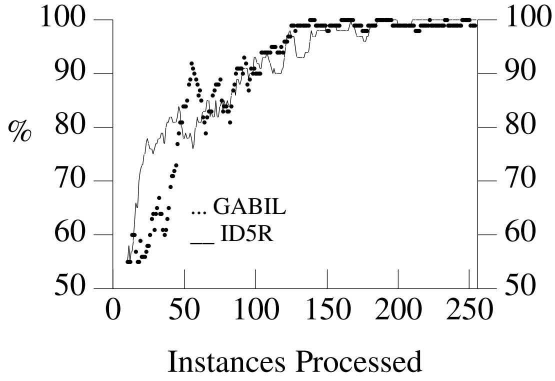


Fig 2. 3D1C

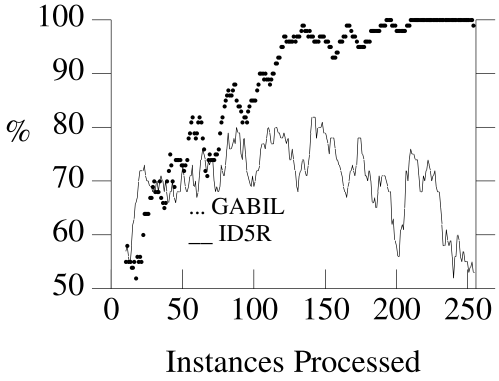


Fig 3. 4D1C

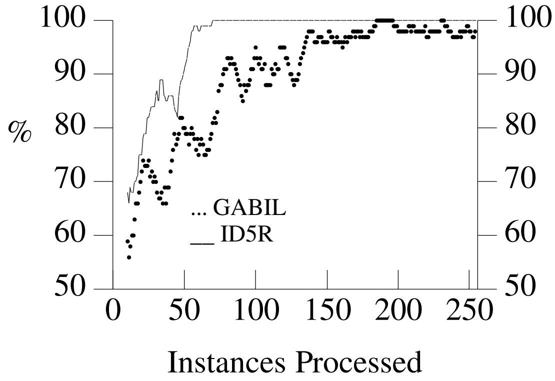


Fig 4. 3D2C

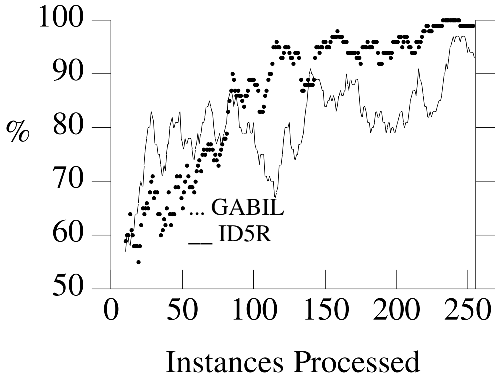


Fig 5. 4D2C

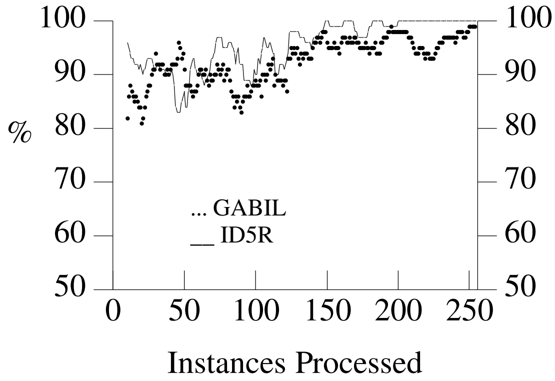


Fig 6. 3D3C

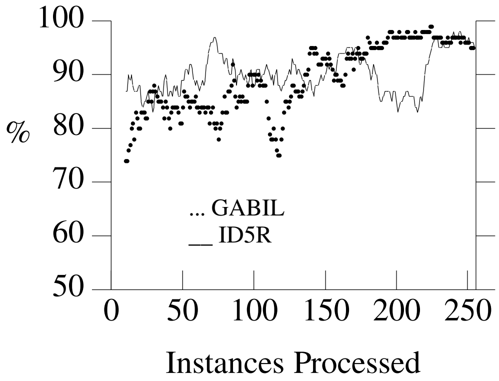


Fig 7. 4D3C

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.

The experiments indicate that ID5R often degrades in performance as the number of disjuncts and conjuncts increases. ID5R's biases favor concepts that can be represented with small decision trees. The information theoretic measure favors those concepts in which individual features clearly distinguish target class membership. GABIL does not have these biases, and appears to be less sensitive to increasing numbers of disjuncts and conjuncts. GABIL does not degrade significantly with increasing target concept complexity and outperforms ID5R on 4 disjunct concepts. Since the syntactic complexity of a target concept corresponds roughly with the size of its decision tree representation, we expect this trend to continue with more difficult target concepts.

5. Further Analysis and Comparisons

In other work we have compared GABIL with AQ15 [16] and Quinlan' C4.5 [18] on the *n*D*m*C target concepts and the breast cancer domain. [8,23] Most recently, GABIL has been augmented with concept learning specific operators, which are selected dynamically by the genetic algorithm while concept learning occurs. [23]. The result is a robust concept learner that is competitive with the more traditional systems.

In the future we plan to perform additional experiments involving the comparison of GABIL with other concept learning programs such as Clark's CN2, [4] Janikow's GIL, [13] and the concept learners based on neural network techniques. We also plan to implement and analyze other GA-based concept learners. The first is a variation of the current one which is truly incremental rather than batch-incremental. We feel that this change will smooth out many of the bumps in the learning curves currently due to completely reinitializing the population when an incorrect classification is made on a new example.

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. We plan to implement such a system and compare the two approaches.

Finally, we noted early in the chapter that there were two basic strategies for selecting a representation for the concept description language. In this chapter 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. In particular, we plan to use Michalski's VL1 language and compare this approach to using GAs with the current work.

6. Conclusions

This chapter presents a series of initial results regarding the use of GAs for symbolic learning tasks. In particular, a GA-based concept learner is developed and analyzed. It is interesting to note that reasonable performance is achieved with minimal bias. There is no preference for shorter rule sets, unlike most other concept learning systems. 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 chapter also sets the stage for additional comparisons between GAs and other concept learning algorithms. We feel that such comparisons are important and encourage the research community 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.

References

1. J. E. Baker, Reducing Bias and Inefficiency in the Selection Algorithm, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, (Lawrence Erlbaum, 1987), pgs. 14 - 21.
2. L. B. Booker, *Intelligent Behavior as an Adaptation to the Task Environment*, Doctoral Thesis, CCS Department, University of Michigan, (1982).
3. L. B. Booker, Triggered Rule Discovery in Classifier Systems, *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, (Morgan Kaufmann, 1989), pgs. 265 - 274.
4. P. Clark and T. Niblett, The CN2 Induction Algorithm, *Machine Learning,* **3**, 4, (1989).
5. K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems,* Doctoral Thesis, Dept. Computer and Communication Sciences, University of Michigan, Ann Arbor, (1975).
6. K. A. De Jong, Genetic Algorithms: a 10 Year Perspective, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, (Lawrence Erlbaum, 1985), pgs. 169 - 177.
7. K. A. De Jong, Using Genetic Algorithms to Search Program Spaces, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, (Lawrence Erlbaum, 1987), pgs. 210 - 216.
8. K. A. De Jong and W. M. Spears, Learning Concept Classification Rules Using Genetic Algorithms, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, (Morgan Kaufmann, 1991), pgs. 651 - 656.
9. D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning,* (Addison-Wesley, 1989).
10. D. Greene and S. Smith, A Genetic System for Learning Models of Consumer

Choice, *Proceedings of the Second International Conference on Genetic Algorithms*, (Lawrence Erlbaum, 1987), pgs. 217 - 223.
11. J. H. Holland, *Adaptation in Natural and Artificial Systems,* (Michigan Press, 1975).
12. J. H. Holland, 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*, **2**, (Morgan Kaufmann, 1986).
13. C. Janikow, *Inductive Learning of Decision Rules from Attribute-Based Examples: A Knowledge Intensive Genetic Algorithm Approach*, TR91-030, Doctoral Thesis, Department of Computer Science, University of Chapel Hill, North Carolina, (1991).
14. T. Mitchell, *Version Spaces : An Approach to Concept Learning*. Doctoral Thesis, Stanford University, Stanford, CA, (1978).
15. R. Michalski, A Theory and Methodology of Inductive Learning. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, **1**, (Tioga Publishing, 1983).
16. R. Michalski, I. Mozetic, J. Hong, and N. Lavrac, The AQ15 Inductive Learning System: An Overview and Experiments. University of Illinois Report Number UIUCDCS-R-86-1260, (1986).
17. J. R. Quinlan, Induction of Decision Trees. *Machine Learning,*, **1**, 1, (1986).
18. J. R. Quinlan, Documentation and User's Guide for C4.5, (unpublished, 1989).
19. L. Rendell, Genetic Plans and the Probabilistic Learning System: Synthesis and Results. *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, (Lawrence Erlbaum, 1985), pgs. 60 - 73.
20. L. Rendell, H. Cho, and R. Seshu, Improving the Design of Similarity-Based Rule-Learning Systems. *International Journal of Expert Systems*, **2**, 1, (1989).
21. S. F. Smith, Flexible Learning of Problem Solving Heuristics Through Adaptive Search, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, (William Kaufmann, 1983), pgs. 422 - 425.
22. W. M. Spears, *Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems*, Masters Thesis, CS Department, George Mason University, (1990).
23. W. M. Spears and D. F. Gordon, Adaptive Strategy Selection for Concept Learning, *Multistrategy Learning (MSL-91) Workshop*, (1991), pgs. 231 - 246.
24. P. E. Utgoff, Improved Training via Incremental Learning, *Proceedings of the Sixth International Workshop on Machine Learning*, (Morgan Kaufmann, 1989), pgs. 62 - 65.
25. S. W. Wilson, Classifier Systems and the Animat Problem, *Machine Learning*, **2**, 4, (1987).