Quasi-Darwinian Learning in a Classifier System

STEWART W. WILSON (WILSON@THINK.COM)

The Rowland Institute for Science,
100 Cambridge Parkway, Cambridge, MA 01742 U.S.A.

Abstract

Classifier systems (Holland, 1986) have a distinct Darwinian flavor, and in this respect contrast sharply with most other learning systems. In this paper we bring out various aspects of the contrast, and provide an example of classifier system learning which illustrates its quasi-Darwinian operation.

1. Introduction

In simplest terms, Darwin's concept of the evolution of organisms (Curtis, 1983) involved: (1) a *population* of individuals similar in their features but with some heritable chance *variations*; and (2) an environment in which certain of the variations gave their possessors a greater likelihood of surviving and reproducing, a process he termed *natural selection*. Over time, the population would slowly change to contain individuals having heredities that made them better equipped to survive in the given environment.

Few machine learning systems exhibit this pattern. In the first place, the notion of population is often absent. The neural-network-like systems (Rumelhart, McClelland, and the PDP Group, 1986) contain numerous processing units, but interconnected so as to form a single individual. Decision tree systems (Quinlan, 1983) also consist of a single structure. However, systems maintaining a set of concepts, as in Mitchell (1982) and Anderson (1983), are population-like in the sense that the concepts may overlap and can be viewed as being in competition with each other.

With respect to variation, machine learning systems tend to use *purposeful* mechanisms instead of Darwin's chance. That is, variation occurs either as a specific response to the system's adequacy in dealing with environmental events or to test some specific implication of existing internal structure. The neural networks are adjusted in a manner directly reflecting experience; those with a stochastic element still only produce a single variant at a time, since there is no population. Variation in systems like Mitchell's depends directly on experience ("data-driven"), while systems like Anderson's use operators (composition, discrimination) which may be driven by experience, or are triggered by relationships between rules (generalization).

In comparison with the above, *classifier systems* relate quite well to the Darwinian pattern. There is a population of condition-action rules called classifiers, each of which may be regarded as a tentative hypothesis about the best way to respond to some aspect of the environment. The classifiers acquire *strength*, a kind of fitness which results from their relative success in dealing with the environment. The strength influences the number of a classifier's offspring (in other systems, there appears to be no analog of reproduction). And variation—using operators such as *crossover* and, less importantly, *mutation*--occurs at random among the offspring.

We shall illustrate this process by showing a simple classifier system incrementally learning a difficult boolean function. The example will give "snapshots" of the population at different epochs, so that the population's changing composition can be observed. Before proceeding to the example, we describe from a Darwinian perspective the system's principles of operation.

2. A Simple Classifier System

Our classifier system for boolean problems—called ***Boole***—is a specialization of Holland's (1986) general model. It differs principally in leaving out the ability to deal with sequential tasks. Learning a boolean function means learning to give the correct value for the function (0 or 1) for any combination of values of the input variables. If the inputs occur simultaneously as a string, the system's decision can be made in one time-step since all needed information is present. ***Boole*** thus omits the parts of the general model that support sequential operation (the message list and bucket-brigade). In return, ***Boole*** gains a certain transparency which has aided in investigating classifier system principles (Wilson, in press); in addition, ***Boole*** or a successor may have practical application in the area of learning from examples.

Boole's classifiers each consist of a single condition, or *taxon*, and an action. For boolean problems with L input variables, the taxon is a string of length L from $\{1, 0, \#\}$. A classifier's condition is satisfied if its taxon *matches* the current input string, which means that for each 1 or 0 in the taxon, the same value occurs at the corresponding position in the input string; "#" functions as a "don't care" symbol and matches unconditionally. The #'s confer generality on the taxon (and thus on the classifier) in the sense that 2^n distinct input strings can be matched by a taxon containing n #'s. The classifier's action is simply a single bit, 0 or 1, representing one of the two possible values for the function. An example of a classifier for $L = 6$ would be:

1 0 # 0 1 0 / 1 ,

where the taxon and action are separated by "/". The classifier population [P] is usually initialized by filling all taxa with 1, 0, and # according to some random rule; actions are similarly filled in. In the experiment illustrated in Section 3, [P] contained 400 classifiers, initialized with a uniform random distribution. Classifiers' strengths were initialized at a common value (100).

Like the general model, ***Boole*** has *performance*, *reinforcement*, and *discovery* components.

2.1 Performance Component

In the performance cycle an input string is presented and the system decides on its "answer": 0 or 1. The cycle has just two steps:

1) Form the *match set* [M] of all classifiers whose taxa match the input string.
2) Select a single classifier from [M] using a probability distribution over the strengths of [M]'s classifiers; that is, the probability of selection of a particular classifier is equal to its strength divided by the sum of the strengths of classifiers in [M]. Output the selected classifier's action as the system's decision.

In effect, the system asks which classifiers in [P] "recognize" (match) the current input, then from these tends to choose the action having the greatest total strength among the classifiers advocating that action. From a Darwinian standpoint, each input defines a kind of "niche" populated by the matching classifiers. The matchers "compete", based on their strengths, for control of the system's decision, which control, as will be seen next, entails the right to benefit from whatever payoff the environment accords the decision. Note that the "niches" defined by input strings tend to overlap,

since classifiers with at least one # compete in more than one niche; such classifiers potentially receive payoff more frequently than other, more specialized classifiers.

2.2 Reinforcement Component

The reinforcement component adjusts classifier strengths in accordance with payoff received from the environment following each performance cycle. Reinforcement improves overall performance by strengthening payoff-achieving classifiers and (possibly) weakening less remunerative ones. Over time, reinforcements give a classifier an average strength which becomes its "fitness" in the discovery component's reproductive competition. We have experimented with three reinforcement schemes, all encompassed by the following algorithm.

1) Form the subset of [M] consisting of classifiers whose action is the same as the action chosen in the performance cycle. This called the *action set* [A]. The remaining members of [M] are the set not[A].
2) Deduct a fraction e from the strengths of all classifiers in [A].
3) If the system's decision was correct, distribute a payoff quantity R to the strengths of [A]; but
4) If the decision was wrong, distribute a payoff quantity R' (where $0 \leq R' < R$) to the strengths of [A] and deduct a fraction p from the strengths of [A] (at least one of R' and p is equal to 0).
5) Deduct a fraction t from the strengths of not[A].

Step 4 defines three different payoff regimes: *payoff-penalty*, in which $p \neq 0$; *payoff-only*, in which both p and R' equal 0; and *payoff-payoff*, in which $R' \neq 0$. The first is designed to reflect environments in which the system can definitely determine that a certain answer was incorrect (*e.g.*, environmental response was painful). The second regime corresponds to environments that are generally indifferent to actions until the "right" action is produced. The third regime, payoff-payoff, reflects the common case where any of several possible actions will receive some payoff, but one of them will get the most (consider, *e.g.*, another's degree of cooperation as a function of one's own actions). The three regimes are in fact characteristic of natural environments in which actions can be rated—as to their degree of need satisfaction, say—but teacher-like feedback disclosing the *identity* of the best action is rare.

The strength fraction deducted in step 2 can be thought of as a "cost" paid by each classifier in [A] for the chance to receive payoff. Technically, it causes the strength of a classifier to tend toward a recency-weighted average of previous payoffs (divided by e).

In the simplest case, the total payoff to [A] is divided equally among [A]'s members, so that the amount each classifier gets is inversely proportional to the number of recipients. From the Darwinian perspective, the fitness of a classifier is thus not only a function of its merit in recommending the best action, but also of its *redundancy*, in the niches it occupies, with respect to other, similar classifiers. Since R is finite, splitting payoff means that if too many classifiers occupy a niche, they will be weak; the action of the discovery component will then tend to reduce their numbers. If very few classifiers occupy a niche, they will be strong and the discovery component will preferentially multiply them, as will be seen. Thus the splitting of payoff leads naturally to a balanced allocation of system resources (classifiers) among different niches.

2.3 Discovery Component

The discovery component is based on Holland's (1975) genetic algorithm and employs *reproduction*, *genetic operations*, and *deletion*. At regular intervals of trials, clas-

sifiers ("parents") are selected probabilistically from [P] according to strength and copied. Genetic operators are applied to at least some of the copies. Then all the copies ("offspring") are added to [P] and a like number of weak classifiers are deleted. The principal genetic operators are crossover and mutation. In crossover, a randomly selected segment is exchanged between a pair of classifiers. In mutation, the values at one or more positions in a classifier are changed to one of the other possible values; *e.g.*, 1 would be changed to either 0 or # in the case of a taxon position.

The Darwinian content of this process resides in the preferential reproduction of more fit (stronger) classifiers, followed by the random application to some of the offspring of variation-inducing operators. Deletion may be regarded as a compensating death process which keeps the population size constant. Crossover is an interesting variation operator for classifiers because it tends to produce offspring having different degrees of generality (numbers of #'s) from either parent. Inserted in the population to compete with the parents and others, a slightly more general or more specific classifier may be more effective in gaining payoff than they are, so it and its offspring will survive and the competitors may eventually be replaced.

Since stronger-than-average classifiers will tend to be preferentially reproduced, under-populated niches containing "well-paid" classifiers will tend to gain members; the opposite will occur for over-populated niches. The variation operators bring into existence new classifiers which better cover the niches in the sense of matching in as many niches as possible while minimizing error. It is important to observe, however, that variation applies *randomly* to offspring of high-performance classifiers. The incidence of variation is prompted not by the form, content, or meaning of existing classifiers, but only by their success. This would appear to be the essence of the Darwinian mechanism, and the principal contrast with the more purposeful mechanisms of other learning systems.

The foregoing discussions have focussed on ***Boole***'s Darwinian aspects. Additional information about the system, and experimental results, will be found in Wilson (in press).

3. A Learning Example

This section illustrates the evolution of a population of classifiers as ***Boole*** learns a disjunctive boolean function.

3.1 The Function

We have experimented extensively with the so-called "multiplexer" function also used by Barto (1985). For a six-bit input string, the value of the function is given by the following boolean expression, where the x_i are the input variables:

$$F_6 = x'_0 x'_1 x_2 + x'_0 x_1 x_3 + x_0 x'_1 x_4 + x_0 x_1 x_5 \quad .$$

Examination shows that F_6 is disjunctive: there is more than one term and no more than one term can have value 1 for any input string. Note that the values of x_0 and x_1 "select" one of the four terms and "read out" the value of the remaining variable in that term as the function's value—suggesting the "multiplexer" interpretation. In general, for each integer $k > 0$ there is a multiplexer function of strings of length $L = k + 2^k$, containing 2^k terms. Besides the 6-multiplexer, we have done experiments in which ***Boole*** learned the 11-multiplexer ($k = 3$, $L = 11$) and the 20-multiplexer ($k = 4$). Because the multiplexers are rather intricate and form a family of increasing difficulty they would appear well suited to learning system experiments.

3.2 Population Evolution

Data from a 6-multiplexer experiment are shown in Table 1. In the experiment, strings were generated by a uniform random process and presented one at a time (*i.e.*, incrementally), with each presentation constituting a *trial*. After each trial ***Boole*** received payoff according its decision and the payoff regime in effect, which in this experiment was payoff-penalty.[1]

The six panels of Table 1 show the 16 most numerous classifier types of [P] at 0, 500, 1000, 3200, 5200, and 12000 trials. The taxon and action of a classifier type are listed, followed by its number of instances (exact copies). The 0 trials panel contains only a fraction of the randomly generated population (53 of 400 instances). Careful inspection will show that four of the classifier types in the 0 trials panel are correct every time they match, three are wrong every time, and the rest are sometimes right and sometimes wrong. The score for the 0 trials population is the average for 50 trials with learning turned off. Other scores average over the previous 50 trials (with learning of course turned on).

As the number of trials increases, scores increase to a high level. Perhaps more significantly from the Darwinian point of view, the total number of instances in the top 16 types rises until by 12000 trials all but 11 population members are included. What has happened is that certain classifier types have turned out to be so superior that they have multiplied and nearly "taken over" the population. In particular, the first eight in the 12000 trials panel account for most of the population (and most of the strength, as well). They are especially interesting because examination shows that they correspond to the terms of F_6. For this reason we called these eight classifier types the "solution set" [S6]. No member of [S6] existed in the original population. The first instance occurred somewhere between 200 and 300 trials. Their gradual appearance and proliferation may be traced in Table 1.

Each member of [S6] matches exactly eight of the possible input strings and the eight matched sets are disjoint. Thus the population might be said to have discovered and adapted itself to eight "niches" which optimally cover the problem environment. It is also noteworthy that once all members of [S6] are found, the allocation of instances to niches becomes approximately even.

4. Conclusion

The point of this paper has been to show how a rather difficult disjunctive incremental learning task can be solved by a method which adheres closely to the Darwinian principles of random variation and selective reproduction based on success. Given its reputation in natural evolution, it is intriguing that the Darwinian process can apparently also be applied quite effectively to the evolution of rules. This suggests it would be fruitful to investigate rule-based inductive systems which use both Darwinian and purposeful mechanisms, as well as to see how far the purely Darwinian approach can be taken.

---

[1] System parameters for the experiment were: $e = 0.1$, $R = 1000$, $R' = 0$, $p = 0.8$, $t = 0.1$, $G = 4.0$ (see Wilson, in press), and the discovery component generated offspring at an average rate of one per trial. Crossover applied, on average, to one-eighth of the offspring. Mutation occurred with probability 0.001 per classifier position.

Table 1. The 16 most numerous classifier types in [P] at six trial epochs.

0 trials, score 48.0% (Taxon/Action	Number)		500 trials, 75.9%			1000 trials, 84.6%		
0 1 1 # 1 0	1	5	1 # 0 # 0 0	0	20	1 1 0 # # 1	1	30
1 # # 1 0 1	0	5	0 # # 1 # 1	1	18	0 # 1 1 # 0	1	24
# 1 0 1 1 #	1	4	0 # 0 # # 1	0	16	1 1 1 # 0 1	1	21
0 1 0 0 1 #	1	3	0 1 # 1 0 0	1	15	1 0 # # 1 #	1	18
1 1 1 1 1 #	0	3	1 1 0 # # 0	0	12	1 # 0 # 0 0	0	18
0 1 # # # 1	1	3	0 # 1 1 # 0	1	11	0 # 1 1 0 1	1	17
1 # 1 # # 1	0	3	# 0 1 # # 0	1	10	1 # # # 1 1	1	16
0 0 # 1 # 1	1	3	0 0 1 0 # 1	1	10	# 1 1 # # 0	0	15
1 1 # 1 # 1	0	3	1 # # # 1 1	1	10	# 0 0 # # 1	0	14
1 # # # 0 1	0	3	0 1 # 0 1 1	0	10	1 0 # # 0 0	0	14
# # 1 # 1 1	1	3	1 0 # # 0 0	0	9	# 0 1 1 0 #	0	13
1 1 1 # 0 1	1	3	# 1 0 1 1 #	1	8	1 1 0 # # 0	0	13
1 1 1 # 1 1	1	3	0 0 1 0 0 #	1	8	# # 1 # 1 1	1	12
1 # # 0 1 1	0	3	1 1 1 # 0 1	1	8	0 1 # 1 0 0	1	12
# 1 0 1 1 1	1	3	1 1 1 # # 0	0	7	0 0 1 0 0 #	1	11
1 1 0 1 1 0	0	3	# 1 # 1 1 1	1	7	1 1 1 # # 0	0	10
	total	53			179			258

3200 trials, score 91.2%			5200 trials, 93.9%			1200 trials, 97.3%		
1 0 # # 1 #	1	53	1 0 # # 1 #	1	60	1 1 # # # 1	1	55
1 0 # # 0 #	0	43	0 0 1 # # #	1	55	1 1 # # # 0	0	52
0 # 1 1 # #	1	38	1 0 # # 0 #	0	50	1 0 # # 0 #	0	49
# 1 # 0 # 0	0	37	1 1 # # # 1	1	47	0 0 1 # # #	1	45
1 1 0 # # 0	0	29	0 1 # 1 # #	1	46	1 0 # # 1 #	1	44
# 1 # 1 # 1	1	28	1 1 # # # 0	0	41	0 0 0 # # #	0	38
1 1 # # # 1	1	25	0 # 0 0 # #	0	22	0 1 # 1 # #	1	35
0 1 # 1 # 0	1	17	# 1 # 0 # 0	0	21	0 1 # 0 # #	0	31
1 # # # 1 1	1	15	0 1 # 0 # #	0	8	0 # 0 0 # #	0	19
1 1 1 # # 0	0	14	# 1 # 0 # #	0	4	0 0 0 1 # #	0	8
0 # 0 0 # 1	0	14	0 # 1 1 # #	1	4	0 # # 0 # #	0	4
# 0 1 # 1 1	1	9	0 # 0 1 1 #	0	4	1 0 1 # # #	1	2
# 0 # 1 # #	0	8	0 1 # # # #	1	3	0 0 0 # 0 #	0	2
0 1 # 1 # #	1	8	0 # # 1 # #	1	3	1 1 0 # # 0	0	2
# 0 # # 0 #	0	8	# 1 # # # 0	0	3	0 0 1 # # 1	1	2
# 0 1 # # 0	1	5	0 # 0 0 # 1	0	3	0 0 # # # #	0	1
	total	351			374			389

References

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge: Harvard University Press.

Barto, A. G. (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, **4**, 229-256.

Curtis, H. (1983). *Biology.* New York: Worth.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Holland, J. H. (1986). Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine Learning, an Artificial Intelligence Approach. Volume II.* Los Altos, California: Morgan Kaufmann.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence* **18**, 203-226.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end-games. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (Eds.), *Machine Learning, an Artificial Intelligence Approach.* Palo Alto, California: Tioga.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986). *Parallel Distributed Processing. Explorations in the Microstructure of Cognition.* Cambridge, Massachusetts: MIT Press (Bradford).

Wilson, S. W. (in press). Classifier systems and the animat problem. *Machine Learning.*