<head>
    BEAGLE -- A Darwinian Approach to Pattern Recognition:

        Polytechnic of North London, 1980.
        Published in Kybernetes, 1981.
</head>

<body>
        BEAGLE -- A Darwinian Approach to Pattern Recognition

        "There is grandeur in this view of life, with its several powers, having been originally breathed by the Creator into a few forms or into one; and that whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being evolved."

                Charles Darwin -- The Origin of Species.

ABSTRACT

BEAGLE (Biological Evolutionary Algorithm Generating Logical Expressions) is a computer package for producing decision-rules by induction from a database. It works on the principle of 'Naturalistic Selection' whereby rules that fit the data badly are 'killed off' and replaced by 'mutations' of better rules or by new rules created by 'mating' two better adapted rules.
The rules are Boolean expressions represented by tree structures.

The software consists of two Pascal programs, HERB (Heuristic Evolutionary Rule Breeder) and LEAF (Logical Evaluator And Forecaster). HERB improves a given starting set of rules by running over several simulated generations. LEAF uses the rules to classify samples from a database where the correct membership may not be known. Preliminary tests on three different databases have been carried out -- on hospital admissions (classing heart patients as deaths or survivors), on athletic physique (classing Olympic finallists as long-distance runners or sprinters) and on football results (categorizing games into draws and non-draws).

It appears from the tests that the method works better than the standard discriminant analysis technique based on a linear discriminant function, and hence that this long-neglected approach warrants further investigation.

1. INTRODUCTION

This report describes BEAGLE (Biological Evolutionary Algorithm Generating Logical Expressions) which is a computer system for producing decision-rules by induction from a database. It works on the principle of natural -- or at least naturalistic -- selection. Thus it represents a weaving-together of strands in the thought of three great 19th-century Englishmen, Boole, Babbage and Darwin.

While 'knowledge engineering' or 'knowledge refining' is currently enjoying something of a vogue and has already begun to

produce impressive results [Buchanan & Mitchell, 1978; Quinlan, 1979], this report contains a plea not to neglect a parallel endeavour with a less mechanistic flavour that we might call 'knowledge farming' or perhaps 'sophiculture'. In particular, it is the author's contention that the great principle of natural selection is a valuable tool in the stock-in-trade of the conscientious knowledge engineer (or farmer).

The idea of systems that improve by a computational analogy with survival of the fittest has been pursued before [Pask, 1961; Bernstein & Rubin, 1965; Fogel et al., 1966] but has lapsed from favour somewhat since the pioneering spirit of Cybernetics was consolidated into the mature (?) discipline of Artificial Intelligence. Selfridge's 'Pandemonium' [Selfridge, 1959] was an early example of a system designed to contain "the seeds of self-improvement" which involved, among other things, replacing 'demons' which discriminated poorly among the input patterns they were supposed to distinguish with new 'demons' formed by randomly altering the parameters of surviving ones. But probably the only really thorough-going attempt to 'breed' intelligence in the abstract was by Barricelli and Bell [Bell, 1972].

Barricelli's 'symbioorganisms' were sequences of integers that existed in a universe consisting of an array of cells. Whenever two organisms both attempted to expand into the same space a game of Tac-Tix was played between them, to the death. The number patterns of the organisms were interpreted as moves in the game. The surviving organisms were allowed to reproduce (asexually, it appears) and some random mutations introduced, after which the process was repeated. After some thousands of generations he had a collection of organisms that were expert at Tac-Tix.

Barricelli found it quite an effective technique,
and it is my view that it is due for a revival.

2. BEAGLE -- THE USER'S VIEW

The system as presently implemented consists of two Pascal programs running on the DEC System-10 at Polytechnic of North London, namely HERB (Heuristic Evolutionary Rule Breeder) and LEAF (Logical Evaluator And Forecaster). They can be accessed like any other statistical package and in funcion correspond most closely to discriminant analysis. Together they perform the task of classifying samples into one of two or more categories on the basis of the values of a number of measures or parameters describing each sample. HERB creates and/or modifies the classification rules which LEAF then uses, typically to forecast group membership for samples whose class is not known.

2.1 The HERB Program

HERB requires three input files -- a datafile, a payoff file and an old rule file (possibly empty). It produces as output a new rule file which is as good as or better than the old one.

The datafile contains a 'training set' of samples for which the categories are known. It should begin with two integers, W and F. W is the width in characters of the description field for each sample (0 if absent). F is the number of features. Then follows the data -- for each case the description field of W

characters, F numbers which are measures for the case on each feature or variable (integers only at present with at least one space or new line to separate them), and lastly a number indicating the actual category to which that case belongs. (The category number must end a line.)

There follows the first three lines of a typical datafile.

```
4 18
517 68 165 1 2 114 88 95 73 17 141 66 115 225 110 562 206 113 340
1
```

This is the beginning of a file of data from 113 patients admitted to hospital with heart complaints [Afifi & Azen, 1972]. Each patient was measured on 18 variables on admission. Preceding the 18 scores is an identification number (4 characters) which is 517 for this patient. Following the scores is the category number (1=lived, 2=died). These cases were used for testing: see section 4. (The first 5 variables are age, height and sex; so this patient was 68 years old, 165 cm tall, and male .... he survived.)

To enable the program to assess each rule's performance the user must also furnish a payoff matrix in a separate file which effectively states the value or cost of each classification or misclassification. The payoff file also indicates how many categories are in use. Since the program works on tri-state logic where 1=yes, 0=don't-know and -1=no this means a 3 by NC table where NC is the number of classes. (Later releases will allow the user to specify one of several multi-state logics of which 0..1, Boolean, will be a special case.)

For the tests on the hospital admission data the payoff matrix was as follows.

	Actual Class	
Computer Decision	1 (lived)	2 (died)
-1 (no)	-1	+1
0 (maybe)	0	0
1 (yes)	+1	-1

Thus a rule gained a point for a correct classification and lost one for an incorrect one. More complex reward/punishment schedules with more classes are of course possible.

Finally the user supplies an initial rule file containing up to 64 rules. Initially there may be none, in which case the program will generate some at random.

A rule is represented by a fully bracketed Boolean expression ended by a dollar sign, such as

```
((#4 GE 20) OL ((#4 LT 10) AN (#17 NE 0))) $
```

which states that variable 4 (#4) should exceed or equal 20 or that both variable 4 should be less than 10 and variable 17 not zero for the rule to give a positive (true) result.

The operators are as follows.

EQ	arithmetic equality
NE	arithmetic inequality
GT	greater than
LT	less than
GE	greater than or equal to
LE	less than or equal to
OL	logical disjunction (Inclusive or)
AN	logical conjunction (and)
NO	negation
PLUS	addition
LESS	subtraction
BY	multiplication
OVER	division

(The odd names such as AN and OL were chosen to avoid a clash with Pascal predefined operators.)

Arithmetic is integrated with logical evaluation because the three truth values are +1, 0 and -1. If a rule yields a final value outside the logic range it will be truncated to the nearest extreme. Arithmetic subexpressions are not truncated (unless they would cause overflow).

2.2 The LEAF Program

LEAF is far simpler. It takes a datafile in the same format as the training set -- the only difference being that the actual classes need not be known, zero indicating unknown class membership -- and runs a rule file on it. The user specifies how many rules to use: these are always left ordered by HERB with the best first. LEAF can be requested to produce: (1) a listing of all cases with predicted class, and actual class and score if known; (2) a summary of the performance of each rule and all the rules jointly; (3) an ordering of cases by rule consensus from most likely Yes to most likely No.

Notice that the rules produced by HERB can be applied by a person. LEAF is merely a convenience. Contrast this with the linear functions with coefficients expressed to 8 or 10 decimal places output by conventional discriminant analysis packages: no one in their right mind would try to use those without machine assistance.

3. HOW HERB WORKS

HERB attempts to mimic evolution by natural selection. Its 'organisms' are the rules and their survival depends on how well they categorize the samples in the training set.

It runs for a number of generations, chosen by the user.
A generation consists of one run through the data during which each rule is evaluated on every case and scored according to the payoff matrix. The rules are then ranked by total score with the best rules at the top, i.e. those with the highest score.

The scoring formula is actually

((GOODNESS-MINSCORE)*100*GFACTOR) / (MAXSCORE-MINSCORE) - SIZE

where MINSCORE and MAXSCORE are the lowest and highest scores possible, GOODNESS is the accumulated payoff and SIZE is the size of the rule measured by counting nodes (terms or subexpressions). What this means is that a long-winded rule scoring the same as a more concise one will be ranked lower. Remember we are treating the rules as organisms: the larger animals need more 'food'. GFACTOR can be set by the user to alter the balance between goodness and size. A high GFACTOR asks for a good rule at, almost, any price; a low setting is a bias towards brevity.

Having been ranked thus, the breeding begins. The top quarter (25%) are left alone. They are good enough to survive untouched. The second quarter are all subjected to a procedure GROW which adds a node composed at random. For example, GROW on

((#1 OL #2 EQ 0)) GT 62)

might produce

((#1 PLUS 5) OL (#2 EQ 0)) GT 62)) .

Rules in the third quarter are subjected to a procedure named SLIM which is the obverse of GROW; they lose a randomly selected term or subexpression. They have survived but are suffering from 'malnutrition'. Finally the bottom 25% are subjected to a procedure called KILL which, squeamish readers may be assured, causes no pain.

To replace the dead rules new ones are formed by mating together elements from the top half of the list. Internally the rules are held as binary trees. The MATE procedure takes a random subtree from one parent rule selected at random from the upper half and combines it with another chosen likewise. The two parts are then linked by a randomly selected connective to give a fully formed expression. For example, the mating of

((#4 GT 62) AN (#3 EQ 0))

with

((#17 BY -2) PLUS ((#15 GT 5) OL (#2 LE #8)))

might result in

((#4 GT 62) LESS #8) .

The next step is to apply the MUTATION procedure to a few (randomly selected) of the lower 7/8ths of the rule list.
This can do various things like altering terms, swapping subtrees, altering operators and so forth. (The top 1/8th is inviolate: rules that high can only be changed if a better 'strain' displaces them.)

Finally, procedure TIDY is applied to all rules.
This cuts down redundancies such as double negatives,
expressions with a constant value and so on, leaving the pruned tree
with the same value but expressed more succinctly. The result of
TIDYing

(((5 BY 4) GT 16) AN (#17 EQ #8))

would be

(#17 EQ #8)

since (5 BY 4) = 20 and (20 GT 16) = +1 (true).

Then the next generation begins. The process continues
for the required number of generations, and then the new rules are
printed onto the output file.

4. SOME TESTS OF HERB

The question is: does it work?

To establish a comparative standard the discriminant analysis
function of the SPSS package on the DEC System-10 library was run
with the hospital admission data. It produced two linear
functions of seven variables plus a constant. Both these
functions are to be evaluated for each case and if function 1
gives a higher value the sample is assigned to group 1 (living)
whereas if function 2 gives a higher value the sample is assigned
to group 2 (dead). (There were 70 survivors and 43 deaths,
but this information was not used to weight the prior probabilities.)

The diagnostic variables chosen were, in descending order of
importance, numbers 6 (mean arterial pressure), 9 (mean venous
pressure), 4 (shock type), 14 (urinary output), 10 (body surface
area). 15 (plasma volume index) and 16 (red cell index). All were
positively loaded on function 1 except 9 (venous pressure). The
CPU time to generate these results was 2.85 seconds.

When re-run on the training-set data the discriminant functions
correctly classified 75% of the cases. The mistakes were:
16 of group 1 classed as group 2; 12 of group 2 classed
in group 1.

The HERB program was then run on the same data, starting
completely from scratch -- i.e. with no pre-determined rules.
For all the tests the number of rules was fixed at 48. After 111
generations a run of LEAF indicated that the top rule was
correctly grouping 73% of the cases in the training set.
This took about 2 minutes of run-time.

After 500 generations the top rule was correct on 81% of the cases
(counting a 0, or don't-know, as incorrect as well as
any outright misclassifications).

The top rule at this stage was

(#6 GE (61 LESS #14))

where #6 is mean arterial pressure and #14 is urinary output.
What it says is that if mean arterial pressure (mm Hg) is greater
than or equal to urinary output (ml/hr) subtracted from 61 the
patient should survive, otherwise the patient is likely to die.
Its mistakes were: 2 survivors classed as group 2; 20 deaths
classed as group 1. (The payoff matrix could have been adjusted
if these different kinds of error were not equally costly, as no
doubt would be the case in practice.)

It is notable that already we have a rule that is better
than the linear discriminant functions; and so much simpler
that a hospital orderly could easily apply it. (Is this a danger?)

Perhaps statisticians, who are on the whole quite content to computerize
techniques worked out by Pearson and Fisher over 50 years ago
and who tend to regard even Bayesian decision-making
as an exciting but not very respectable novelty, should wake up
to the potential of today's expert systems.

A second test was run on data concerned with the physique of male athletes.
Here the data was the age (#1), height in inches (#2),
weight in pounds (#3) and race (#4) of the medallists
in the running and walking events of the 1968 Mexico Olympic Games.
Race was either 0 (white) or 1 (black). (One Japanese was arbitrarily
assigned to race 0 and Mohammed Gamoudi, who appeared twice
by virtue of winning medals in two different events, was classed as 0
the first time and 1 the next: he is Tunisian.)

The aim was to arrive at a rule that would distinguish
the sprinters from the long-distance men on the basis of the data
about age, height, weight and race. The events were actually put
into 5 classes, from shortest to longest.

Class	Events
1	100m, 200m, 110m hurdles
2	400m, 400m hurdles
3	800m 1500m 3000m steeplechase
4	5000m, 10000m, 20km walk
5	Marathon, 50km walk

The various payoffs were assigned accordingly.

	Actual Class				
Rule Decision	1	2	3	4	5
-1	2	1	0	-1	-2
0	0	0	0	0	0
+1	-2	-1	0	1	2

A decision of +1 is interpreted as long-distance competitor,
-1 as sprinter.

After 666 generations the top rule was

((155 LESS #3) PLUS (-5 BY #4))

which was only making one mistake on the 52 samples in the training set.
What it says, in brief, is that if you are white and weigh over 155 pounds you
are
a sprinter, if you weigh less you are a distance runner;
if you are black and weigh over 150 pounds you are a sprinter,

otherwise you are a long-distance runner.

As a test 12 gold medallists from the 1980 Moscow Olympics were rated by this rule. This was fresh data, not used in the training phase. All were correctly categorized except Pietro Mennea who, at 150 pounds, is a bit light for a white sprinter.

N.B. These figures apply to Olympic athletes:
just because you weigh over 155 pounds do not get the idea
that you are a match for Allan Wells!

5. REMARKS

I see three justifications for this kind of exercise.
Firstly, it is interesting in its own right; secondly, the rules behave in an interesting fashion; and thirdly, it seems to work.

In the first place it is fun to try a little abstract gardening, growing an orchard of binary trees. And it might be fruitful in another sense. After all, we are only here by courtesy of the principle of natural selection, AI workers included, and since it is so powerful in producing natural intelligence it behoves us to consider it as a method for cultivating the artificial variety.

The second justification is the surprisingly lifelike behaviour of the rules themselves. It can be appreciated by looking at the top four rules produced by HERB on the hospital admissions data after 1, 11, 111 and 1111 generations.

1 generation	Age	Score
(#1 PLUS 0)	1	27
(2 PLUS 0)	1	27
(#1 PLUS 0)	1	27
(#1 PLUS 0)	1	27
[27 = chance expectation]		

11 generations		
(NO (#16 LE -1) BY (#6 GT 53)))	5	49
(#6 GE #1)	3	40
(((#16 LE -1) BY (#6 GT 53)) LT ((#1 EQ #11) NE -10000))	4	39
(#1 PLUS 0)	10	27

111 generations		
(137 OVER (#6 GT 53))	77	51
(135 OVER (#6 GT 53))	55	51
(1 OVER (#6 GT 53))	41	51
(137 OVER (#6 GT 53))	33	51

1111 generations		
(#6 GE (61 LESS #14))	691	69
(#6 GE (62 LESS #14))	502	69
((61 LESS #14) LE #6)	479	69
((61 LESS #14) LT #6)	478	69

What we see here is the appearance (and subsequent disappearance) of dominant 'species'. Each type flourishes for some time until quite suddenly supplanted by a new and superior line -- typically a mutation of one of its own offspring. When this happens the extinction of the more primitive forms is rapid and complete.

It seems that once a rule fastens on a particular indicator variable or combination of variables it will give rise to several copies or near-copies forming a family which thrive until a better rule appears, possibly using an entirely different set of indicators. It is as if the new variety have found a more nourishing 'diet'.

There is nothing to prevent the user inserting a man-made rule at any stage; indeed it is salutory to do so since all trace of it is normally lost within a few generations. (If it were easy to cast your eye over a large mass of figures and extract an efficient classification rule for the cases there would be little need for this kind of program.)

The third point is that the system works quite well, even though this is version 1.0 of the program. The rules produced are short and to the point, though it is fair to mention that the computing cost is quite heavy -- almost 2 minutes of runtime per 100 generations on the DEC System-10 (KL 10 processor) on the hospital data.

As BEAGLE is quite successful on toy databases the reader with gambling instincts may care to participate in a little field testing on far more messy data. For what it is worth, here is the top rule produced by HERB after 400 generations on a data file containing 1000 English and Scottish League and Cup football results (1979-80).

```
      (NO ((#58 OVER #45) AN #77))
```

Its value is meant to be true (positive) for drawn games, negative otherwise. The variables are: #58 the away team's ground capacity (in thousands of spectators) subtracted from the home team's crowd capacity (thousands); #45 goals scored by the away side in their last away game; #77 difference formed by adding home team's home goals scored in their last 8 home matches to away team's goals conceded in their last 8 away games and subtracting home team's goals against plus away team's goals for in the same matches.

6. FUTURE DIRECTIONS

BEAGLE is still at a prototype stage, and can be considerably improved. One planned enhancement was mentioned in section 2.1 -- allowing the multi-state logic range to be specified by the user.

A second extension that would not be difficult to implement would be to allow floating-point arithmetic as well as integers, though the interaction with logical values would have to be carefully considered first. It might be an opportunity to introduce 'fuzzy logic' [Zadeh, 1965] at the same time. (HERB and LEAF already use a 'slightly fuzzy' comparison scheme such that 64

GE 65 is not quite so false as 60 GE 65, but the usefulness of this has not been assessed.)

Another planned improvement is the inclusion of additional operators. The MOD (remainder) function will be one, but more important will be the pair SO .... OS to allow constructions of the form

(B SO (Al OS A2))

which serves for

if B then Al else A2

and will give a rudimentary programming ability. Of course this highlights the fact that the rules are really programs in a special-purpose language, which might lead to the conclusion that the system should ultimately generate LISP functions. But this is rather a distant goal. It would remove all restrictions, but whether HERB or anything like it could cope with the extra power remains to be seen. Probably a compromise, such as generalizing the rules as far as production systems of a limited complexity, would be more manageable.

In effect the rules as they stand simulate single-celled organisms, without specialization of function. To move on to a hierarchic structure, corresponding to multi-cellular animals. HERB would need 2nd-level organisms (strategies) with 'green fingers' whose welfare depended on successfully managing the first-level ones.

A more serious need is to make greater use of information provided by bad rules, rather than just discarding them and quite possibly regenerating them later. This is a major weakness, and all the obvious remedies (e.g. a rote memory of bad rules) would introduce considerable overheads.

7. CONCLUSIONS

I would like to present Naturalistic Selection as a viable AI technique. This is not to say it is a panacea. I suspect that there is always a better way (cheaper and/or quicker); but no single method is more appropriate for 'satisficing' [Simon, 1969] in such a wide variety of problems.

On the credit side Naturalistic Selection is absolutely general. A user can hurl any data at HERB, however nonlinear, however 'noisy', however much it violates the assumptions about distribution and scaling underlying most statistical tests, and get a reasonable set of discrimination rules in reasonable time. And since those rules are public and comprehensible, not arcane technocratic black magic, man-machine cooperation is facilitated. The human can do some of the hypothesizing (which people are good at) leaving the testing (which people are bad at) to the computer.

For example, to classify the test data used in this paper a sequential decision procedure such as that proposed by Hunt [Hunt et al., 1966] might have been more economical; but the trouble with stepwise algorithms which yield a discrimination net or progressive filter network is their susceptibility to noise in the data. They work best in situations where there can be no error in the training data and where the variables have rather few discrete values (e.g. chess endgames). HERB, though rarely if ever optimal,

will almost always come up with something usable.

Lastly there is the matter of image. Is the designer of expert systems to be seen as a soulless white-coated machine-minder or as someone who, for the first time since the expulsion from Eden, is not merely picking new fruit from the forbidden (Binary) Tree of Knowledge, but actually making it grow?

8. REFERENCES

Afifi & Azen -- "Statistical Analysis; A Computer Oriented Approach"; Academic Press (1972).

A.G. Bell -- "Games Playing with Computers": Allen & Unwin (1972).

Bernstein & Rubin -- "Artificial Evolution of Problem-Solvers"; American Behavioral Scientist (May, 1965).

Buchanan &. Mitchell -- "Model-Directed Learning of Production Rules" in Waterman & Hayes-Roth (1978).

Fogel et al. -- "Artificial Intelligence through Simulated Evolution": Wiley (1966).

Hunt et al. -- "Experiments in Induction": Academic Press (1966).

Donald Michie (ed.) -- "Expert Systems in the Micro Electronic Age": Edinburgh University Press (1979).

Gordon Pask -- "An Approach to Cybernetics": Hutchinson (1961).

J.R. Quinlan -- "Discovering Rules by Induction from Large Collections of Examples": in Michie (1979).

O.G. Selfridge -- "Pandemonium, a Paradigm for Learning": NPL Symposium on Mechanization of Thought Processes, HMSO (1959).

Herbert Simon -- "The Sciences of the Artificial": MIT Press (1969).

Waterman & Hayes-Roth (eds.) -- "Pattern-Directed Inference Systems": Academic Press (1978).

L.A. Zadeh -- "Fuzzy Sets": Information & Control 8 (1965).

[Pascal source listings of HERB and LEAF are available on request from the author at: Maths Dept., Polytechnic of North London, N7 8DB.]

</body>

<tail>

by=Richard Forsyth
from=typescript
refline=Forsyth, R.S. (1981). BEAGLE -- a Darwinian approach to pattern
recognition: Kybernetes, 10. 159-166.
year=1980
textype=tech
area=computing
note=written 1980, published 1981
</tail>