NEWBOOLE: A Fast GBML System

Pierre Bonelli
bonelli@lri.lri.fr
Equipe Inférence et Apprentissage
LRI
bât. 490
Université de Paris-Sud
91405 Orsay, France

Alexandre Parodi
parodi@lri.lri.fr
INFODYNE SARL
10, rue de la Paix
75002 Paris
France
Tel.: (33)-(1)-42-61-56-08

Sandip Sen
sandip@dip.eecs.umich.edu
EECS Department
University of Michigan
Ann Arbor, MI 48109
USA

Stewart Wilson
wilson@think.com
The Rowland Institute for Science
100 Cambridge Parkway,
Cambridge, MA 02142
USA

Abstract[1]

Genetics based machine learning systems are considered by a majority of machine learners as slow rate learning systems. In this paper, we propose an improvement of Wilson's classifier system BOOLE that shows how Genetics based machine learning systems learning rates can be greatly improved. This modification consists in a change of the reinforcement component. We then compare the respective performances of this modified BOOLE, called NEWBOOLE, and a neural net using back propagation on a difficult boolean learning task, the multiplexer function. The results of this comparison show that NEWBOOLE obtains significantly faster learning rates.

1 Introduction

In recent years, Genetics Based Machine Learning (GBML) has received increasing attention from the ML community due to the emergence of Classifier Systems. However, despite the demonstrative results obtained by various researchers with Classifier Systems (CS) [Goldberg, 89], the slow learning rates that are usually observed have considerably affected their credibility. The results reported in this paper, using an improvement of Wilson's BOOLE system, tend to show that convergence speed of GBML systems can be greatly improved.

[1] This research was partially supported by MRT through PRC IA.

2. Slow learning rates in GBML systems.

A number of realizations in the domain of CS have shown their undisputed ability to learn. Classifier Systems [Holland, 86] form a family of inductive learning systems which acquire *rules incrementally*. However, these realizations all share the common drawback of slow rate learning when compared to other widespread learning algorithms such as decision tree classification (such as ID3) or neural net back propagation.

Wilson's classifier system BOOLE [Wilson, 87] is an example of such a realization. BOOLE is an *incremental* learning system that learns intricate boolean functions such as logic multiplexers. However, more recently, Quinlan [Quinlan, 88] compared the respective performances of an improved version of the ID3 algorithm (C4) and BOOLE on the multiplexer problem, and evidenced a much faster convergence rate with C4. We show in this paper that this drawback can be greatly weakened by modifying the reinforcement component of the original algorithm. Furthermore, C4 *is non incremental*, and as Booker mentions in [Booker, 89], having access to all the examples at once is a definite advantage. Therefore, in our experiments, we decided to compare the improved version of BOOLE (NEWBOOLE) with a widely used incremental learning system: a neural net using back-propagation.

3. The BOOLE Classifier System

We now present BOOLE with some detail concerning the parts that have been changed, in order to explain the NEWBOOLE system in the following section.

BOOLE is a simplified version of the standard Classifier System (CS) which was designed by Wilson to test the ability of a GBML system to learn difficult boolean functions.

Like any CS, Boole maintains a population of classifiers (which can be thought of as bit-level zero order rules) according to Darwinian evolution principles. However, classifiers are not chained; they directly provide an output and the decision is made within a single step during recognition; consequently there is no message list nor Bucket Brigade Algorithm. Thus each classifier consists of a condition (taxon) and an action which are fixed length strings over the {0,1,#} alphabet.

Like other CS, BOOLE has the following components:

1/ **Performance component**: in the performance cycle, an input string is presented to the system, the match set M of all classifiers whose taxa match the input string is formed, and a single classifier from M is selected (using a probability that is proportional to its strength) whose action is output as the system's decision.

2/ **Reinforcement component**: this component modifies the strengths of classifiers according to performance level:

a/ Form the action set [A] consisting of classifiers from [M] whose action is the same as the chosen action; the remaining members of [M] form the set Not[A];

b/ Deduct a fraction e from the strengths of all classifiers in [A];

c/ * If the system's decision was correct, distribute a payoff quantity R to the strengths of [A]; but
* 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);

d/ Deduct a fraction t from the strengths of Not [A].

The distribution of payoff is done so that rules which have many # 's (thus more general) are favored.

3/ **Discovery component,** which modifies the classifier population according to Holland's genetic algorithm [Holland, 75] and employs reproduction, genetic operations (crossover and mutation), and deletion.

BOOLE's version of the genetic algorithm is quite particular in the sense that only one offspring is added per invocation of the genetic algorithm. In this context, the parameter ρ will represent the average number of invocations of the genetic algorithm per cycle (i.e. the number of offspring added per cycle). For the detailed algorithm, please see [Wilson, 1987].

Wilson experimented in [Wilson, 87] with this system using a highly disjunctive function, the multiplexer function, also used by Barto [Barto,1985]. In the case of the "6-multiplexer", for each six-bit input string (a_0, a_1, x_0, x_1, x_2, x_3), the boolean expression of the output is:

$$F_6 = \neg a_0 \cdot \neg a_1 \cdot x_0 + a_0 \cdot \neg a_1 \cdot x_1 + \\ + \neg a_0 \cdot a_1 \cdot x_2 + a_0 \cdot a_1 \cdot x_3 \qquad (1)$$

Figure 2 shows an experiment in which BOOLE learned to respond correctly to this problem. The parameters used for this experiment are given in section 4.

At each cycle, an example is chosen at random and is presented to the system. The graph plots the system's *average score* which is the percentage of correct decisions over the past 50 cycles versus the number of cycles since the experiment began.

The results obtained by BOOLE show that a "rather difficult disjunctive incremental learning task" can be solved by GBML. However, the learning rate is extremely slow, as was pointed out by Quinlan in [Quinlan, 88], where he compares the respective performances of BOOLE and C4 on the same multiplexer task.

4. The NEWBOOLE CS

NEWBOOLE is a CS derived from Boole which obtains much faster learning rates. We examine in this section the improved learning algorithm.

4.1 A new payoff strategy: "Symmetrical payoff-penalty

BOOLE's reinforcement component, under the "payoff-penalty" reinforcement regime ($p \neq 0$) adjusts classifier strengths in the following way:

- if the system's decision is correct, distribute a quantity R to the strengths of the Actionset [A].

- if the system's decision is false, penalize the strengths of [A] by deducting a fraction p from their values.

- finally, whether the system's decision is correct or not, deduct fractions e and t respectively from the strengths of [A] and Not[A].

Thus, following each performance cycle, *only the strengths of [A] are ajusted according to the system's performance.*

However, once we know that [A] contains accurate classifiers, we also know that Not[A] only contains inaccurate classifiers; in this case it would make sense to penalize the rules in Not[A]. This acknowledgement led us to a "symmetrical payoff-penalty" algorithm, in which we respectively reward and penalize the accurate and inaccurate classifiers present in the Matchset.

The new reinforcement component is the following:

1/ Form the subset of [M] consisting of those classifiers whose action is accurate; this is the correct set [C]. The remaining members of [M] form the set NOT[C].

2/ Deduct a fraction e from the strengths of [C].

3/ Since [C] contains the accurate classifiers, distribute a payoff quantity R to the strengths of [C].

4/ Since Not[C] contains the inaccurate classifiers, deduct a fraction p from the strengths of Not[C].

Thus, the effect of the reinforcement component can be written as:

$$S_{[C]}(t+1) = (1-e) \times S_{[C]}(t) + R \qquad (3)$$

$$S_{Not[C]}(t+1) = (1-p) \times S_{Not[C]}(t) \qquad (4)$$

where $S_{[C]}$ and $S_{Not[C]}$ are respectively [C]'s and Not[C]'s total strengths.

This new algorithm constitutes a clear departure from Boole: indeed, if we have several possible output values then the knowledge of the correctness of the output of each classifier from the match set is used. This information can be provided by the knowledge of the correct output for each example, as is done in most learning systems. However, this does not make any difference with boolean functions such as the Multiplexer since only two values are possible: if one is known as wrong, then the other one is right.

As in Boole, the payoff R to [C] is distributed by a biased distribution function D, which favors more general rules (i.e. with many "don't cares" #) as follows.

First, the generality of each classifier i of length L is computed as:

$$g_i = \frac{\text{number of \#'s in i}}{L} \qquad (5)$$

Let us define:

$$d_i = 1 + G \times g_i \qquad (6)$$

where G is a "generality emphasis" parameter.

Then, the portion of reward R_i that is given to classifier i becomes:

$$R_i = D(i) \times R = \frac{d_i}{\sum_i d_i} R \qquad (7)$$

4.2 Experiments with NEWBOOLE

We experimented with NEWBOOLE using the multiplexer problem, our main concerns being on the one hand to compare BOOLE's and NEWBOOLE's respective performances, and on the other, to compare NEWBOOLE and a neural net using Back Propagation (BP). We describe two sets of experiments, one with the 6-multiplexer, the other with the 11-multiplexer.

Each experiment was conducted by making 4 independent runs with different random initializations and averaging the values over these runs.

The table below gives a complete description of the genetic experimental parameters used.

4.2.1 NEWBOOLE and the 6-multiplexer problem.

1/ In the first experiment (Figures 1, 2), we tested NEWBOOLE's performance using exactly the same parameter values as in BOOLE in order to evaluate the effect of the change in the reinforcement component. As it can be seen, the results are quite demonstrative: without any "parameter tuning", we are able to enhance importantly the system's learning rate to 97.3 % after only 2000 trials. The learning rate only takes into account the system's performance.

The lower plot in Figure 1 shows a quantity called the relative solution count, equal to the relative number of instances of the *solution set* [S6], which represents the minimal set of classifiers capable of solving the problem perfectly.

Param. Experi.	e	χ cross-over rate	μ mutation rate	G generality enforcement	R reward	p penalty coef.	ρ renewing	t	P population	deterministic output
Boole 6Mux Fig1,2	0.1	0.125	0.001	4	1000	0.8	1	0.1	400	NO
NewBoole 6Mux (Fig 1, 2)	0.1	0.125	0.001	4	1000	0.78[1]	1	////	400	NO
NewBoole 6Mux (Fig. 3)	0.1	0.5	0.001	4	1000	0.95	4	////	400	YES
NewBoole 11 Mux (Fig. 4)	0.1	0.5	0.001	4	1000	0.95	4	////	1000	YES

Table 1: Experimental parameters for Boole and NewBoole.

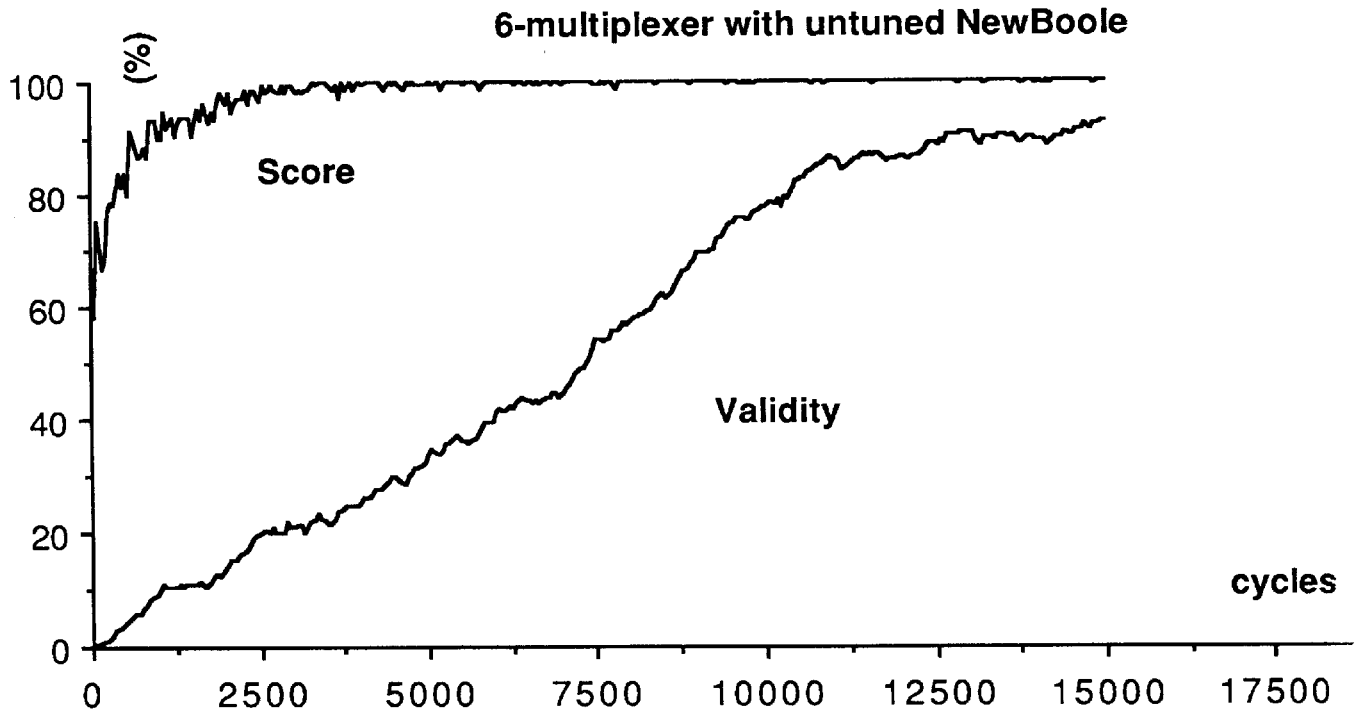

Figure 1: Untuned New Boole with stochastic output

[1] p is taken as 0.78 and not 0.80 in order to ensure total equivalence between the experiments with Boole and untuned Newboole, since the parameter t is no longer used in the Newboole algorithm.

Hence, this ratio is a measure of the *validity* of the population: the predominance of [S6] in the evolved population would show that the system is capable of finding the best among the accurate classifiers.

It is interesting to notice that eventhough the system attains quasi-perfect response (99.8) after 3200 cycles, the validity (solution count) continues to grow at a similar rate than in BOOLE.

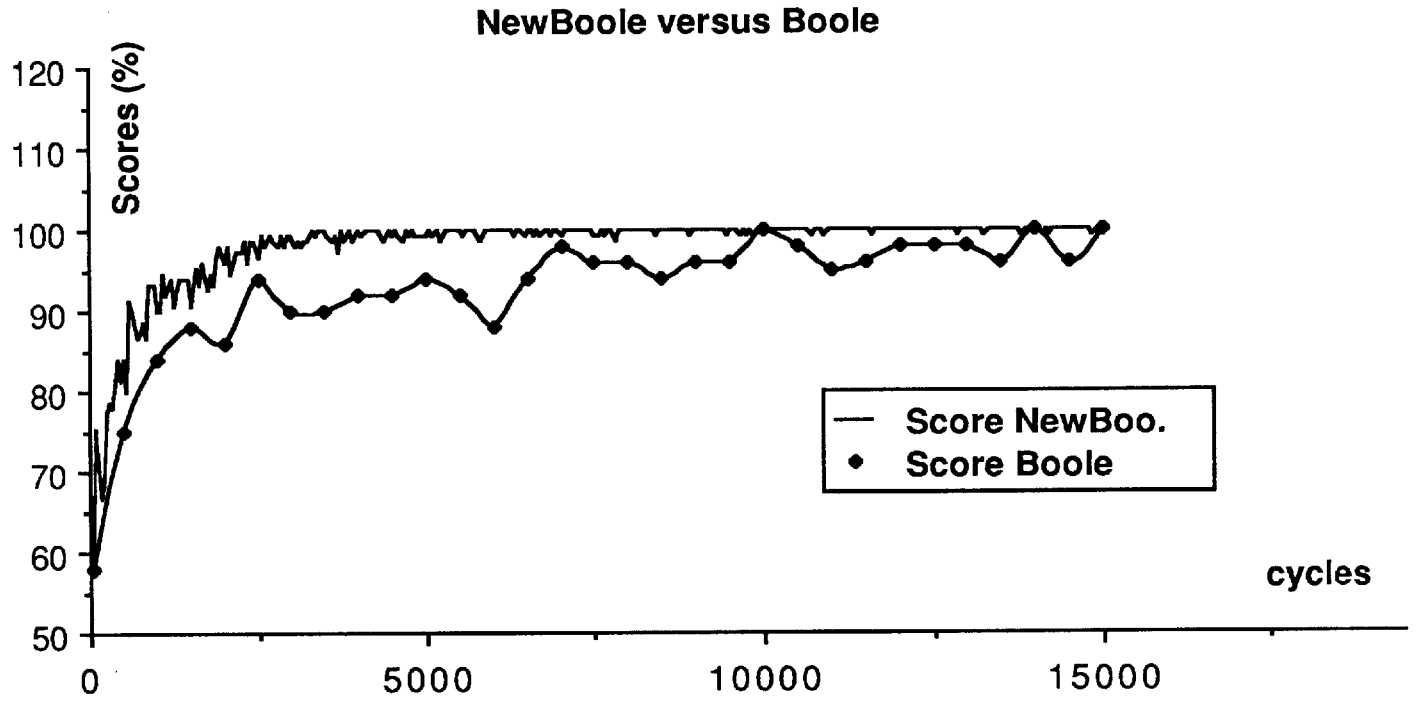

Figure 2: Comparison between untuned NewBoole and Boole.

Cycles	Score (%) Boole	Score (%) untuned NewBoole	Error(%) Boole	Error (%) NewBoole
0	48	50	52	50
500	75,9	86,7	24,1	13,3
1000	84,6	92,9	15,4	7,1
3200	91,2	99,8	8,8	0,2
5200	93,9	100	6,1	0
12000	97,3	100	2,7	0

Table 2: Comparison between Boole's and NewBoole's convergence rates

2/ We present in the second experiment (Figure 3) a "tuned" version of the NEWBOOLE algorithm.

* Firstly, we modified the values of certain parameters in order to speed up the learning process (see Table 1).

* Secondly, we modified the Performance Component in the following way: instead of selecting the "decision" classifier probabilistically, we systematically picked the highest ranked classifier in the match set: this deterministic selection affects in no way the learning process, since the Correct and NotCorrect sets are not determined in function of the selected classifier. This modification, as noted in [Booker, 89], permits a more steady convergence level; furthermore, since we are comparing NEWBOOLE with a deterministic algorithm (Back Propagation), it seemed logical to include some "determinism" in the algorithm.

We obtained an impressive learning rate after only 800 trials by modifying the value of the fraction deducted from the set of inaccurate classifiers ($p = 0.95$), the frequence of invocation of the genetic algorithm per cycle ($\rho = 4$), and the crossover rate ($\chi = 0.5$). This constitutes approximately a 17 fold improvement over Boole's original convergence rate.

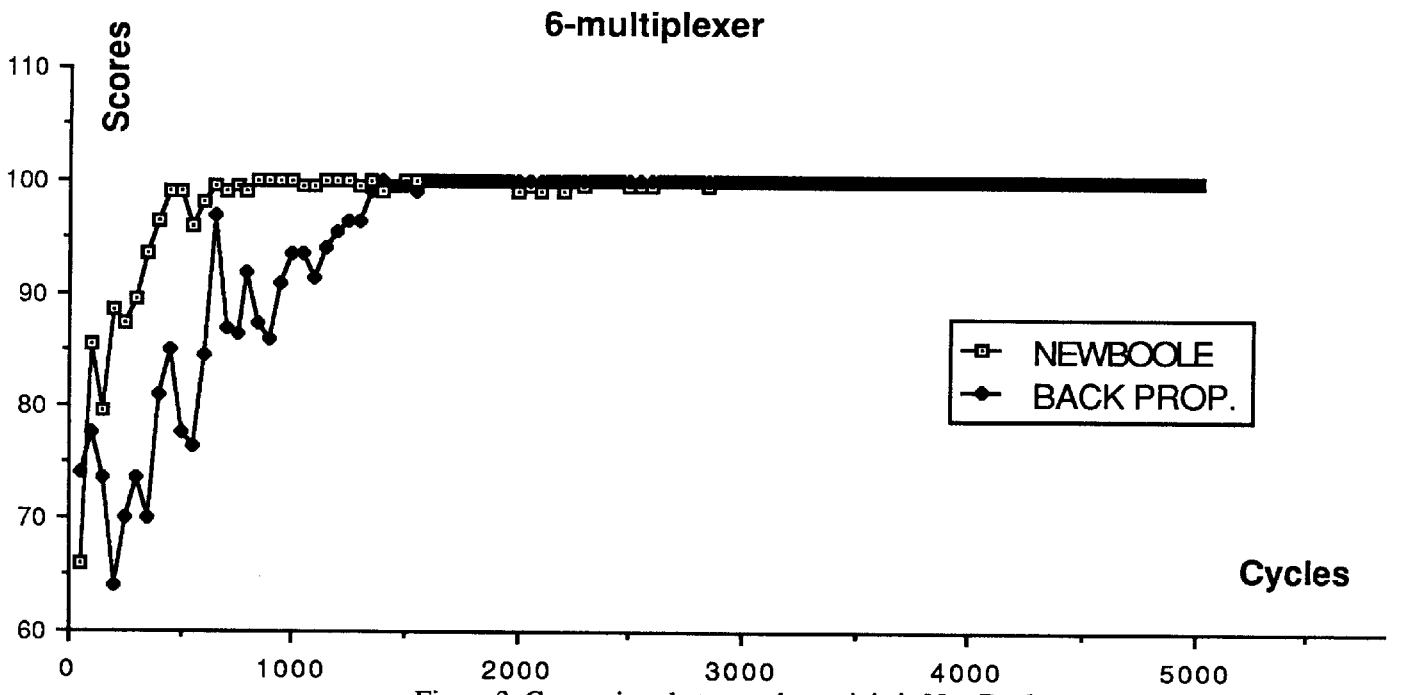

Figure 3: Comparison between deterministic NewBoole and Neural Net with BP on the 6 multiplexer problem

3/ Also in Figure 3, we present an experiment using a Neural Network with a Back-Propagation learning algorithm. The architecture (6:20-20-10-10:1)[1] and the parameters were tuned for this problem. Indeed, we see that convergence is reached after 1600 trials.

Of course, we noticed that a more complex network (6:100-50-40-30:1) converges within 900 trials. However, the performance is not better than with NEWBOOLE, and the memory occupied (6 x 100 + 100 x 50 + 50 x 40 + 40 x 30 + 30 = 8830 connections = 8830 x 4 = 35320 bytes) is unreasonably beyond what our population occupies (400 classifiers of 7 units each with 2 bits per unit = 800 bytes).

Therefore, when comparing NEWBOOLE with a Neural Net (NN) using BP, we restricted ourselves to reasonable networks.

[1]This notation means that the multilayered network has 6 inputs, then two layers of 20 cells, then two layers of 10 cells, and one output layer of one cell.

In all the networks, the parameters of each cell depend on the number of cell inputs N_{in}:
learning rate $\varepsilon = 0.1/\mathrm{sqrt}(N_{in})$, decay $\delta = 0$, noise $\theta = 0$, momentum $\alpha = 0$; weights $W_{ij}(0)$ are initialized randomly over the interval $[-1.5/N_{in}; 1,5/N_{in}]$.

Please also note that the simplest NN that can solve our problem (6:4:1) needs 7500 cycles to converge; this compares with the number of cycles NEWBOOLE needs to find the minimal set (around 5000 cycles for 80 % of minimal rules; the other rules have a very low strength and can easily be removed); however, NEWBOOLE finds this set automatically, whereas the NN architecture had to be provided at first.

4.2 NEWBOOLE and the 11-multiplexer problem.

Figure 4 shows the results obtained using NEWBOOLE to solve the 11-multiplexer compared with those obtained using the following neural net: (11:40-40-20-20:1).

The population was increased by a factor of 2,5 in order to fit the considerably larger classifier search space which grew by a factor of $3^{11} \times 2 / (3^6 \times 2) = 243$.

The number of links of the neural net rose much more (by a factor of 3260 / 730 = 4,5) than the population size.

Nonetheless, one notices that NEWBOOLE still converges at a faster rate than the neural net.

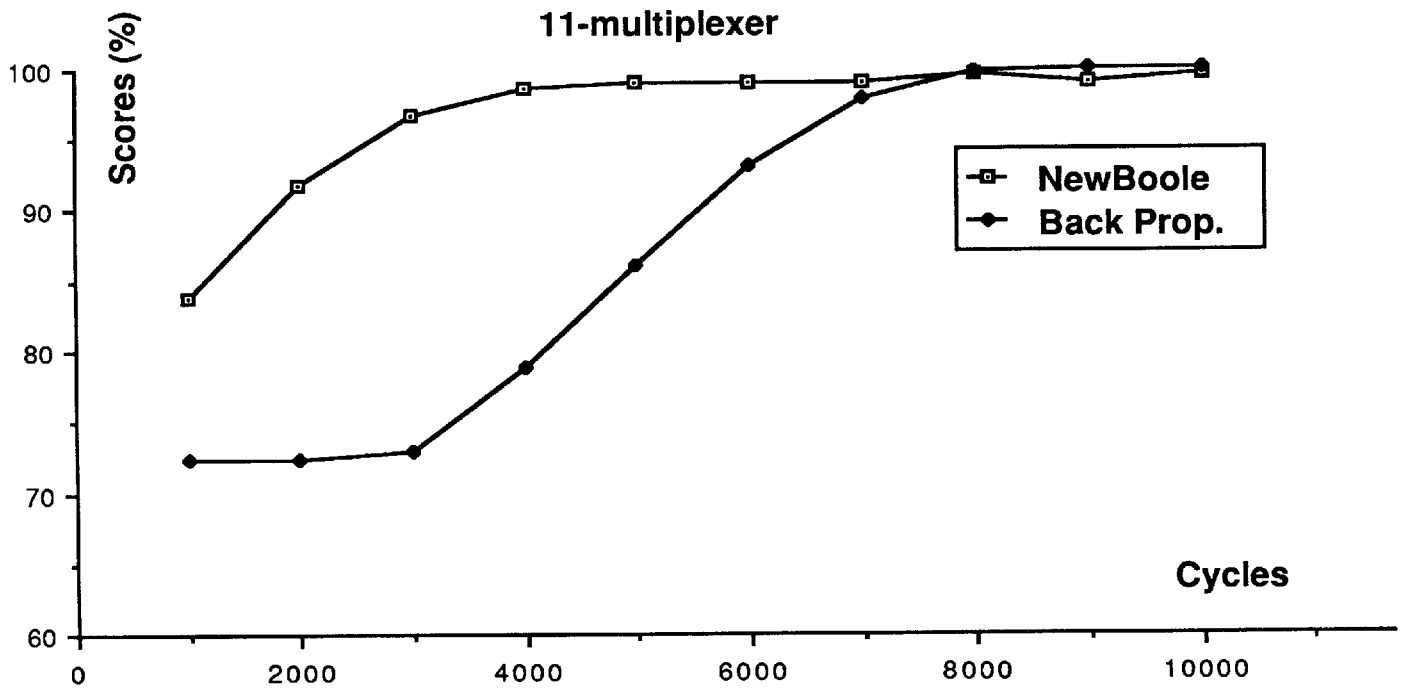

Figure 4: Comparison between deterministic NewBoole and Back-Propagation on the 11-multiplexer problem.

(symmetrically smoothed over 1000 cycles)

5. Conclusion

In this paper, we presented an improvement to the Boole system; indeed, we showed that convergence speed could be drastically improved, thus showing that GBML systems can learn far faster than were portrayed by Quinlan in [Quinlan, 88]. Furthermore, the comparison with an other incremental learning algorithm, the widely used neural net with back-propagation, showed that NEWBOOLE converges at least as fast. The basic difference between GBML and Connectionist learning thus seems to reside in the fact that Classifier Systems are rule based systems which provide comprehensive solutions, whereas neural networks merely provide sets of numerical coefficients without any semantic meaning.

6. References

[Booker, 89] "Triggered Rule Discovery in Classifier Systems", in ICGA 89, Morgan Kaufmann.

[Goldberg, 89] "Genetic Algorithms in search, Optimization, and Machine Learning", Addison Wesley.

[Holland, 75] "Adaptation in natural and artificial systems", Ann Arbor: University of Michigan Press.

[Holland 86] "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems" In "Machine Learning: an Artificial Intelligence Approach, Vol 2, Michalski R.S., Carbonell J.G., Mitchell T.M. eds, Morgan Kaufmann, Los Altos (CA), 1986.

[Quinlan, 88], "An Empirical Comparison of Genetic and Decision-Tree Classifiers", Proceedings of the fifth International Conference on Machine Learning.

[Wilson,87] "Classifier Systems and the Animat Problem", Machine Learning 2,199-228, 1987.

[Wilson & Goldberg, 89] "A critical Review of Classifier Systems", in ICGA 89, Morgan Kaufmann.