ETL-TR-93-25
received Oct. 18, 1993

# Evolutionary Learning of Boolean Concepts: An Empirical Study

Hitoshi IBA[1] Tatsuya NIWA[2] Taisuke SATO[1]

1)Machine Inference Section,
2)Computational Models Section,
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba-city, Ibaraki, 305, Japan
{iba,niwa,sato}@etl.go.jp, +81-298-58-5918

**Abstract** This paper explores various problems which arise with the learning of Boolean concepts. Boolean concept learning is an important part of more traditional machine learning. We begin by comparing the learning performances of stochastic or evolutionary methods, namely neural networks (NN), classifier systems (CS) and adaptive logic networks (ALN). The learning and testing tasks were performed on two sets of examples independently drawn with uniform probability. This learning environment was strictly designed according to computational learning theory. The number of learning examples we used for training and testing was derived from the PAC learnability formula [Anthony92]. Although it is widely believed that neural networks provide robust learning methods for the XOR and multiplexor problems, we have confirmed that they are inferior to other evolutionary methods when it comes to learning more complex Boolean concepts, because of their distributed characteristics. We also discuss both merits and demerits of the above three methods. Finally we discuss a new evolutionary learning method which overcomes the above difficulties.

## Extended Summary

Boolean concept learning has is an important part of machine learning. Although earlier algorithmic approaches to Boolean concept learning such as decision trees [Quinlan86] or enumeration [Anthony92] proved to be sound and complete, they suffered from computational complexity. Alternatively several stochastic or evolutionary methods have been proposed, which aim at improving efficiency by using probabilistic search at the expense of completeness. Among them are neural networks (NN) [Rumelhart86], classifier systems (CS) [Wilson87] and adaptive logic networks (ALN) [Armstrong79]. These can be classified roughly into analog approaches (NN) and into digital approaches (ALN, CS). However there have been few comparative studies between their performances from the viewpoint of computational learning theory [Anthony92]. In order to evaluate the merits and demerits of these methods, we designed our experimental conditions based on PAC (Probably Approximately Correctly) learnability theory.

The theoretical background for our experiments is as follows. Let $N$ be the number of attributes and $K$ the number of literals needed to write down the smallest DNF (Disjunctive Normal Form) description of the target concept. Let $\epsilon$ be the maximum percentage error that can be tolerated during the testing task. The number of learning examples we used is given by the following formula:-

$$\frac{K \times log_2 N}{\epsilon} \tag{1}$$

Qualitatively the formula indicates that we require more training examples as the complexity of the concept increases or the error decreases [Pagallo90].

In our experiments we set $\epsilon = 10\%$ and used 2000 examples to test classification performance. Thus 90% ($2000 \times 0.9 = 1800$ examples) correctness of testing data is the expected learning success rate.

We used the following 3 problems (target concepts) [Pagallo90]; dnf3 (randomly generated DNF, 32 attributes, 6 terms), mx6 (6-multiplexor, 16 attributes with 10 irrelevant attributes), and par4 (4-parity, 16 attributes with 12 irrelevant attributes).

| Name | description | attributes | terms | #training data |
|---|---|---|---|---|
| **dnf3** | random DNF | 32 | 6 | 1650 |
| **def.** | $x_1x_2x_6x_8x_{25}x_{28}\overline{x_{29}} \vee x_2x_9x_{14}\overline{x_{16}}\ \overline{x_{22}}\ \overline{x_{25}} \vee x_1\overline{x_4}\ \overline{x_{19}}\ \overline{x_{22}}x_{27}x_{28} \vee \overline{x_2}x_{10}x_{14}\overline{x_{21}}\ \overline{x_{24}} \vee x_{11}x_{17}x_{19}x_{21}\overline{x_{25}} \vee \overline{x_1}\ \overline{x_4}x_{13}\overline{x_{25}}$ | | | |
| **mx6** | 6-multiplexor | 16 | 4 | 720 |
| **def.** | $x_{13}x_{16}x_1 \vee \overline{x_{13}}x_{16}x_7 \vee x_{13}\overline{x_{16}}x_4 \vee \overline{x_{13}}\ \overline{x_{16}}x_{10}$ | | | |
| **par4** | 4-parity | 16 | 8 | 1280 |
| **def.** | $x_1 \oplus x_5 \oplus x_9 \oplus x_{13}$ (where $\oplus$ is the XOR operator) | | | |

**Table 1 Test Functions**

The number of training data is derived from equation (1); i.e. 1280 ($= \frac{32 \times log_2 16}{0.1}$) training data are given for par4.

Each method was run according to standard operational criteria. The following parameters were used for each method.

| | |
|---|---|
| Population Size | 400 |
| Crossover Rate | 12% |
| Crossover TYPE | One-Point |
| Mutation Rate | 0.1% |
| Payoff Quantity ($R$) | 1000 |
| Decay by Error ($e$) | 80% |
| Bias for # ($G$) | 4.0 |
| Reference | **Boole** [Wilson87] |

**Table 2 Parameters for Classifier Systems**

| | |
|---|---|
| Initial Nodes | 29999 |
| Node Types | AND, OR, LEFT, RIGHT |
| Reference | [Armstrong79,91] |

**Table 3 Parameters for ALN**

| | |
|---|---|
| Learning Rate | 0.01 |
| Momentum | 0.5 |
| # of Hidden Layers | 1 |
| # of Hidden Nodes | 4 (3 for dnf3) |
| Reference | [Gorman88] |

**Table 4 Parameters for Neural Networks**

These parameters were chosen to obtain the most effective learning results after several experimental runs. Learning was terminated after convergence is attained. Thus the numbers of iterations needed in training phases differ for the 3 methods; i.e. $O(10000)$ for NN, $O(1000)$ for CS and $O(100)$ for ALN. However, this number did not necessarily reflect the computational complexity, because each iteration included qualitatively different computations. We executed several independent runs for each test function. The results are shown below.

| | CS | ALN | NN |
|---|---|---|---|
| mx6 | 10 | 10 | 9 |
| par4 | 10 | 10 | 3 |
| dnf3 | 0 | 10 | 0 |

**Table 5 Number of Successes for the 3 Methods**

| | CS | | ALN | | NN | |
|---|---|---|---|---|---|---|
| mx6 | 100.0 | 0.0 | 98.8 | 0.50 | 100.0 | 0.0 |
| par4 | 100.0 | 0.0 | 98.6 | 1.63 | 100.0 | 0.0 |
| dnf3 | — | – | 87.6 | 0.94 | — | – |

**Table 6 Generalization Ability of the 3 methods**

Table 5 shows the number of successes in the learning training data (100% correctness) for ten runs. Table 6 shows the averages and the standard deviations of correctness for the testing data when the training data were learned successfully (note, in Table 6, a "-" indicates that training was unsuccessful). Following equation (1), the success rate for Boolean learning is expected to be above 90%.

The performance results obtained from these empirical studies were as follows:-

1. Although it is widely believed that NN performs boolean concept learning well, no significant superiority of NN was observed. As can be seen in Table 6, NN dose not always succeed in learning the training data. NN shows poor results for mx6 or par4. This is because the distributed representations prevent NN from distinguishing between relevant and irrelevant attributes for mx6 and par4. Dnf3 is a hard problem for NN.

2. CS is superior to the other two methods for mx6 and par4. CS can cope with the irrelevant attributes. Actually CS is successful in acquiring a perfect set of rules for mx6. For instance, the acquired rules are as follows:-

| Condition ($x_1\ x_2\ x_3\ \cdots\ x_{16}$) | Action | Strength |
|---|---|---|
| # # # 0 # # # # # # # # 1 # # 0 | 0 | 5620 |
| # # # # # # 0 # # # # # # 0 # # 1 | 0 | 5526 |
| # # # # # # # # # 1 # # 0 # # 0 | 1 | 5512 |
| # # # # # # 1 # # # # # # 0 # # 1 | 1 | 5503 |
| 1 # # # # # # # # # # # # 1 # # 1 | 1 | 4222 |
| 0 # # # # # # # # # # # # 1 # # 1 | 0 | 4090 |
| # # # 1 # # # # # # # # 1 # # 0 | 1 | 3633 |
| # # # # # # # # # 0 # # 0 # # 0 | 0 | 3060 |

Notice that these rules express the concept of mx6 by using significant bits ($x_1, x_4, x_7, x_{10}, x_{13}, x_{16}$) and ignoring the irrelevant attributes. On the other hand, CS fails to solve dnf3. This is because it is difficult for CS to represent the concept of dnf3 in the form of classifier rules. So many classifier rules are required to express 0-valued actions for dnf3 whereas # (wild-card) works very well for par4 and mx6. Therefore the rule size is an important factor for CS. For mx6 and par4, 400 rules were enough. On the other hand, $O(1000)$ rules were necessary for dnf3.

3. ALN performs better for all 3 tests. However, as can be seen in Table 6 (the average performance is below 90% for dnf3), ALN was not successful in generalizing the training data. This results from the fact that ALN simply memorizes part of the training data, and lacks the ability to generalize. For these reasons, ALN, in general, requires a large number of initial nodes (for instance, [Armstrong91] used $O(60000)$). Although the final node size might well be reasonable ($O(100)$), a small number of initial nodes results in failure. It should be noted that ALN's performance is heavily dependent upon the problem size. For example, ALN failed to solve par5 (5-bit parity problem with 27 irrelevant bits)

We also conducted an experiment with the learning of noisy Boolean concepts. In noisy environments, learning attribute values are inverted from 1 to 0 or from 0 to 1 (with a probability less than 5%). The results are shown below:-

| Noise | Func. | CS | | | | ALN | | | | NN | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Train | | Test | | Train | | Test | | Train | | Test | |
| 0% | mx6 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 98.9 | 0.7 | 99.0 | 1.2 | 98.7 | 1.4 |
| | par4 | 100.0 | 0.0 | 100.0 | 0.0 | 100.0 | 0.0 | 98.6 | 1.3 | 89.1 | 12.7 | 85.9 | 18.3 |
| | dnf3 | 90.0 | 1.8 | 87.8 | 3.1 | 100.0 | 0.0 | 87.6 | 1.6 | 96.7 | 0.8 | 92.7 | 3.0 |
| 2% | mx6 | 100.0 | 0.0 | 100.0 | 0.0 | 96.4 | 0.6 | 95.5 | 3.5 | 95.7 | 1.0 | 95.6 | 1.0 |
| | par4 | 98.2 | 2.3 | 97.1 | 3.3 | 92.6 | 0.8 | 99.9 | 0.3 | 84.8 | 11.4 | 81.9 | 17.2 |
| | dnf3 | 71.0 | 27.7 | 66.2 | 31.9 | 96.4 | 0.6 | 86.4 | 1.2 | 94.5 | 1.4 | 92.4 | 1.7 |
| 5% | mx6 | 98.3 | 2.5 | 98.4 | 2.4 | 90.9 | 1.2 | 99.8 | 0.4 | 90.7 | 1.1 | 90.1 | 0.5 |
| | par4 | 44.8 | 2.2 | 36.6 | 1.8 | 74.4 | 6.7 | 71.5 | 23.7 | 76.8 | 8.8 | 74.7 | 13.1 |
| | dnf3 | 27.0 | 29.0 | 20.3 | 32.2 | 90.1 | 0.8 | 89.9 | 2.5 | 92.0 | 0.8 | 90.7 | 0.8 |

**Table 7 Learning Performances in Noisy Environments**

Table 7 shows the averages and the standard deviations of correctness for training and testing data by using the 3 methods. Noise-free environments (0%) are the same as those used for Table 5 and Table 6, but the averages of correctness for the testing data were taken only when the training data were learned successfully (i.e. 100% correct). On the other hand, in Table 7, the averages and the deviations are calculated over all ten runs so as to estimate the influence of the noise. Some remarkable points from the table are described below.

1. As can be seen in mx6 (2% and 5% noise) and par4 (0% and 2%), overfitting phenomena were observed for ALN.
2. CS has poor records abruptly when the noise level exceeds 2%.
3. Considering their high deviations, the performance of CS is not stable; that is, results of CS are likely to be influenced by noise.
4. NN copes with noise relatively successfully; i.e. so called "graceful degradation" was observed.

The detailed discussions of these experimental results are omitted here for reasons of space limitation.

The empirical studies of the 3 methods demonstrate the difficulties of learning concepts defined over the space of Boolean features. Both approaches, i.e. digital (ALN, CS) or analog (NN), have their own merits and demerits. In order to solve these difficulties, we currently research on a new learning method, which integrates analog and digital approaches. This is an extended version of our implemented system STROGANOFF [Iba93a,b]. We show the experiments conducted so far and discuss the validity of our approach.

# References


[Anthony92] Anthony, M. and Biggs, N. Computational Learning Theory, Cambridge Tracts in Theoretical Computer Science 30, Cambridge, 1992

[Armstrong *et al.*79] Armstrong, W.W. and Gecsei, J. Adaptation Algorithms for Binary Tree Networks, *IEEE TR. SMC*, SMC-9, No.5, 1979

[Armstrong91] Armstrong, W.W. Learning and Generalization in Adaptive Logic Networks, *Artificial Neural Networks*, (T.Kohonen eds.), Elsevier Science Pub., 1991

[Gorman88] Gorman, R. P. and Sejnowski, T. J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets, *Neural Networks*, vol.1, 1988

[Iba *et al.*93a] Iba, H., Kurita, T., deGaris, H. and Sato, T. System Identification using Structured Genetic Algorithms, ETL-TR93-1, to appear in *Proc. of 5th International Joint Conference on Genetic Algorithms*, 1993

[Iba *et al.*93b] Iba, H., Higuchi, T., deGaris, H. and Sato, T. A Bug-Based Search Strategy for Problem Solving, ETL-TR92-24, to appear in *Proc. of 13th International Joint Conference on Artificial Intelligence*, 1993

[Pagallo *et al.*90] Pagallo, G. and Hausslear, D. Boolean Feature Discovery in Empirical Learning, *Machine Learning*, vol.5, 1990

[Quinlan86] Quinlan, J.R., Induction of Decision Trees, *Machine Learning*, vol.1, 1986

[Rumelhart *et al.*86] Rumelhart, D.E. and McClelland, J.L. Parallel Distributed Processing, MIT Press, 1986

[Wilson87] Wilson,S.W. Classifier Systems and the Animat Problem, *Machine Learning*,vol.2, no.3, 1987