Lifetime learning in multi–agent systems: Examining Robustness in Changing Environments

Dara Curran and Colm O'Riordan
Dept. of Information Technology,
National University of Ireland, Galway.

1 Introduction

The two primary evolutionary forces in nature can be described as population learning and lifetime learning. The first examines the effect of genetic inheritance and its role in the behaviour of living things. Populations of creatures evolve through the processes of genetic recombination and mutation. Over long periods, genetic encodings emerge which produce phenotypic traits suitable for a particular environment, such as webbed feet or enhanced vision, giving individuals competitive advantage over others. As generations progress, useful traits will be passed down successive populations resulting in an overall fitness improvement.

Lifetime learning represents each individual's ability to interact and learn from its environment. Creatures which are capable of correctly interpreting novel situations will be more likely to survive the un–predictability of many habitats. Furthermore, creatures which are capable of memory will recall previous errors in similar situations, greatly reducing the risk of repeated damaging behaviour. This learning mechanism is also a driving force of evolution: as a result of fitness gains brought by lifetime learning, the number of creatures capable of lifetime learning increases and the population improves its overall performance.

These two evolutionary forces can be simulated using genetic algorithms and neural networks respectively. Genetic algorithms represent potential problem solutions as genetic codes which are then evaluated for fitness. Pairs of codes are selected in proportion to their fitness and are combined together to produce offspring. These offspring become part of the next generation and the process is repeated. Genetic algorithms have been shown to be useful in a vast variety of problem domains[1, 2].

Neural networks are simplified mathematical models of nervous systems, inspired by the neurons and synapses of living creatures. Neural networks function by reading input patterns from specified input neurons, feeding these pattern values through a succession of weighted synapses linking other neurons, and finally displaying output values at specified output neurons. By examining a network's output pattern and the desired output pattern for a given input, a measure of error can be obtained. This error is then used to alter the weighting value of synapses connecting neurons in the network. One algorithm for performing this weighting adjustment is known as error back propagation. Through a series of training iterations, the overall error of a network is reduced, improving its performance.

The combination of genetic algorithms and neural networks provides a framework for evolutionary experimentation popular with much research including language evolution[3], neural network design optimization [4, 5, 6] and games[7, 8]. To combine the two approaches, the structure of the neural network must be converted to a format which is suitable for the genetic algorithm. For these experiments, a previously developed artificial life simulator was used which employs a process known as marker based encoding[7] to generate genetic encodings of neural network structures and allows both lifetime and population based learning[9, 10].

The next section of this paper briefly outlines the structure and functionality of the artificial life simulator. Section 3, describes the experimental setup used for each experiment. Section 4 shows the results obtained and finally, Section 5 ends with a conclusion.

2 Simulator

The architecture of the artificial life simulator can be seen as a hierarchical structure. At the top-level of the simulator is a command interpreter which allows users to define an experiment's variables including the number of networks, the number of generations to run the experiment, mutation and crossover rates and the actual problem set which the population will be attempting to solve.

The neural network layer takes the variables set using the command interpreter and initializes a given number of neural networks. The layer then performs training and testing of the networks according to the parameters of the experiment. These network memory structures are then passed to the encoding layer which transforms them into genetic code structures for use in the genetic algorithm.

The genetic algorithm layer uses the genetic codes and the data retrieved from the neural network layer's testing of the networks to perform its genetic operators on the population. A new population is produced in the form of genetic codes.

These are passed to the decoding layer which transforms each code into a new neural network structure. These structures are then passed up to the neural network layer for a new experiment iteration. Once the required number of generations has been reached the experiment finishes.

3 Experiment Setup

A population of 200 agents is randomly initialized at the beginning of each experiment. The population is then allowed to forage for food items, and each agent is allocated fitness according to how well it distinguishes between food and poison bit patterns. The food and poison bit patterns used in this set of experiments are the 16 bit patterns of the 4–bit parity problem thus generating an equal number of food and poison elements in the population's environment.

Two experiments are presented here - the first examining the population's performance in the face of gradual changes in food and poison patterns and the second observing the population's behaviour in a environment which suddenly changes completely. Each experiment is divided in two parts – one using lifetime learning and the other not. In order to obtain reliable results, each experiment is carried out 20 times to derive an average population performance. The crossover rate for each experiment was set at 0.6 and the mutation rate at 0.02. Each experiment allowed populations to evolve for 300 generations.

3.1 Gradual Environment Change

In order to simulate a gradual change in the environment, the bit patterns representing food and poison must be altered over time. The approach taken for this experiment is to change one of the sixteen food or poison values every 19 generations. Thus, patterns which once represented food items become poison and the makeup of the environment eventually reverses with respect to its initial parameters. The goal of this experiment is to ascertain to what degree the population is capable of recovering from each iterative change in its environment.

3.2 Sudden Environment Change

The second experiment examines the behaviour of the population in the face of a dramatic change in food and poison patterns. After a period of 150 generations, each of the sixteen patterns is altered simultaneously to represent its opposite, meaning that all food becomes poison and vice-versa - a potentially catastrophic event for the population. The goal of this experiment is to observe whether such a change will force a population into virtual extinction or whether the population is robust enough to recover.

4 Results

4.1 Gradual Environment Change

The results of the first experiment, gradual change without lifetime learning, are illustrated in figure 1. Each change iteration can be clearly seen in the graph as a sudden plunge in fitness at the change generation. This plunge is immediately followed by quite steep fitness gains suggesting that the population does not take long to recover from such events. The level of fitness peaks at around generation 160 before a gradual descent in fitness. This descent is characterized again by sharp falls in fitness but with subsequent climbs slowing down, resulting in an overall loss.

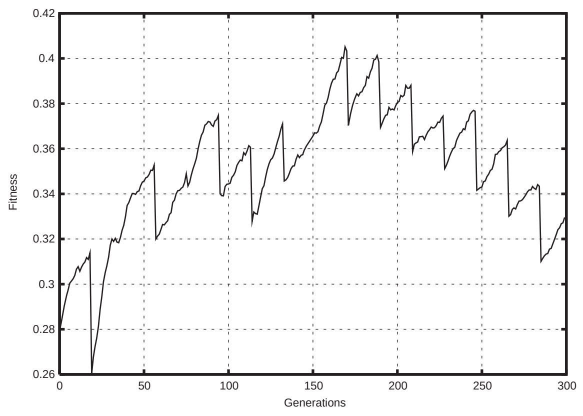

Figure 1. *Gradual Change - No Lifetime Learning*

It is interesting to note that the population peaks at around the half–way mark in the experiment. At this point 8 of the 16 food and poison patterns have been reversed. Therefore, the first eight patterns represent an inverse of the XOR problem, while the second half represents the original XOR problem. It is possible that such a problem set is easier to solve than either the full XOR or the full inverse XOR problems, thereby resulting in higher population fitness. This would explain the gradual drop in fitness following the half–way mark. As the problem begins to resemble the full inverse XOR problem more and more, the population is increasingly unable to solve the problem.

The second experiment immediately shows the gains produced by applied lifetime learning (figure 2). The population achieves much higher levels of fitness than the previous experiment. The first half of the experiment does not show the same sudden drops in fitness (although smaller ones still occur) which were present in the previous experiment results implying that lifetime learning is softening the effects of these changes.

Once again, the population fitness peaks at around the half–way mark, obtaining near optimum levels. Past this level, the drops in fitness become more and more acute, as the problem becomes more difficult for the networks to solve. The gains

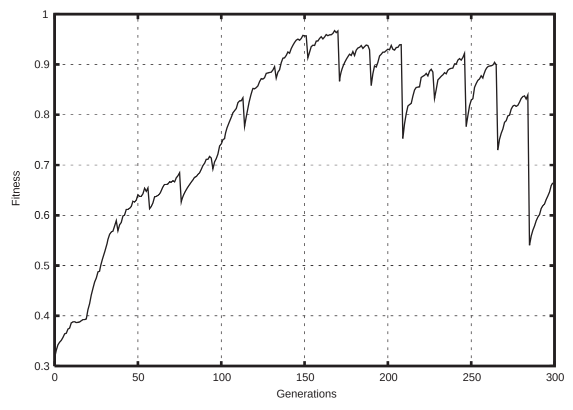

Figure 2. *Gradual Change - Lifetime Learning*

in fitness after each of these drops is significant however, suggesting that given more time between changes, the population would be able to recover.

4.2 Sudden Change

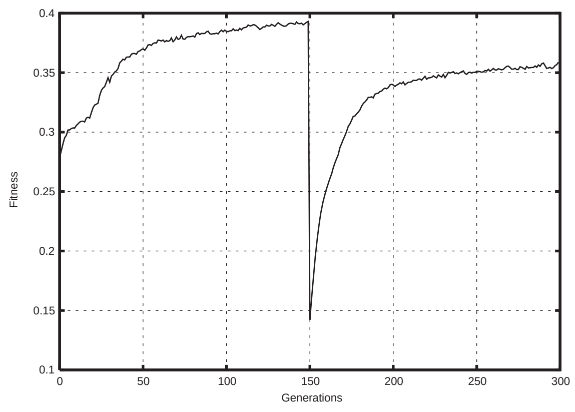

Figure 3. *Sudden Change - No Lifetime Learning*

The first sudden change experiment results are illustrated in figure 3. The population seems capable of achieving moderate levels of fitness without lifetime learning, but the fitness increase slows down considerably as the experiment nears the half-way mark. The sudden change in environment at generation 150 causes a disastrous drop in fitness, down to around 0.15 – much lower than even the randomly generated initial population at generation 1. This low level indicates that the networks are completely incapable of reacting to a change of this magnitude. The population then begins quite a sharp recovery which slows down considerably after generation 200 and stagnates to a level which is considerably lower than the one previously achieved.

The second experiment again shows the benefit of introducing lifetime learning in a population faced with changing environments. The graph in figure 4 shows a steady rise in fitness

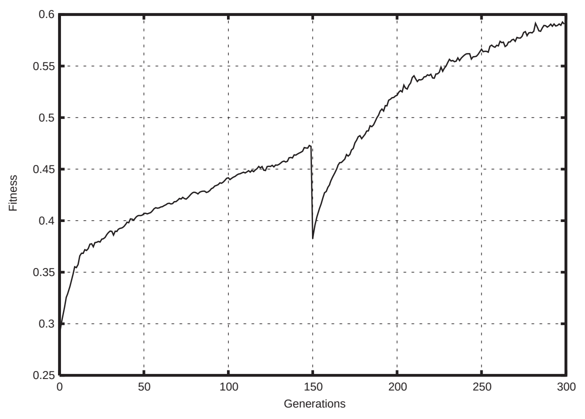

Figure 4. *Sudden Change - Lifetime Learning*

up to generation 150 with a fitness level already surpassing that of the previous experiment. While the drop in fitness at generation 150 is significant, it is not nearly as disastrous as the one seen previously. Compared to a drop of nearly 0.3 in the last experiment, the population's fitness falls a mere 0.07 – showing that the population has been able to adjust to the new problem without the need for any genetic change. The fitness levels quickly rise again and surpass the level obtained prior to the drop, continuing in to increase steadily.

5 Conclusion

These results clearly outline the robustness of a population endowed with the ability to learn from its environment. In each experiment, populations capable of learning were able to sustain their fitness growth. Perhaps the more realistic gradually changing environment proved more of a challenge to the populations. As the experiments near their centre point, both learning and non—learning populations achieved their fitness peaks. This might imply that the evolved network architectures happen to be suitable for the particular problem at this point.

In such an environment, selection would ensure that networks which exhibit the ability to solve similar problems to a reasonable level would propagate through the population. However, once the problem begins to become more difficult, these networks become less valuable as their ability to perform well is impaired. The absence of networks capable of solving the new problem causes the decrease in fitness witnessed in both experiments.

The sudden change experiments highlight the superiority of the lifetime learning approach. Not only is the drop in fitness far less severe in the lifetime learning experiment, but the fitness continues to rise steadily after this point. Perhaps it is easier for evolutionary neural network populations to start afresh with a completely different problem set rather than a continually shifting one. Clearly, an environment which is constantly changing presents far more difficulties to the evolutionary process than one which changes, even dramatically, only rarely. However, it is clear from both experiment sets that the ability to learn during one's lifetime is beneficial to the population as a whole in both types of environment.

6 Acknowledgements

This research is funded by the Irish Research Council for Science, Engineering and Technology.

REFERENCES

[1] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Ann Arbor MI: The University of Michigan Press, 1975.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA, Addison-Wesley, 1989.

[3] B. MacLennan. Synthetic ethology: An approach to the study of communication. In *Artificial Life II: The Second Workshop on the Synthesis and Simulation of Living Systems, Santa Fe Institute Studies in the Sciences of Complexity*, pages 631–635, 1992.

[4] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity, 1990.

[5] R. K. Belew, J. McInerney, and N. N. Schraudolph. Evolving networks: Using the genetic algorithm with connectionist learning. In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Artificial Life II*, pages 511–547. Addison-Wesley, Redwood City, CA, 1992.

[6] Yao. Evolving artificial neural networks. *PIEEE: Proceedings of the IEEE*, 87, 1999.

[7] D. Moriarty and R. Miikkulainen. Discovering complex othello strategies through evolutionary neural networks. *Connection Science*, 7(3–4):195–209, 1995.

[8] Norman Richards, David Moriarty, Paul McQuesten, and Risto Miikkulainen. Evolving neural networks to play go. In *Proceedings of the 7th International Conference on Genetic Algorithms*, East Lansing, MI, 1997.

[9] D. Curran and C O'Riordan. On the design of an artificial life simulator. In R.J.Howlett V.Palade and L.C.Jain, editors, *Proceedings of the Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES 2003)*, University of Oxford, United Kingdom, 2003.

[10] D. Curran and C O'Riordan. Artificial life simulation using marker based encoding. In *Proceedings of the 2003 International Conference on Artificial Intelligence (IC-AI'03)*, Las Vegas, Nevada, USA, 2003.