The Effect of Genotype Representation in the Evolution of Neural Network Connection Weights

James Watson
School of Computer Science and Electrical Engineering
The University of Queensland, QLD 4072 Australia

Abstract

The application of genetic crossover in the evolution of neural networks often gives rise to the 'Competing Conventions Problem' – the tendency of two high performing neural networks to combine to form poorer solutions. In order to determine the extent to which the format of the genetic encoding contributes to this observation, performance comparisons between two different genetic representations are presented. The first encoding format saves the neural network connection weights according to the order they leave the preceding unit layer, while the second keeps the input and output connections of the hidden layer together in the genotype. The performance of networks evolved with the genetic operators of mutation, 1-point crossover and uniform crossover are shown, with no significant differences observed between the two encodings. In addition, no variance was observed between the performances of networks evolved with the three different genetic operators.

1. The Evolution of Neural Networks

Popular training algorithms for neural networks (NN's) such as backpropagation suffer from the inherent complications of gradient-descent techniques – particularly local minima in the error function. Genetic Algorithms (GA's) offer an alternative solution to traditional techniques in that they do not rely on gradient information – they can sample the search space irrespective of where the current solution is located, while remaining biased towards good solutions.

However, the application of GA's (and other evolutionary computing methods) to NN's presents significant hurdles. Unlike other problem domains, where the combination of two good solutions can lead to the creation of another, NN's use distributed representations so that any one connection may not be significant; but in the context of other connections it *is* significant.

The problem of combining two NN's by genetic crossover is variously described as *The Competing Conventions Problem* (Branke, 1995; Friedrich & Moraga, 1996; Hancock, 1992; Krishnan & Ciesielski, 1994; Riley & Ciesielski, 1998) *The Permutation Problem* (Friedrich & Moraga, 1996, Hancock, 1992; Riley & Ciesielski, 1998), *The Isomorphism Problem* (Hancock, 1992), or *The Structural/Functional Mapping Problem* (Hancock, 1992). This issue stems from the fact that even though two neural networks have a different structural composition, they can be functionally equivalent. If crossover attempts to combine two genetic strings whose functionally equivalent aspects are encoded in a different order, the offspring will possibly contain two copies of some aspects, and none of the others.

Due to the difficulties of crossover with NN's, many systems ignore recombination and focus on mutation with a small population (Hancock, 1992). Yao (1999) argues that crossover produces more harm than benefit in evolving NN's. However, Nagao, Agui and Nagahashi (1993) claim results that show uniform crossover

performing better than mutation alone, while Spears and Anand (1991) claim that a GA is better with crossover than without – providing the population size is large enough. Hancock (1992) even claims that permutations are more of a help than a hindrance for GA's, stating that the permutation problem is not as severe as many suppose.

Although it has already been determined that different representations can lead to different network performances (Yao, 1999), no literature has been found that directly compares different genotype representations with the same NN and GA combination. Such comparisons are presented in this paper. The simulations described below were designed to evaluate the relative performances between a system where the encoding is indexed according to outgoing unit connections, versus an encoding that is indexed by the input and output connections of the hidden units. Although the experiments were limited in size and scope (any conclusions offered are only valid for the framework presented below), an analysis of the differences between various genetic encodings can lead to a better understanding of how to specify the genotype of evolving neural networks.

2. Simulation Design and Results

Six experiments were run with identical input data on totally-connected 10-5-1 multi-layer perceptrons. In all experiments, networks were trained with a standard backpropagation algorithm (Rumelhart, Hinton & Williams, 1986). Three different genetic operators were employed: (i) mutation, (ii) 1-point crossover and (iii) uniform crossover. The mutation operator modified each value of the genetic encodings by a number taken from a Gaussian distribution with a mean of 0 and a standard deviation of 0.01. The 1-point crossover operator randomly chose a point along the encodings of two parent networks, and swapped the resulting halves. Uniform crossover implemented a 50% chance of swapping each element of the genotype between two parents. The genetic operators used in each experiment are summarised in Table 1.

Each network in the population was evaluated on its performance in solving the parity bit problem. That is, given a binary set of inputs, the networks were required to learn whether the sum of '1' bits was odd or even. The input sets were comprised of 10 bits (corresponding to the 10 input units of the networks), with the sigmoidal value of the output unit taken as the network's response (< 0.5 for even parity, ≥ 0.5 for odd parity).

Table 1: Genetic operators used in the experiments

Experiment #	Genetic Operators Used
1	None
2	Mutation
3	1-Point Crossover
4	Uniform Crossover
5	Mutation & 1-Point Crossover
6	Mutation & Uniform Crossover

The initial network population was generated with randomly assigned weights and biases (taken from a Gaussian distribution with mean 0 and standard deviation 0.1). Roulette wheel selection (selection whereby the parent's chance of having offspring is proportional to their fitness), coupled with ranking (to promote population diversity by minimizing the differences between agents by using relative population positioning) were used to select the parents of each new generation. Elitism was also employed to maintain good solutions The population was encoded into the two genetic formats summarised below:

Encoding 1:

LWV'S	HBV'S	UWV'S	OBV'S

where

LWV'S = lower weight values
HBV'S = hidden bias values
UWV'S = upper weight values
OBV'S = output bias values

Encoding 2:

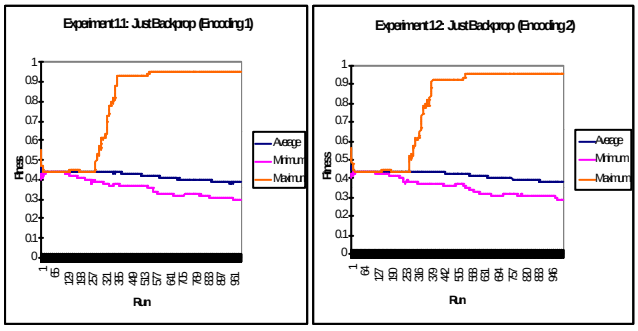
WIH1	BH1	WOH1	WIH2	...	BO

where

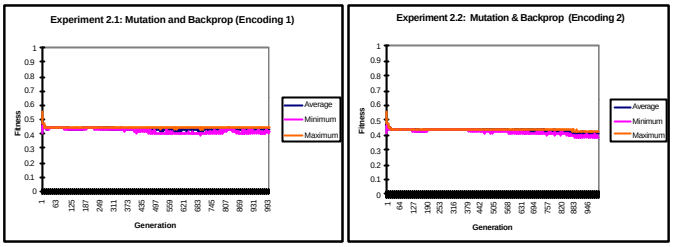
WIH1 = weights into hidden unit 1
BH1 = bias of hidden unit 1
WOH1 = weights out of hidden unit 1
WIH2 = weights into hidden unit 2
BO = biases of output layer

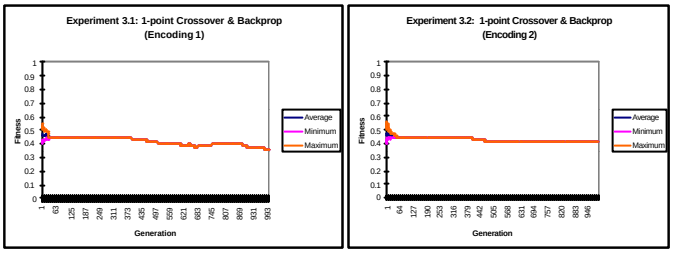
All members of the populations were first trained with backpropagation on a training set of 900 randomly chosen bit sequences, and tested on the remaining 124 10-bit variations.

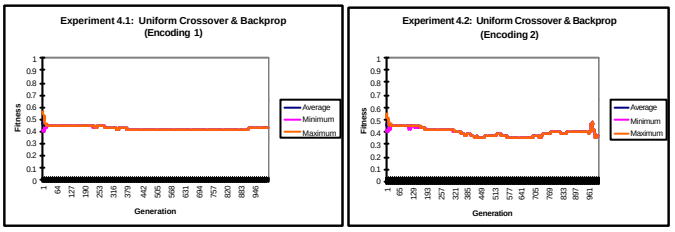
The simulation runs employing just back-propagation returned the network performances shown below (with fitnesses normalised as number of correct responses divided by size of the test set).

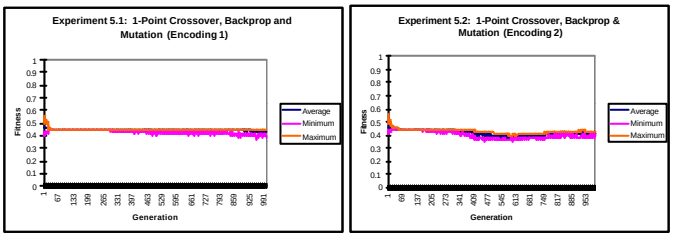


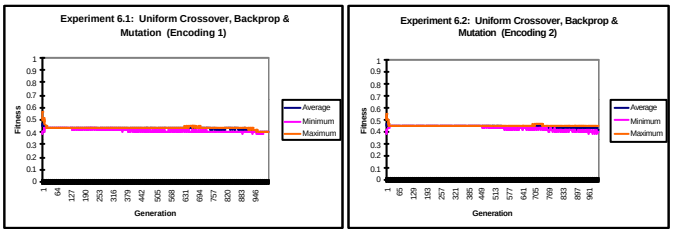
The performances exhibited by the networks of the remaining experiments are as follows:











3. Result Summary and Discussion

As can be seen by the graphs illustrated in Section 2, no significant difference exists between the performance of neural nets genetically encoded with the different formats. Some differences in the stability of agent performance were observed, however these are minor variations and further simulation runs with different parameters are required to make stronger claims on their prevalence.

Note the large variance in performance of the best and worst networks in experiment 1. These graphs show the results of a single population, trained and tested on the same data set 1000 times. It is interesting to note that the minimum fitness value continued to fall, even though backpropagation was repeatedly performed with this data set. Further investigation is needed to determine the reason for this observation, with the first lead to be following the effect the '0' and '1' bit representation had on the training of the network weights (strong weights would have been masked by a 0, then brought back into play with a 1 – poor training ability could have been exhibited by a neural net that was initialized with a detrimental set of random values).

Note too that the average performance of networks trained with backpropagation alone tended to stay around 0.45, which is the same as the predominant value returned by the populations of the other five experiments. This would indicate that the mutation, 1-point crossover and uniform crossover operators did not improve the network populations as generations were evolved, although further work is needed to determine whether or not the trend continues when the simulations are run for many more generations. In addition, a larger population size may need to be introduced since it has been claimed that crossover improves the genetic algorithm, provided the population size is large enough (Spears & Anand, 1991). The results presented above are for populations with 50 members.

The high maximum performance of experiment 1 could be attributed to the neural network population being repeatedly trained on the same data, without the chance of genetic operators altering information structures developed through the learning process (see the rapid rise in maximum fitness in the graph representing Experiment 1 at around run 290). A first conclusion could then be that backpropagation alone finds better solutions more quickly than the operators introduced by evolutionary programming. Before claiming this, however, the nature of the task presented to the networks needs to be fully determined. Whether this problem space suffers from local minima or not is a major determinant in the ability of backpropagation to quickly find an ideal solution (and the relative performance gain garnered from an evolutionary computation paradigm). Such an analysis has not been undertaken in this study, however future work would fully investigate the issue.

4. Conclusions

It was initially thought that the genetic encoding that attempted to preserve hidden layer connections would have superior performance to that of the first encoding. Since there was half the number of hidden units as there were input units, backpropagation was expected to construct a compressed representation at the hidden layer that would allow the networks to generalize to the parity problem – and thus perform well on the unseen test set. This representation had a good chance of being broken up by the crossover operators, the chance being higher in the encoding format that did not attempt to preserve hidden unit connections.

However, no difference was observed between simulations that used either encoding. In addition, no significant differences were observed between simulations that used different genetic operators. Although further work needs to be done to determine to what extent these

observations hold for variations of the task (such as differently sized input sequences, different numbers of units in network layers, different encodings, numbers of generations, etc.), the question has been raised as to whether crossover does indeed affect the performance of evolving populations under *all* conditions, and whether the choice of genotypic representation is always a critical one.

Acknowledgements

I would like to thank Janet Wiles for her discussions and valued input into the project. I would also like to thank the School of Computer Science and Electrical Engineering at the University of Queensland for the use of their computing facilities.

References

Branke, J. (1995). Evolutionary algorithms for neural network design and training. Retrieved 9 August 2000 from ftp://ftp.aifb.uni-karlsruhe.de/pub/jbr/Vaasa.ps

Friedrich, C.M. & Moraga, C. (1996). An evolutionary method to find good building-blocks for architectures of artificial neural networks. In *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems.* pp. 951-956.

Hancock, P.J.B. (1992). Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. Retrieved 9 August 2000 from ftp://nevis.stir.ac.uk/pub/users/pjh/papers/cogann.ps.Z

Krishnan, R. & Ciesielski, V.B. (1994). 2DELTA-GANN: A new approach to training neural networks using genetic algorithms. In Tsoi, A.C. (Ed). *Proceedings of the Fifth Australian Conference on Neural Networks.* pp. 38-41.

Nagao, T. & Agui, T. & Nagahashi, H. (1993). Structural evolution of neural networks having arbitrary connections by a genetic method. In *IEICE Transactions on Information and Systems.* ***E76-D*** (6). pp. 689-697.

Riley, J. & Ciesielski, V.B. (1998). An evolutionary approach to training feed-forward and recurrent neural networks. Retrieved 9 August 2000 from http://goanna.cs.rmit.edu.au/~vc/papers/kes98.ps.Z

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning internal representation by error propagation. In Rumelhart, D.E. & McClelland, J.L. (Eds). *Parallel Distributed Processing* ***1***. MIT Press. pp. 318-362.

Spears, W.M. & Anand, V. (1991). A study of crossover operators in genetic programming. In *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems.* pp. 409-418.

Yao, X. (1999). Evolving artificial neural networks. In *Proceedings of the IEEE*. ***87*** (9). pp. 1423-1447.