Evolving Neural Networks: Is it Really Worth the Effort?

John A. Bullinaria

School of Computer Science, The University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK

Abstract. The idea of using simulated evolution to create neural networks that learn faster and generalize better is becoming increasingly widespread. However, such evolutionary processes are usually extremely computationally intensive. In this paper, I present an empirical study to investigate whether the improved performance obtained really does justify the extra effort, and whether it might be possible to extract some general principles from existing evolved networks that can be applied directly to our hand-crafted networks.

1. Introduction

In the same way that we have taken ideas from biological neural networks to produce powerful artificial neural networks, it is now becoming increasingly common to use ideas from natural evolution to optimise our artificial neural networks [1]. However, such evolutionary approaches are rather computationally intensive, and for individual problems it is often quicker to use traditional cross-validated model selection and wait for slow networks to train, than to evolve a good network model that will learn at a maximal rate. An evolutionary approach that might be more useful, is to evolve neural networks that work well (e.g. learn quickly to generalize well) on a whole *class* of problems, and then apply the evolved results directly to any specific problem that we are presented with. We may even be able to extract some *general principles* that can be applied directly even more widely to the traditional non-evolutionary neural network approach. Alternatively, of course, it may be that the evolutionary approach will just tell us what we already know from decades of practical experience. In this paper I present an empirical study that investigates these issues.

Two of my previous studies suggest that evolving neural networks *can* provide useful new general results. The first involved the evolution of optimal target and sigmoid prime offsets to avoid the well known problem of incorrect output saturation for binary outputs in sigmoidal multi-layer perceptrons trained using gradient descent with the sum-squared error cost function [2]. Unexpectedly, rather than finding minor improvements over the traditionally used values around 0.1 for the offsets, the parameters took on vastly different values, that corresponded to the sum-squared error function evolving into the cross-entropy cost function with no offsets. The general implication was: forget about using offsets, just use cross-entropy from the start. The second study investigated the evolution of neural architectures [3], and revealed that the earlier conclusion [4] that modular architectures were better was wrong, in the sense that evolution resulted in even better non-modular architectures if the learning

algorithm is allowed to evolve at the same time as the architecture. The remainder of this paper will further explore the properties of evolved neural networks.

2. The Evolutionary Approach

My evolutionary simulations follow a nature inspired approach. I take a whole population of individual neural networks, each specified by a set of innate parameters. Then, during each simulated year, they learn from their training data, and compete with each other, with only the fittest being allowed to survive and procreate. By randomly sampling classification training data from a continuous input space with pre-defined classification boundaries, the performance on each year's new training items, before training on them, provides a measure of their generalization ability that can be used as the individual's fitness. Since individuals of different ages (i.e. after different amounts of training) are competing with each other, this provides an incentive to learn quickly, so there is no need to incorporate that explicitly in the fitness. Each year, some of the least fit and some of the oldest die and are removed from the population, to be replaced by children of the fittest members. Each innate parameter of a child is chosen randomly from the range spanned by the corresponding values from its two parents, plus random Gaussian mutation that allows it to end up outside that range. By taking a randomly generated initial population, and following this regime, with appropriate death and mutation rates, for many simulated years, increasingly fit populations consistently emerge [2, 3].

3. Simulation Results

Obviously, to explore explicitly the properties of the evolved networks, we need to specify particular training data and properties to evolve. For simplicity, we draw the input data randomly from the continuous two dimensional space $[0.0, 1.0]^2$, and the target output is one of two classification classes determined according to a random circular boundary in the input space. By assigning a different random circle to each new individual, we eventually evolve neural networks that are able to learn well *any* classification problem of that form, not just one in particular. For this study, our individuals will be standard fully connected one hidden layer feed-forward sigmoidal networks trained by gradient descent with momentum and weight decay using the cross-entropy cost function [2]. The obvious innate parameters in this case are the number of hidden units, the learning rates and momentum, a weight decay parameter, and the initial weight distributions. Traditionally, we use the same learning rate for all components L of the network, but this is unlikely to be optimal. We therefore allow different learning rates η_L for the input to hidden layer weights (η_{IH}), the hidden to output weights (η_{HO}), the hidden unit biases (η_{HB}), and the output unit biases (η_{OB}). If appropriate, evolution will still be free to give them equal values. Similarly, we allow four corresponding different ranges r_L to specify uniform random initial weight distributions $[-r_L, +r_L]$. The number of hidden units will be fixed at 20, which is

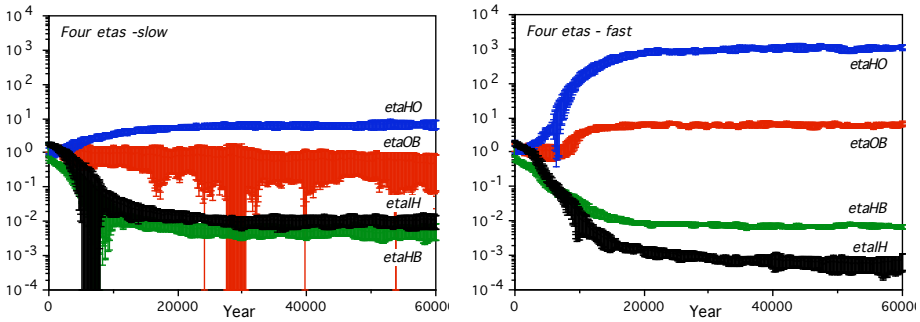

Figure 1: Evolution of the four learning rates, and their variances, for two sets of simulations.

plenty for the given task. Then presenting 1200 training patterns per simulated year leads to time-scales emerging that are comparable to those found in humans.

Starting the initial populations with random parameter values from the range zero to around twice the traditional handcrafted values gets the simulations off to a good start, and evolution soon gives rise to even better values. Obviously, we need to run many simulations, starting with different random number seeds, to check the consistency of the results. The first analysis of 30 runs yielded disappointing results – the variance across runs was enormous! However, further analysis revealed that each run was converging on to one of two rather different 'species'. Figure 1 shows the evolution of the learning rates for each of the two species, and the relatively low variance within each. The most obvious thing to notice is the large (up to six orders of magnitude) difference in the learning rates that emerge for the different layers, and how far removed they all are from the values traditionally used. Unfortunately, the ratios are rather problem dependent, so no general principles here. If we allow only a single learning rate across all layers, the evolved value is 0.19 ± 0.02, which is more in line with the 'default' values traditionally used in handcrafted networks.

The big question, of course, is do our two evolved species really perform any better than the traditional networks? To get reliable statistics, individuals from ten evolved populations of each type were each tested on 50 different classification problems, with different random initial weights in each case. The upper two plots of Figure 2 shows the median and quartile error rates against age for the two species of Figure 1. For comparison, the lower left plot shows the corresponding performance if only a single learning rate is evolved. The fact that any single evolutionary run could produce the inferior of the two species is not ideal, but we see that even the worst performing fully evolved species is still far superior to the traditional networks with a single learning rate, so we are still much better off using evolution.

Although the evolved networks with four independent learning rates appear to be far superior, the enormous learning rates could still be problematic. We really need to check the error distributions more carefully. Figure 3 shows the average error counts, and variances, for the evolved networks as they learn. From this point of view, the slower learning species of Figure 1 no longer looks so bad compared with the faster

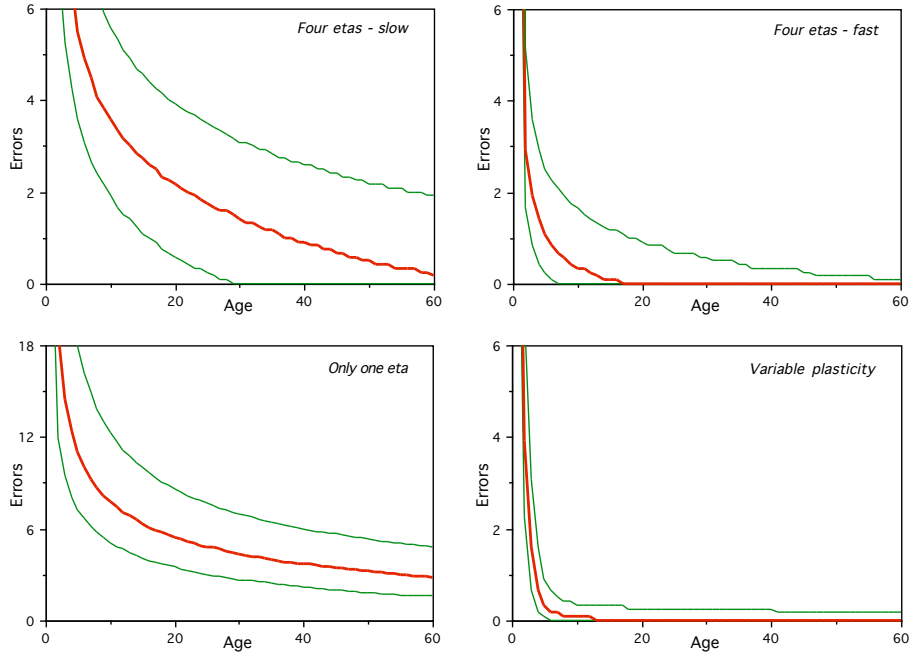

Figure 2: Median and quartile error rates for the evolved individuals: the two Figure 1 species (top), the single learning rate case (bottom left), and with variable plasticity (bottom right).

learning species. The final average performance is actually better, and the variances across individual runs are lower. This suggests that the tails of the error distribution for the fast learners could be hiding a problem. This is confirmed by the average error distributions for individuals between the ages of 50 and 60 shown in Figure 4. The fast learners do better overall, but are prone to many more very large errors, even more than the single learning rate population. A detailed analysis of the large error cases reveals that it is not just a few very poor individuals that are poor on every run, but that many individuals occasionally get dealt a particular random combination of classification problem and initial weights that they perform very poorly on throughout their life. The 'slow but sure' individuals rarely have such problems.

Having evolved fast learners that sometimes perform very much worse than slow learners is not the successful outcome we had hoped for. We need more flexibility in our learning algorithm, so that evolution can avoid this problem. A natural way to do this is to mimic the 'critical periods' for learning found in humans, and allow learning rates that vary with age t. Adaptive learning rates are not new [5], but here we want to evolve a simple robust training data independent pattern of plasticity variation. A natural start is to evolve a two parameter exponential scale factor $s(t)$ of the form

$$s(t) = \beta + (1-\beta)\mathrm{e}^{-t/\tau} \quad , \quad \eta_L(t) = s(t)\,\eta_L(0)$$

in which the baseline β and the time constant τ can take on any positive values.

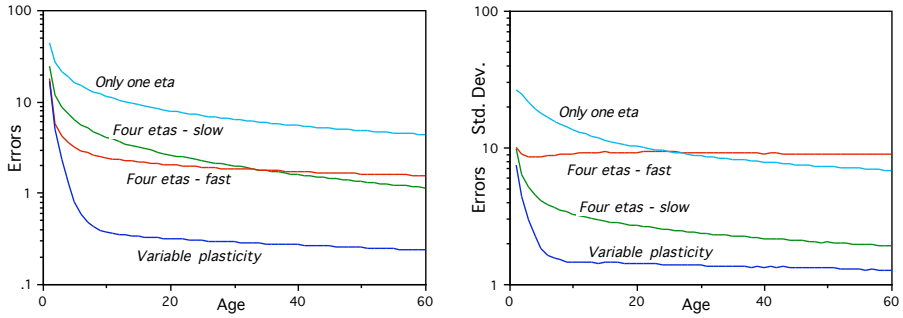

Figure 3: Mean error rates (left) and variances (right) for the evolved populations.

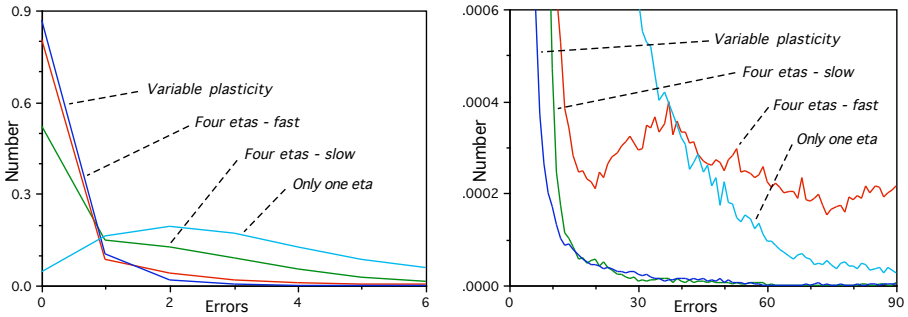

Figure 4: Error distributions for trained evolved individuals: peaks (left) and tails (right).

Repeating the previous simulations with these two additional evolvable parameters resulted in fast learning populations evolving in all 10 out of 10 runs. Figure 5 shows the evolution of the learning rates and scale factor parameters.

We can now look back at all our earlier graphs and consider the performance of the evolved variable plasticity populations. First, in Figure 4 we see that the tail of the error distribution now shows the good performance of the previous slower learners, while the peak is comparable with the good performance of the fast learners. So we appear to have solved the problem of our fast learners showing occasional very poor performance. Moreover, in Figures 2 and 3 we see that the learning speeds and generalization performance are vastly improved in all respects.

A closer inspection of the evolved populations reveals subtle interactions between the evolved parameters. Turning off the variable plasticity, but keeping the evolved values for all the other parameters, results in performance far worse than both species of Figure 1. Taking the average (exponentially falling) plasticity scale factor as a 'general design principle' and inserting it into the fast learning species of Figure 1, while keeping all the other parameters as evolved with no plasticity variation, does solve the problem of occasional very poor performance, and massively improves the average speed of learning, but not to the level of the fully evolved case. If we want to get the best performance out of variable plasticity, we really do need to make appropriate changes to the other parameters too. The larger number of parameters

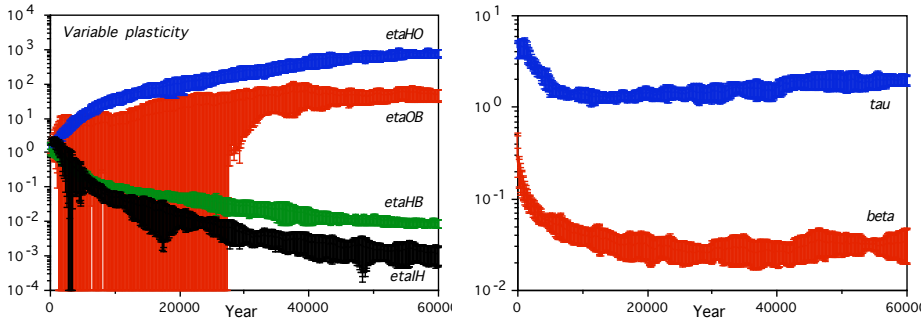

Figure 5: Evolution of age dependent plasticity: learning rates (left), scale parameters (right).

here, and their non-trivial interactions, effectively render traditional model selection infeasible. Evolution has proved to be a good way of finding the right combinations of parameter values, and it will be hard to find a procedure that works better.

4. Conclusions

Evolution has clearly shown that having *four* independent learning rates and initial weight distributions results in far superior learning performance compared with just *one* for the whole network, and how age dependent plasticity can generate further vast improvements. However, given the complex interactions of the evolved parameters, and the extreme values some of them take, it seems unlikely that we will be able to take these observations and dispense with the evolutionary process. So, if we do require fast learning (e.g. for real time autonomous systems), evolution really *is* worth the effort. If we want to avoid evolving everything, the best we can hope for is that it will be possible to evolve some fairly robust parameter configurations that can cope well with new problems from particular identifiable classes of problems.

There exist other evolutionary strategies, whereby whole generations are tested and replaced at each stage, rather than just a few of the oldest and least fit individuals. The study presented here is currently being extended to include those.

References

[1] X. Yao, Evolving Artificial Neural Networks. *Proceedings of the IEEE,* vol. 87, pp. 1423-1447, 1999.

[2] J.A. Bullinaria, Evolving Efficient Learning Algorithms for Binary Mappings. *Neural Networks*, vol. 16, pp. 793-800, 2003.

[3] J.A. Bullinaria, Simulating the Evolution of Modular Neural Systems. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society,* Mahwah, NJ: Lawrence Erlbaum Associates, pp. 146-151, 2001.

[4] J.G. Rueckl, K.R. Cave and S.M. Kosslyn, Why are "What" and "Where" Processed by Separate Cortical Visual Systems? A Computational Investigation. *Journal of Cognitive Neuroscience*, vol. 1, pp. 171-186, 1989.

[5] R.A. Jacobs, Increased Rates Of Convergence Through Learning Rate Adaptation. *Neural Networks*, vol. 1, pp. 295-307, 1988.